ODYSSEUS 2018
Seventh International Workshop on Freight Transportation and Logistics
June 3-8, 2018, Cagliari, Sardinia - Italy

TOPICS OF INTEREST
Topics of interest cover a broad spectrum and include but are not limited to:
- Vehicle Routing
- Fleet and Crew Management
- Modal/Intermodal Transportation
- Intelligent Transportation Systems
- Terminal Management
- Supply Chain Logistics
- Network Design and Planning
- City Logistics
- Facility Logistics
- Humanitarian Logistics

CONTACT DETAILS
For further information, please visit http://convegni.unica.it/odysseus2018
Contact person: Massimo Di Francesco, mdifrance@unica.it
Dear ODYSSEUS 2018 participant,

The Organizing Committee has great pleasure to welcome all participants and accompanying guests to Cagliari (Sardinia, Italy) for the 7th International Workshop on Freight Transportation and Logistics, ODYSSEUS 2018. This workshop is organized by the members of the Department of Mathematics and Computer Science of the University of Cagliari.

ODYSSEUS is a triennial series of international workshops providing a high quality forum on recent developments, trends and advances in the theory, practice and application of mathematical models, methodologies and decision support systems in the field of Freight Transportation and Logistics. This year ODYSSEUS continues to build upon the great success of our earlier meetings in Crete (2000), Sicily (2003), Altea (2006), Cesme (2009), Mykonos (2012) and Ajaccio (2015). It brings together more than 180 academics, researchers and practitioners from around the world to discuss recent experiences, exchange ideas, disseminate research results, and present advanced applications and technologies. ODYSSEUS 2018 has enjoyed an enthusiastic reception from the Freight Transportation and Logistics community, as shown by the large number of submissions. We believe the rigorously selected contributions will continue this tradition of excellence and advance the field.

There are 144 presentations grouped into 50 sessions arranged in three parallel streams. The program covers a wide range of topics, featuring sessions on various facets of Routing (e.g., Vehicle Routing, Arc Routing, Inventory Routing, Location-Routing, Dynamic Routing, Stochastic Routing), Logistics (e.g., Supply Chain Logistics, Humanitarian Logistics) and Transportation Networks (e.g., Freight Transportation Networks, Network Design and Planning), as well as several sessions on emerging topics (e.g., Drones, Car and Bike Sharing). We hope that you will be inspired by the presentations, as well as by the less formal discussions with friends and colleagues participating to ODYSSEUS 2018.

A two-day school for young researchers and PhD students is organized by VeRoLog in conjunction with the workshop. Four lectures will be given within the workshop venues on Friday 1st and Saturday 2nd of June by Teodor Gabriel Crainic, Guy Desaulniers, Ola Jabali, Thibaut Vidal on different methodologies for Vehicle Routing Problems.

Several social events are offered to the conference participants and their guests: a welcome party on Sunday, a guided visit of the city of Cagliari on Monday evening, a visit to the Nuraghe of Barumini and a typical dinner on Tuesday evening, a visit to the beach of Chia and the old city of Nora on Wednesday evening and the conference dinner on Thursday.

We are grateful to the organizations and the individuals that have supported the organization of this event. In particular, we are grateful to the ODYSSEUS 2018 Scientific Committee, as well as to all the individuals who either contributed abstracts or served as reviewers of these abstracts. We wish you all an excellent week, an exciting meeting with informative presentations and stimulating discussions, and an unforgettable staying in Cagliari.

Looking forward to welcoming you in Cagliari, The Organizing Committee of Odysseus 2018
Organizing Committee
- Andrea Loi, Head of the Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Federica Bomboi, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Carlino Casari, CRS4, Cagliari
- Massimo Di Francesco, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Ali Ghezelsoflu, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Enrico Gorgone, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Daniela Lera, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Luca Meloni, Department of Mathematics and Computer Science, Università degli Studi di Cagliari
- Simone Zanda, Department of Mathematics and Computer Science, Università degli Studi di Cagliari

Program Committee
- Teodor Gabriel Crainic, Université du Québec à Montréal and CIRRELMT
- Massimo Di Francesco, Università degli Studi di Cagliari
- Enrico Gorgone, Università degli Studi di Cagliari
- Paola Zuddas, Università degli Studi di Cagliari (chair)

Scientific Committee
- Maria Battarra, University of Bath
- Tolga Bektas, University of Southampton
- Michel Bierlaire, Ecole Polytechnique Fédérale de Lausanne
- Olli Bräysy, University of Jyväskylä
- Ann Melissa Campbell, University of Iowa
- Angel Corberán, Universitat de València
- Jean-François Cordeau, HEC Montréal and CIRRELMT
- Teodor Gabriel Crainic, Université du Québec à Montréal and CIRRELMT
- Guy Desaulniers, Ecole Polytechnique de Montréal and GERAD
- Wout Dullaert, Vrije Universiteit Amsterdam
- Karl F. Dörner, University of Vienna
- Richard Eglese, Lancaster University
- Güneş Erdoğan, University of Bath
- Kjetil Fagerholt, Norwegian University of Science and Technology
- Elena Fernández, Universitat Politècnica de Catalunya
- Antonio Frangioni, Università di Pisa
- Emma Frejinger, Université de Montréal
- Manlio Gaudioso, Università della Calabria
- Michel Gendreau, Ecole Polytechnique de Montréal and CIRRELMT
- Bernard Gendron, Université de Montréal and CIRRELMT
- Bruce Golden, University of Maryland
- Luis Gouveia, Universidade de Lisboa
- Tore Grünert, GTS Systems and Consulting GmbH
- Richard Hartl, University of Vienna
- Geir Hasle, SINTEF
- Michael Hewitt, Loyola University Chicago
- Jose Holguin-Veras, Rensselaer Polytechnic Institute
- Lars Magnus Hvattum, Molde University College
- Manuel Iori, Università degli Studi di Modena e Reggio Emilia
- Stefan Irnich, Johannes Gutenberg University of Mainz
- Nicolas Jozefowiez, Université de Lorraine
- Anton Kleywegt, Georgia Institute of Technology
- Martine Labbé, Université Libre de Bruxelles
- Gilbert Laporte, HEC Montréal and CIRRELMT
- Jesper Larsen, Technical University of Denmark
- Der-Horng Lee, National University of Singapore
- Janny Leung, Chinese University of Hong Kong
• Angélica Lozano, Universidad Nacional Autónoma de México
• Hani Mahmassani, Northwestern University
• Stefan Minner, TUM School of Management
• Benoit Montreuil, Georgia Institute of Technology
• Guido Perboli, Politecnico di Torino
• Raffale Pesenti, Università Ca’ Foscari di Venezia
• Olivier Péton, Ecole des Mines de Nantes
• David Pisinger, Technical University of Denmark
• Marcus Poggi, Pontifical Catholic University of Rio de Janeiro
• Warren B. Powell, Princeton University
• Christian Prins, Université de Technologies de Troyes
• Harilaos Psaraftis, Technical University of Denmark
• Panagiotis Repoussis, Stevens Institute of Technology
• Nicoletta Ricciardi, Universita di Roma “La Sapienza”
• Mikael Rönnqvist, Université Laval
• Juan José Salazar-González, Universidad de La Laguna
• Martin Savelsbergh, Georgia Institute of Technology
• Michael Schneider, RWTH Aachen University
• Frederic Semet, Ecole Centrale de Lille
• Karen Smilowitz, Northwestern University
• Kenneth Sörensen, Universiteit Antwerpen
• Maria Grazia Speranza, Università degli Studi di Brescia
• Anand Subramanian, Universidade Federal da Paraíba
• Roberto Tadei, Politecnico di Torino
• Christos Tarantilis, Athens University of Economics and Business
• Barrett Thomas, University of Iowa
• Paolo Toth, Università di Bologna
• Walter Ukovich, Università di Trieste
• Tom Van Woensel, Eindhoven University of Technology
• Vedat Verter, McGill University
• Thibaut Vidal, Pontifical Catholic University of Rio de Janeiro
• Daniele Vigo, Università di Bologna
• Stefan Voss, Universität Hamburg
• Stein Wallace, Norwegian School of Economics
• Roberto Wolfler Calvo, Università degli Studi di Cagliari
• Paola Zuddas, Università degli Studi di Cagliari
1. Christian Tilk and Michael Forbes. **Branch-and-Cut for the Active-Passive Vehicle Routing Problem** .......................................................... 1

2. Pedro Munari, Twan Dollevoet and Remy Spliet. **A p-step formulation for the CVRP** ........ 5

3. Guy Desaulniers, Timo Gschwind and Stefan Irnich. **Variable fixing based on two-arc sequences in branch-price-and-cut algorithms** ........................................................ 8

4. Timo Gschwind. **A Branch-Price-and-Cut Algorithm for the Order Batching Problem** .... 13

5. Christopher Bayliss, Antonio Martinez-Sykora, Julia Bennell and Christine Currie. **Robust Queue Constrained Packing: a Vehicle Ferry Case Study** .................................................. 17

6. Teodor Gabriel Crainic, Emma Frejinger, Gianluca Morganti and Serge Bisaillon. **Block & Car Planning for Intermodal Rail** ................................................................. 21

7. Maria Battarra, Antonio Rodriguez-Chia and Gunes Erdogan. **Hub Location and Arc Pricing Problem** ...................................................................................................... 25

8. Marilène Cherkesly, Mercedes Landete and Gilbert Laporte. **The Covering Location Problem with Interconnected Facilities** ................................................................. 29

9. Massimo Di Francesco, Manlio Gaudioso, Enrico Gorgone and Ishwar Murthy. **A Lagrangian-based Decomposition Method on a New Formulation of the Capacitated Concentrator Location Problem** ........................................................................ 33


11. Felipe Lagos, Mathias Klapp and Alejandro Toriello. **Branch and Price for Probabilistic Routing** ........................................................................................................ 42
12. Stein W. Wallace, Zhaoxia Guo and Michal Kaut. **High-dimensional dependent random travel times in vehicle routing** .......................................................... 46


15. Jean Bertrand Gauthier and Stefan Irnich. **Machine learning to guide sequential search - Part II** ........................................................................................................ 58

16. Jian Gang Jin, Qiang Meng and Hai Wang. **Column Generation Approach for Feeder Vessel Routing and Synchronization at a Congested Transshipment Port** ........................................... 63

17. Marcello Sammarra, Manlio Gaudioso and M. Flavia Monaco. **A Lagrangian approach to a cross-docking problem with multiple gates** ............................................. 67

18. Alp Arslan and Niels Agatz. **Dynamic Crowd-shipping with Transfers** ................................. 71

19. Aliaa Alnaggar, Fatma Gzara and James Bookbinder. **Distribution Planning with Consolidation: A Two-Stage Stochastic Programming Approach** .............................................. 75

20. Gianfranco Guastaroba, Jean-Francois Cote and M. Grazia Speranza. **The value of integrating loading and routing** .................................................................................. 79

21. Nabil Absi, Claudia Archetti, Stéphane Dauzère-Pérès, Dominique Feillet and M. Grazia Speranza. **Comparing Sequential and Integrated Approaches for the Production Routing Problem** .................................................................................. 83

22. Estèle Glize, Sandra Ulrich Ngueveu and Nicolas Jozeowicz. **Branch-and-price algorithms for bi-objective vehicle routing problems** .................................................... 87

23. Antoine Giret, Yannick Kergosien and Emmanuel Néron. **Solving a multi objective shortest path problem** ................................................................................................. 91


25. Xingyin Wang, Bruce Golden and Edward Wasil. **A Steiner Zone Variable Neighborhood Search Heuristic for the Close-Enough Traveling Salesman Problem** ........................................... 100


27. Mike Hewitt, Natasha Boland, Minh Duc Vu and Martin Savelsbergh. **Solving Time Dependent Traveling Salesman with Time Windows Problems** ..................................... 108

28. Christos Orlis, Roberto Roberti, Nicola Bianchessi and Wout Dullaert. **The Team Orienteering Problem with Overlaps** ................................................................. 112
29. Said Hanafi, Renata Mansini and Roberto Zanotti. **Multi-visit Clustered Team Orienteering Problem** ................................................................. 115

30. Dario Bezzi, Alberto Ceselli and Giovanni Righini. **Dynamic programming for the Electric Vehicle Orienteering Problem with multiple technologies** .................................................. 119

31. Lacomme Philippe and Prins Christian. **An efficient multi-thread metaheuristic for the CARP** .................................................................................. 123

32. Hani Zbib and Sanne Wøhlk. **Clustering Techniques for Very-Large Scale Arc Routing Problems in Curbside Waste Collection** ........................................................................ 127

33. James F. Campbell, Isaac Plana, José M. Sanchis and Ángel Corberán. **The Drone Arc Routing Problem** ........................................................................ 131

34. Mariangela Rosano, Guido Perboli, Teodor Gabriel Crainic and Walter Rei. **Multimodal capacity planning with uncertainty on contract fulfillment and suppliers reliability** .......... 135

35. Eyal Zvi Tenzer and Tal Raviv. **Routing of small parcels in a crowd-sourced network** .... 139

36. Daniele Manerba and Guido Perboli. **A Progressive Hedging approach for a Supplier Selection Problem under Total Quantity Discount and Demand Uncertainty** .................. 142

37. Samuel Pelletier and Fan E. **The electric vehicle routing problem with energy consumption uncertainty** .................................................................................. 146

38. Anirudh Subramanyam, Panagiotis Repoussis and Chrysanthos Gounaris. **Robust Optimization of Heterogeneous Vehicle Routing Problems under Demand Uncertainty** ...................... 150

39. Anirudh Subramanyam, Frank Mufalli, Jose Lainez-Aguirre, Jose Pinto and Chrysanthos Gounaris. **Robust Multi-Period Vehicle Routing under Customer Order Uncertainty** ....... 154

40. Artur Pessoa, Michael Poss, Ruslan Sadykov and François Vanderberk. **Solving the robust CVRP under demand uncertainty** ................................................................................. 158

41. Pamela Nolz, Diego Cattaruzza, Nabil Absi and Dominique Feillet. **Two-echelon distribution with city hub capacity management** ................................................................................ 162

42. Marlin Ulmer. **Incentive Schemes for Same-Day Delivery Routing** ............................... 166

43. Pirmin Fontaine, Teodor Gabriel Crainic, Ola Jabali and Walter Rei. **Tactical planning for Two-Tier City Logistics Systems under disturbances** ............................................. 170

44. Charlotte Köhler, Jan Fabian Ehmke and Ann Melissa Campbell. **Flexibility Schemes for Attended Home Deliveries** ................................................................. 174

45. Alena Otto, Stefan Poikonen and Bruce Golden. **Planning deliveries in disaster relief by truck and drone** ................................................................................................. 179

46. Ohad Eisenhandler and Michal Tzur. **Operational Challenges of a Food Bank in a Gleaning Network** ................................................................................................. 184

47. Selene Silvestri, Marie-Ève Rancourt and Gilbert Laporte. **Optimizing access to drinking water for remote populations affected by the Nepal earthquake** ................................. 188
48. Annelieke C. Baller, Guy Desaulniers, Said Dabia and Wout E.H. Dullaert. The Inventory Routing Problem with Demand Moves ................................................................. 192

49. Pawel Hanczar, Maciek Nowak and Jacek Kaleta. Fuel delivery in large networks .......... 196

50. Cristián E. Cortés, Daniel Herl, Pablo A Rey, Alejandro Cataldo and Leandro Coelho. A Branch-and-Price approach for an inventory and routing problem to address the replenishment of a network of automated teller machines ......................................................... 200

51. C. A. Benjamin Medbøen, Magnus B. Holm, Kjetil Fagerholt and Peter Schütz. Combining Optimization and Simulation for Designing Shortsea Feeder Networks ........................................ 204


53. Simon Belieres, Nicolas Jozefowiez and Frédéric Semet. A Graph Reduction Heuristic Supply Chain Transportation Plan Optimization .......................................................... 212

54. Bruno Bruck, Jean-Francois Cordeau and Emma Frejinger. An integrated approach for inbound train split and container loading in an intermodal railway terminal .......... 216

55. Konstantinos Androutsopoulos, Michael Madas and Vassileios Sitokonstantinou. A Fairness-driven Scheduling Model for Airport Slot Allocation: is there a cost for fairness? .......... 220

56. Nikola Obrenović, Virginie Lurkin, Michel Bierlaire and Vincent J. Baeriswyl. Capacity-Oriented Marshaling and Shunting Yards Location Problem ...................................................... 224

57. Nicholas D. Kullman, Justin C. Goodson and Jorge E. Mendoza. Dynamic Electric Vehicle Routing with Mid-route Recharging and Uncertain Availability ........................................... 228

58. Szymon Albinski and Stefan Minner. A Data-Driven Approach to the Vehicle Routing Problem with Time Windows under Uncertain Travel and Service Times ........................................ 233

59. Edison Avraham and Tal Raviv. The stochastic and time-dependent single vehicle field service personnel routing and scheduling problem ......................................................... 237

60. Maciek Nowak, Mike Hewitt, Bogumil Kaminski and Michal Pliszka. Strategic network design at the Polish Post ........................................................................................................ 242

61. Giacomo Lanza, Teodor Gabriel Crainic, Walter Rei and Nicoletta Ricciardi. A Study on Travel Time Stochasticity in Service Network Design with Quality Targets ........................................ 246


64. Merve Keskin, Laporte Gilbert and Bülent Çatay. Electric Vehicle Routing Problem with Time Dependent Waiting Times at Recharging Stations .................................................. 258
<table>
<thead>
<tr>
<th>Page</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>Aurélien Froger, Jorge E. Mendoza, Ola Jabali and Gilbert Laporte</td>
<td>Modeling and solving the electric vehicle routing problem with nonlinear charging functions and capacitated charging stations</td>
</tr>
<tr>
<td>66</td>
<td>Claudia Archetti, Leandro C. Coelho, Maryam Darvish and M. Grazia Speranza</td>
<td>Flexible two-echelon location-routing</td>
</tr>
<tr>
<td>67</td>
<td>Elena Fernández, Gilbert Laporte and Jessica Rodríguez Pereira</td>
<td>Models and solution algorithms for Location-Arc Routing Problems</td>
</tr>
<tr>
<td>68</td>
<td>Shu Zhang, Ann Campbell and Jan Ehmke</td>
<td>Impact of Congestion Pricing Schemes on Costs and Emissions of Commercial Fleets in Urban Areas</td>
</tr>
<tr>
<td>69</td>
<td>Marcel Turkensteen and Geir Hasle</td>
<td>Carbon emission effects of consolidating shipments taking topological effects and temporal constraints into consideration</td>
</tr>
<tr>
<td>70</td>
<td>Denise Holfeld, Axel Simroth, Yuanyuan Li, Daniele Manerba and Roberto Tadei</td>
<td>Risk Analysis for synchro-modal freight transportation: the SYNCHRO-NET approach</td>
</tr>
<tr>
<td>71</td>
<td>Yunfei Wang, Ioana Bilegan, Teodor Gabriel Crainic and David Duvivier</td>
<td>A reactive decision support system for freight intermodal transportation - a Revenue Management perspective</td>
</tr>
<tr>
<td>72</td>
<td>Qie He, Stefan Irnich and Yongjia Song</td>
<td>Branch-Cut-and-Price for the VRP with Time Windows and Convex Node Costs</td>
</tr>
<tr>
<td>73</td>
<td>Alexander Döge, Guy Desaulniers, Markus Frey and Rainer Kolisch</td>
<td>An Exact Algorithm for the Vehicle Routing Problem with Time Windows and Flexible Delivery Locations</td>
</tr>
<tr>
<td>74</td>
<td>Ali Alyasiry, Michael Forbes and Michael Bulmer</td>
<td>An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and its Variants</td>
</tr>
<tr>
<td>75</td>
<td>Kevin Dalmeijer and Guy Desaulniers</td>
<td>A branch-price-and-cut algorithm for the TWAVRP -- addressing symmetry in route synchronization</td>
</tr>
<tr>
<td>76</td>
<td>Monica Gentili, Pitu Mirchandani and Alessandro Agnetis</td>
<td>Locating and Scheduling Drones for Delivering Perishable Items</td>
</tr>
<tr>
<td>77</td>
<td>Stefan Poikonen and Bruce Golden</td>
<td>Ship and Drone Routing Problems</td>
</tr>
<tr>
<td>78</td>
<td>Barrett Thomas, Xinwei Chen and Marlin Ulmer</td>
<td>AlphaGo and Amazon: Reinforcement Learning for Same-Day Delivery with a Heterogeneous Fleet of Drones and Vehicles</td>
</tr>
<tr>
<td>79</td>
<td>Martin Savelsbergh, Iman Dayarian and John-Paul Clarke</td>
<td>Same-Day Delivery with Drone Resupply</td>
</tr>
<tr>
<td>80</td>
<td>Kamilla Bolstad, Manu Joshi, Magnus Stålhamne and Lars Magnus Hvattum</td>
<td>Optimizing the vessel fleet size and mix to perform maintenance at offshore wind farms</td>
</tr>
<tr>
<td>81</td>
<td>Bo Dong, Marielle Christiansen, Kjetil Fagerholt and Tolga Bektaş</td>
<td>Combined Fleet Deployment and Inventory Management with Flexibility in Port Choice</td>
</tr>
<tr>
<td>82</td>
<td>Gabriel Homsi, Thibaut Vidal, Rafael Martinelli and Kjetil Fagerholt</td>
<td>Ship routing and speed optimization with heterogeneous fuel consumption profiles</td>
</tr>
</tbody>
</table>
83. David Franz Koza. **Liner shipping service scheduling and cargo allocation** ………… 335

84. Raphael Kramer, Jean-François Cordeau and Manuel Iori. **A rich vehicle routing problem for pharmaceutical distribution** …………………………………………………………………………………………………………………………… 339

85. Mauricio Cerda, Cristián E. Cortés, Pablo A Rey and Zdenko Koscina. **GRASP heuristic scheme for solving a real case ready mixed concrete dispatching problem** ………………… 343

86. Amalia Nikolopoulou, Panagiotis Repoissis and Christos Tarantilis. **The Vehicle Routing Problem with Cross-docking and Capacity Smoothing Restrictions** …………………… 347

87. Hugo Simao and Warren Powell. **Decomposition Methods for Dynamic Load Planning and Driver Management in LTL Trucking** ………………………………………………………………………………… 351

88. Florian Martin, Vera Hemmelmayr and Tina Wakolbinger. **Service level selection and pricing for multimodal package distribution networks** ……………………………………… 355

89. Ramon Auad, Damian Reyes, Martin Savelsbergh and Alan Erera. **Managing Demand in Dynamic Delivery Operations** ……………………………………………………………………………………. 359

90. Alexis Bretin, Guy Desaulniers and Louis-Martin Rousseau. **The Traveling Salesman Problem in Postal Services** ………………………………………………………………………………………………. 364

91. Edoardo Fadda, Roberto Tadei, Guido Perboli and Lohic Fiotio Tiotso. **The multi-path Traveling Salesman Problem with dependent random cost oscillations** …………………………… 368

92. Valentina Cacchiani, Carlos Contreras Bolton and Paolo Toth. **Algorithms for the Traveling Salesman Problem with Time-Dependent Service Times** …………………………………………………………………………………………… 372

93. Majid Salavati-Khoshgalb, Ola Jabali, Walter Rei and Michel Gendreau. **An Exact Algorithm to Solve the Vehicle Routing Problem with Stochastic Demands under an Optimal Restocking Policy** ………………………………………………………………………………………………. 376

94. Justin Goodson, Marlin Ulmer and Barrett Thomas. **Dynamic Time Window Allocation and Sizing for Service Routing Applications** ……………………………………………………………………………………………… 380

95. Fausto Errico, Guy Desaulniers, Andrea Lodi and Borzou Rostami. **Exact and approximate solution methods for the vehicle routing problem with stochastic and correlated travel times** ……………………………………………………………………………………………………………………………… 384


98. Marta Toschi, Ana María Anaya-Arenas, Ettore Lanzarone, Valérie Bélanger and Angel Ruiz. **A matheuristic for the Biomedical Sample Transportation Problem with Interdependent Pickups** ……………………………………………………………………………………………………………………………… 397

99. Maurizio Bruglieri, Simona Mancini and Ornella Pisacane. **A Combinatorial Benders' Cuts based Exact Method for the Multi Trip Containers Drayage Problem** ………………………………………………………………………………………………………………… 401
<table>
<thead>
<tr>
<th>No.</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.</td>
<td>Arturo E. Pérez Rivera and Martijn R.K. Mes.</td>
<td><strong>Integrated Scheduling of Drayage and Long-haul Transportation in Synchromodality</strong></td>
</tr>
<tr>
<td>101.</td>
<td>Bernard Zweers, Sandjai Bhulai and Rob van der Mei.</td>
<td><strong>Optimizing barge utility in hinterland container transportation</strong></td>
</tr>
<tr>
<td>102.</td>
<td>Francesco Bertoli and Philp Kilby.</td>
<td><strong>An Analysis of Territory Design For Routing Problems with Time Windows and Balance Requirements</strong></td>
</tr>
<tr>
<td>103.</td>
<td>Fabien Lehuédé, Olivier Péton and Fabien Tricoire.</td>
<td><strong>Multi-directional local search for the leximax-VRP</strong></td>
</tr>
<tr>
<td>105.</td>
<td>Demetrio Laganà, Enrique Benavent, Ángel Corberán and Francesca Vocaturo.</td>
<td><strong>A Heuristic Algorithm for the Periodic Rural Postman Problem with Irregular Services</strong></td>
</tr>
<tr>
<td>106.</td>
<td>Sanne Wøhlk and Gilbert Laporte.</td>
<td><strong>A Districting-Based Heuristic for the Coordinated Capacitated Arc Routing Problem</strong></td>
</tr>
<tr>
<td>107.</td>
<td>Asvin Goel and Christian Tilk.</td>
<td><strong>Bidirectional labelling for the European Union Truck Driver Scheduling Problem</strong></td>
</tr>
<tr>
<td>108.</td>
<td>Samuela Carosi, Antonio Frangioni, Laura Galli, Leopoldo Girardi and Giuliano Vallese.</td>
<td><strong>A Lagrangian Heuristic for Integrated Timetabling and Vehicle Scheduling</strong></td>
</tr>
<tr>
<td>109.</td>
<td>Homero Larrain, Leandro Coelho, Claudia Archetti and Maria Grazia Speranza.</td>
<td><strong>A VMND algorithm for the multi-period vehicle routing problem with due dates</strong></td>
</tr>
<tr>
<td>110.</td>
<td>Teobaldo Bulhões, Minh Hoang Hà, Rafael Martinelli and Thibaut Vidal.</td>
<td><strong>The vehicle routing problem with service level constraints</strong></td>
</tr>
<tr>
<td>111.</td>
<td>Wenjuan Gu, Diego Cattaruzza, Maxime Ogier and Frederic Semet.</td>
<td><strong>Adaptive large neighborhood search for multicommodity VRP</strong></td>
</tr>
<tr>
<td>112.</td>
<td>Alessandro Hill, Stefan Voss and Roberto Baldacci.</td>
<td><strong>Vehicle Routing Problems with Matches and Conflicts</strong></td>
</tr>
<tr>
<td>113.</td>
<td>Emilio Jose Alarcon Ortega, Michael Schilde and Karl Franz Doerner.</td>
<td><strong>Matheuristic for the Consistent Inventory Routing Problem with Time Windows</strong></td>
</tr>
<tr>
<td>114.</td>
<td>Annarita De Maio, Leandro C. Coelho and Demetrio Laganà.</td>
<td><strong>A Variable MIP Neighborhood Search for the Multi-attribute Inventory Routing Problem</strong></td>
</tr>
<tr>
<td>115.</td>
<td>Fabián Penaranda, Marcus Poggi and Rafael Martinelli.</td>
<td><strong>A Benders Decomposition for the Inventory Routing Problem</strong></td>
</tr>
<tr>
<td>116.</td>
<td>Claudia Archetti, Maria Grazia Speranza, Maurizio Boccia, Antonio Sforza and Claudio Sterle.</td>
<td><strong>A branch and cut algorithm for single and multiple vehicle IRP with pickups and deliveries</strong></td>
</tr>
<tr>
<td>117.</td>
<td>Margaretha Gansterer, Richard F. Hartl and Rudolf Vetschera.</td>
<td><strong>Auction-based mechanisms in carrier collaborations: challenges and limitations</strong></td>
</tr>
<tr>
<td>No.</td>
<td>Authors</td>
<td>Title</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>118</td>
<td>Mario Guajardo, Ondrej Osicka and Thibault van Oost</td>
<td>Collaborative location-routing</td>
</tr>
<tr>
<td>119</td>
<td>Joydeep Paul, Niels Agatz and Martin Savelsbergh</td>
<td>Shared Capacity Routing Problem with Transfer Points</td>
</tr>
<tr>
<td>120</td>
<td>Mathijs van Zon, Remy Spliet and Wilco van den Heuvel</td>
<td>Cost allocation in cooperative transport: The Joint Network Vehicle Routing Game</td>
</tr>
<tr>
<td>121</td>
<td>Peng Sun, Lucas Veelenturf, Tom Van Woensel and Mike Hewitt</td>
<td>The time dependent pickup and delivery problem with time windows</td>
</tr>
<tr>
<td>122</td>
<td>Dominik Goeke, Roberto Roberti and Michael Schneider</td>
<td>Heuristic Solution of the Consistent Vehicle-Routing Problem</td>
</tr>
<tr>
<td>123</td>
<td>Nicola Bianchessi, Michael Drexl and Stefan Irnich</td>
<td>The SDVRP with Time Windows and Customer Inconvenience Constraints</td>
</tr>
<tr>
<td>124</td>
<td>Jean Bertrand Gauthier, Jacques Desrosiers and Marco Lübbecke</td>
<td>A strongly polynomial Contraction-Expansion algorithm for network flow problems</td>
</tr>
<tr>
<td>125</td>
<td>Markus Leitner, Ivana Ljubic, Martin Riedler and Mario Ruthmair</td>
<td>The Directed Network Design Problem with Relays</td>
</tr>
<tr>
<td>126</td>
<td>Karen Smilowitz, Liwei Zeng and Sunil Chopra</td>
<td>Leveraging network structure in covering path problems: An application to school bus routing</td>
</tr>
<tr>
<td>127</td>
<td>Henrik Andersson, Carl Axel Folkestad, Nora Hansen, Kjetil Fagerholt and Giovanni Pantuso</td>
<td>A Memetic Algorithm for Optimal Charging and Repositioning of Electric Vehicles in a Free-Floating Carsharing System</td>
</tr>
<tr>
<td>128</td>
<td>Georg Brandstätter, Markus Leitner and Mario Ruthmair</td>
<td>Branch-and-price for charging station placement in free-floating electric car sharing systems</td>
</tr>
<tr>
<td>129</td>
<td>Burak Boyacı and Konstantinos G. Zografos</td>
<td>Considering Spatial and Temporal Flexibility in Optimizing One-Way Electric Carsharing Systems</td>
</tr>
<tr>
<td>130</td>
<td>Claudia Archetti, Dominique Feillet, Andrea Mor and M. Grazia Speranza</td>
<td>Dynamic travelling salesman problem with uncertain release dates</td>
</tr>
<tr>
<td>131</td>
<td>Demetrio Laganà, Gilbert Laporte and Francesca Vocaturo</td>
<td>A Dynamic Multi-Period General Routing Problem</td>
</tr>
<tr>
<td>132</td>
<td>Adria Soriano, Margaretha Gansterer and Karl F. Dörner</td>
<td>A multi-graph approach for the periodic vehicle routing problem with time spread constraints</td>
</tr>
<tr>
<td>133</td>
<td>Dimitris Paraskevopoulos, Panagiotis Repoussis and Christos Tarantilis</td>
<td>Districting of Urban Service Networks</td>
</tr>
<tr>
<td>134</td>
<td>Yuan Yuan, Diego Cattaruzza, Maxime Ogier and Frederic Semet</td>
<td>Last mile delivery problem: the one-vehicle case</td>
</tr>
<tr>
<td>135</td>
<td>Tolga Bektas, Antonio Martinez-Sykora and Tom Cherrett</td>
<td>Clustered routing for last-mile goods deliveries in urban areas: Formulation and a branch-and-cut algorithm</td>
</tr>
</tbody>
</table>
136. Francesca Maggioni, Rossana Cavagnini, Luca Bertazzi and Mike Hewitt. Multi-objective stochastic optimization models for managing a bike sharing system ........................................ 554

137. Mauro Dell'Amico, Manuel Iori, Stefano Novellani and Ananad Subramanian. The Bike sharing Rebalancing Problem with Stochastic Demands ........................................ 558


139. Konstantinos Androutsopoulos and Nikolaos Kamaras. Modeling and Solving a Real-Life Fuel Delivery Problem with Safety Considerations ........................................ 566

140. Luigi De Giovanni, Nicola Gastaldon and Filippo Sottovia. Express delivery in freight transportation: an application to the trucking industry ........................................ 570

141. Jamie Fairbrother and Konstantinos Zografos. Introducing flexibility and demand-based fairness in slot scheduling decisions ................................................................. 574

142. Masoud Mirzaei, Nima Zaerpour and René de Koster. An Integrated Cluster-Based Storage Assignment .............................................................................................................. 578

143. Annunziata Esposito Amideo, Maria Paola Scaparra, Antonio Sforza and Claudio Sterle. An integrated user-system approach for shelter location and evacuation routing ................. 582

144. Teodor Gabriel Crainic, Massimo Di Francesco, Enrico Gorgone and Paola Zuddas. A multicommodity location routing problem in a maritime urban area ........................................ 586
Branch-and-Cut for the Active-Passive Vehicle Routing Problem

Michael Forbes
School of Mathematics and Physics, Faculty of Science,
The University of Queensland, St Lucia, Brisbane, Australia

Christian Tilk
Chair of Logistics Management, Johannes Gutenberg University Mainz,
Jakob-Welder-Weg 9, D-55128 Mainz, Germany.
Email:tilk@uni-mainz.de

1 Introduction

Many applications in the area of transport logistics involve problems in which the execution of transport requests calls for a joint operation of active vehicles such as trucks, tractors, and passive vehicles, e.g., trailers, semitrailers, and containers. While active vehicles can move on their own from one location to another, passive vehicles require an active vehicle for being repositioned. The active-passive vehicle routing problem (APVRP) was introduced in [1]. It explicitly distinguishes active and passive vehicles enabling a flexible modeling of complex transport operations in which an active vehicle carries a passive vehicle to some pickup location, drops it off there, and leaves this location for performing transports of other passive vehicles elsewhere. Later, when the passive vehicle has been loaded, the same or some other active vehicle returns to the customer, picks up the container, and carries it to the delivery location. This flexibility raises the need to synchronize the operations and the movements of active and passive vehicles in time and space [2].

The APVRP can be formally described as follows: Let $A$ be a set of $K$ active vehicles, $P$ be a set of passive vehicles and $R$ be a set of pickup-and-delivery requests. The set of locations $L$ contains the start and end depot of all active vehicles ($o$ and $d$), start and end location of all passive vehicles ($o_p$ and $d_p$), and the pick-up and delivery locations of all requests ($\ell_r^+$ and $\ell_r^-$). The travel time between two location $(i, j) \in L$ is given by $t_{ij}$ and the travel distance by $c_{ij}$. Each request $r \in R$ has a time window $[a_r, b_r]$ and unserved requests imply a penalty. Times for loading and unloading a request $r$ are given by $s_{r}^+$ and $s_{r}^-$, respectively. The objective is to minimize a weighted sum of the total distance traveled, the total completion time of the routes, and the number of unserved requests. The respective weights are $\alpha, \beta, \gamma \in \mathbb{R}_+$. To fulfill a request $r \in R$, an active vehicle must
carry a passive vehicle to $\ell^+_r$ for loading. Each passive vehicle can load only one request at a time, and each active vehicle can transport only one passive vehicle at a time. The loaded passive vehicle must then be transported directly to $\ell^-_r$ for unloading. Afterwards, the empty passive vehicle must be carried away from $\ell^-_r$. Note that these three operations can be performed by up to three different active vehicles.

The contribution of the paper at-hand is twofold. First, we present an exact solution approach for the APVRP that is based on combinatorial Benders cuts [3]. Second, we show how this approach can be generalized to deal with other vehicle-routing problems with timing aspects and synchronization constraints, especially for the more complicated cases in which completion time or duration of routes is part of the objective.

2 Solution Approach

We introduce a Benders reformulation of the APVRP based on a compact formulation that we omit here for the sake of brevity. Therein, we model the route of a passive vehicle $p \in P$ as a sequence of locations. Hence, the set of passive-flow variables is given by $X := \{X^p_{ij} : p \in P, (i,j) \in E^p\}$ with $X^p_{ij}$ indicating that passive vehicle $p$ moves from location $i$ to location $j$ and $E^p := \{(o_p, d_p), (o_p, l^+_r), (l^-_r, l^-_r), (l^-_r, l^+_q), (l^-_r, d_p) : \forall r, q \in R\}$ denoting the set of feasible arcs. Binary variables $U_r$ for all $r \in R$ indicate if request $r$ remains unserved.

The route of an active vehicle is modelled by a sequence of tasks. The set of tasks is defined as $\Gamma := \{(p, i, j) : X^p_{ij} \in X\} \cup \{X_{oo} := \tau_o, X_{dd} = \tau_d\}$, where $\tau_o$ and $\tau_d$ modelling the start and the end of a route. Henceforth, we use abbreviation $\tau$ for $(p, i, j) \in \Gamma$ and $\tau'$ for $(p', i', j') \in \Gamma$. Now, the set of active-flow variables is given by $Y := \{Y_{\tau\tau'} : (\tau, \tau') \in \Gamma \times \Gamma, \tau \neq \tau'\}$ with $Y_{\tau\tau'}$ indicating that an active vehicle transports passive vehicle $p'$ from $i'$ to $j'$ immediately after it has transported passive vehicle $p$ from $i$ to $j$. The Benders Master Problem reads as follows:

$$\begin{align*}
\text{min} & \quad \alpha \left( \sum_{p \in P} \sum_{(i,j) \in E^p} c_{ij}X^p_{ij} + \sum_{(\tau,\tau') \in \Gamma \times \Gamma} c_{ji'}Y_{\tau\tau'} \right) + \gamma \sum_{r \in R} U_r \\
\text{s.t.} & \quad U_r + \sum_{p \in P} X^p_{\tau_o, l^-_r} = 1 & r \in R \\
& \quad \sum_{r \in R} X^p_{o_p, l^+_r} + X^p_{o_p, d_p} = 1 & p \in P \\
& \quad \sum_{(i,j) \in E^p} X^p_{ij} = \sum_{(h,i) \in E^p} X^p_{hi} & p \in P, i \in L \setminus \{o_p, d_p\} \\
& \quad \sum_{\tau \in \Gamma} Y_{\tau\tau} = K \\
& \quad \sum_{\tau \in \Gamma} Y_{\tau\tau'} = \sum_{\tau' \in \Gamma} Y_{\tau\tau'} & \tau \in \Gamma \setminus \{\tau_o, \tau_d\} \\
& \quad \sum_{\tau' \in \Gamma} Y_{\tau'(p,i,j)} = X^p_{ij} & X^p_{ij} \in X \\
& \quad X^p_{ij}, Y_{\tau\tau'}, U_r \in \{0, 1\} & X^p_{ij} \in X, Y_{\tau\tau'} \in Y, r \in R
\end{align*}$$

2
1a is the objective function. Constraints 1b ensures that each request is either performed exactly once or a penalty is paid. Constraints 1c and 1d define the journey of passive vehicles. The number of active vehicles is limited by 1e. Flow conservation of active vehicles is ensured by constraints 1f and constraints 1g couples the flow of active and passive vehicles. The variable domains are given by 1h.

Let \( \bar{S} = (\bar{X}, \bar{Y}, \bar{U}) \) be the set of variables with value 1 in an integer solution of 1 with solution value \( \bar{z}_{\text{master}} \). Since Formulation 1 disregards timing aspects, we have to check if \( \bar{S} \) violates them. Thus, we solve the corresponding Benders subproblem that utilizes the variables set \( T := \{ T_\tau : Y_{\tau_\tau'} \in \bar{Y} \} \) with \( T_\tau \) indicating the start time of task \( \tau \).

\[
\min \sum_{\tau : Y_{\tau_\tau'} \in \bar{Y}} T_\tau + t_{ij} + t_{jd} \\
T_\tau + t_{ij} + t_{ji} \leq T_{\tau'} Y_{\tau_\tau'} \in \bar{Y}, \tau' \neq d \\
T_{(p,i,\ell_\tau^+)} + t_{\ell_\tau^+} \leq T_{(p,i,\ell_\tau^-)} Y_{(p,i,\ell_\tau^-_\tau')} \in \bar{Y} \\
T_{(p,\ell_\tau^+,\ell^-_\tau)} + t_{\ell^-_\tau} \leq T_{(p,\ell^-_\tau,j)} Y_{(p,\ell^-_\tau,j_\tau')} \in \bar{Y} \\
a^-_\tau \leq T_\tau \leq b^-_\tau - t_{ij} \quad Y_{\tau_\tau'} \in \bar{Y}
\]

The objective 2a minimizes the sum of the completion time of all routes in \( \bar{S} \). Constraints 2b ensure time feasibility regarding the sequence of tasks in the routes of all active vehicles. Time synchronization of the passive vehicles’ flow is ensured by constraints 2c and 2d. The variable domains are given by 2e in which \([a_\tau, b_\tau] \) is the time window of Task \( \tau \) that can be computed based on the time windows of the involved requests.

On the one hand, if the linear system 2 is infeasible then there exists no combination \( T^* \) of the variables such that \((\bar{S}, T^*)\) is a feasible solution to the APVRP. Hence, we need to generate a Benders feasibility cut to exclude solution \( \bar{S} \) from the feasible region of Formulation 1. The general form of this cut is given by:

\[
\sum_{X_{ij}^p \in \bar{X}} X_{ij}^p + \sum_{Y_{\tau_\tau'} \in \bar{Y}} Y_{\tau_\tau'} \leq |\bar{X}| + |\bar{Y}| - 1
\]

However, this cut can be strengthened since the infeasibility relies only on some of the \( X_{ij}^p \) and \( Y_{\tau_\tau'} \) variables that occur in constraint 3. As explained in [3], a stronger cut can be derived by computing a minimal (or irreducible) infeasible subset (MIS) \((X^* \cup Y^*) \subset (\bar{X} \cup \bar{Y})\) and defining the cut on that subset.

On the other hand, if the linear system 2 has a feasible solution \( \bar{T} \) with solution value \( \bar{z}_{\text{sub}} \), then \((\bar{X}, \bar{Y}, \bar{U}, \bar{T})\) is a feasible solution to the APVRP with solution value \( \bar{z}_{\text{master}} + \beta \bar{z}_{\text{sub}} \). To incorporate the additional cost \( \beta \bar{z}_{\text{sub}} \) in Formulation 1, we need (i) an additional variable \( \Theta \) that needs to be included with \( + \beta \Theta \) in the objective 1a and (ii) a Benders optimality cut forcing \( \Theta \) to take the value \( \bar{z}_{\text{sub}} \) for the solution \( \bar{S} \):

\[
\Theta \geq \bar{z}_{\text{sub}} (1 - |\bar{Y}| + \sum_{Y_{\tau_\tau'} \in \bar{Y}} Y_{\tau_\tau'}). \quad (4)
\]
3 Computational Results

We implemented the branch-and-cut algorithm in Python 3.6 using Gurobi 7.5.1 to solve Benders master and subproblem. The combinatorial Benders cuts are embedded in the algorithm using the “MipNode-Callback” function of Gurobi to solve Benders subproblem at each integer node of the branch-and-bound tree. To speed-up the solution procedure, we also use several heuristic procedures to generate optimality and feasibility cuts and solve Formulation 2 only if all heuristic procedures fail. Moreover, we use a disaggregated variant of the optimality cuts and lifted variants of the feasibility cuts.

We tested our algorithm on the benchmark sets from [1] and [4]. All results were obtained using a standard PC with an Intel(R) Core(TM) i7-5500k processor clocked at 2.4 GHz, 8 GB RAM, and 64-bit Windows 10 Enterprise. Table 1 shows our preliminary results compared to the branch-and-price-and-cut approach of [4]. The table shows the average solution time in seconds for each instance set and the number of solved instances. The results indicate that an approach incorporating combinatorial Benders cuts is well suited for the APVRP. Moreover, the algorithm solves a good number of previously unsolved instances to optimality.

<table>
<thead>
<tr>
<th>instance class</th>
<th>Our approach</th>
<th>[4]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>#Sol</td>
</tr>
<tr>
<td>10 25</td>
<td>8.5</td>
<td>20</td>
</tr>
<tr>
<td>10 50</td>
<td>8.5</td>
<td>20</td>
</tr>
<tr>
<td>10 100</td>
<td>9.5</td>
<td>20</td>
</tr>
<tr>
<td>10 200</td>
<td>36</td>
<td>20</td>
</tr>
<tr>
<td>10 wide</td>
<td>2744</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>instance class</th>
<th>Our approach</th>
<th>[4]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>#Sol</td>
</tr>
<tr>
<td>20 25</td>
<td>1457</td>
<td>19</td>
</tr>
<tr>
<td>20 50</td>
<td>2351</td>
<td>18</td>
</tr>
<tr>
<td>20 100</td>
<td>3446</td>
<td>16</td>
</tr>
<tr>
<td>20 200</td>
<td>6728</td>
<td>5</td>
</tr>
</tbody>
</table>

References


A $p$-step Formulation for the CVRP

Pedro Munari
Production Engineering Department,
Federal University of São Carlos, Brazil

Twan Dollevoet
Econometric Institute,
Erasmus University Rotterdam, the Netherlands

Remy Spliet
Econometric Institute,
Erasmus University Rotterdam, the Netherlands

1 Introduction

We present a new formulation for the capacitated vehicle routing problem (CVRP) based on partial paths of length exactly $p$, or at most $p$ in case the partial path starts at the depot, see [4] for an early version. We refer to such partial paths as $p$-steps, and include a variable in our formulation for each $p$-step, indicating whether such a $p$-step is used.

The $p$-step formulation can be considered a family of formulations, one for each value of $p \in \{1, \ldots, n+1\}$. For $p = 1$ the $p$-step formulation is an arc-flow formulation while for $p = n+1$ the $p$-step formulation is a set-partitioning formulation. The $p$-step formulation could therefore be considered a generalized formulation for the CVRP, with arc-flow and set-partitioning at its extremes. Moreover, it provides new formulations based on partial paths in between these extremes.

Set-partitioning formulations for the CVRP are known for providing stronger LP bounds than arc-flow formulations. We analyze how the LP bounds progress when $p$ increases from 1 to $n+1$. It turns out that the LP bound does not necessarily increase monotonically in $p$, although a gradual increase does occur. We will make this statement more precise later. Moreover, it turns out that strengthening our formulation yields the strongest compact formulation currently known.

We suggest using a column generation algorithm to compute the LP bounds, as the number of variables grows large for increasing $p$. Incorporating this formulation in a branching algorithm yields a standard CVRP algorithm for $p = 1$ and for $p = n+1$. Our goal is to investigate whether the computation time can be reduced by another choice of $p$.

Next, we present a $p$-step formulation followed by a summary of our main results.
2 A p-step formulation

We first present the CVRP to introduce relevant notation. Consider a complete graph $G = (N, A)$ where $N = \{0, \ldots, n\}$ is a set of locations such that 0 represents the depot and $N' = \{1, \ldots, n\}$ are the customers. Each customer $i \in N'$ has a demand $0 \leq q_i$, for ease of notation let $q_0 = 0$. An unlimited amount of vehicles of capacity $Q$ is available for satisfying demand. Vehicles traverse a simple cycle starting and ending at the depot, referred to as a route, to satisfy demand of all customers along the path. The total demand of the customers on a single route cannot exceed the capacity of a vehicle and every customer is visited exactly once. We assume $q_i \leq Q$ for all $i \in N'$. The cost of traversing an arc $(i, j) \in A$ is $c_{ij} \geq 0$. The CVRP is the problem of finding routes to satisfy all customer demands while the total costs are minimized. Next, we present a new mixed integer programming formulation for the CVRP.

Let a $p$-step $r$ be a pair $(P_r, d_r)$. Here $P_r$ is a path in $G$ that 1) either traverses exactly $p$ arcs, visiting $p + 1$ nodes, or 2) starts at 0 and traverses at most $p - 1$ arcs. Also, the total demand corresponding to the customer nodes on $P_r$ does not exceed the vehicle capacity. Furthermore, $d_r$ represents the cumulative demand of the customers on a route prior to arriving at the first location on $P_r$, such that $0 \leq d_r \leq Q - \sum_{i \in P_r} q_i$. We will refer to $d_r$ as the starting load of $r$. Let $R^p$ be the collection of all $p$-steps.

Furthermore, we introduce additional parameters. Let $a^i_r$ be the degree of customer $i$ for $p$-step $r$. That is, $a^i_r$ is 2 if customer $i \in N'$ is visited once by $p$-step $r$, unless $i$ is the first or last node on $P_r$ in which case $a^i_r$ is 1, and 0 otherwise. To represent the coupling of $p$-steps, we introduce $e^i_r$ and $q^i_r$. Let $e^i_r$ be 1 if $i$ is the first location on $P_r$, -1 if $i$ is the last location on $P_r$, and 0 otherwise. Furthermore, let $q^i_r$ be $d_r + q_i$ if $i$ is the first location on $P_r$, $-d_r - \sum_{i \in P_r} q_i$ if $i$ is the last location on $P_r$, and 0 otherwise. Finally, let the binary $p$-step variable $x_r$ indicate whether $p$-step $r$ is used. The CVRP can be formulated using the following mixed integer linear program.

\[
\min \sum_{r \in R^p} c_r x_r \quad (1)
\]

\[
\sum_{r \in R^p} a^i_r x_r = 2 \quad \forall i \in N' \quad (2)
\]

\[
\sum_{r \in R^p} e^i_r x_r = 0 \quad \forall i \in N' \quad (3)
\]

\[
\sum_{r \in R^p} q^i_r x_r \geq 0 \quad \forall i \in N' \quad (4)
\]

\[
x_r \in \{0, 1\} \quad \forall r \in R^p \quad (5)
\]

The objective function (1) represents the total costs. Constraints (2) ensure that every location is visited exactly once. Constraints (3) and (4) ensure that if a partial route is
selected that ends at node \(i\) then also a partial route is selected that starts at node \(i\) with sufficient starting load. Constraints (5) specify the domains of the decision variables. We will refer to formulation (1)-(5) as the \(p\)-step formulation.

3 Results and Discussion

Denote by \(z_p\) the LP bound of the \(p\)-step formulation for a specific choice of \(p\). The following proposition describes the gradual increase in \(z_p\).

**Proposition 3.1** For \(p \in \{1, \ldots, n\}\) and \(m \geq 1\), it follows that \(z_p \leq z_{pm}\).

The proof of Proposition 3.1 is omitted for brevity, but the main argument is that any \(pm\)-step can be split into \(p\)-steps. Hence any solution with \(pm\)-steps can be transformed in a solution using only \(p\)-steps without altering the objective value.

However, if \(k > p\) is not a multiple of \(p\) it might occur that \(z_p > z_k\). Furthermore, \(z_1 \leq z_p \leq z_{n+1}\) for any \(1 < p < n + 1\).

Note that as the starting load \(d_r\) is continuous, the number of \(p\)-steps in \(R^p\) is infinite. However, we can redefine \(R^p\) to only include \(p\)-steps with minimal starting load \(d_r = 0\) or maximal starting load \(d_r = Q - \sum_{i \in P_r} q_i\). Taking convex combinations of these extreme \(p\)-steps allows us to represent any other \(p\)-step, without explicitly including them in the formulation. This yields a formulation with the number of variables in the order of \(n^{p+1}\).

Furthermore, observe that any partial path that does not end at the depot needs to be followed by another \(p\)-step. That is, at least \(p - 1\) more customers need to be visited. Using this observation we can limit the capacity on such a \(p\)-step and exclude \(p\)-steps with a load that is too high. This both improves the LP bound and can be exploited for computational gains.

Allowing cycles is a common technique in column generation for routing problems, to speed up branch-price-and-cut algorithms. An example of this is ng-path relaxation as introduced by [1]. When allowing cycles, the above results are still valid.

We have implemented a column generation algorithm to compute LP bounds. The pricing problem is decomposed per starting and ending location of a path, and per choice of starting load, either minimal or maximal. This yields a number of pricing problems in the order of \(n^2\) which we can model as elementary shortest path problems with a capacity constraint. We solve each pricing problem with a standard labeling algorithm, with computation time which is in the worst case exponential in \(p\). Preliminary experiments indicate that solving all pricing problems in parallel might yield computation times which are increasing in \(p\).

In Table 1, we present the LP bounds for some of the E instances by [2]. Here we apply ng-path relaxation. The LP bounds illustrate the gradual increase when \(p\) grows. Notice the decrease in LP-bound for example for E-n31-k7 from \(p = 4\) to \(p = 5\).
Table 1: LP bounds, with ng-path relaxation

<table>
<thead>
<tr>
<th>Instance \ p</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>( n + 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-n13-k4</td>
<td>239.278</td>
<td>239.711</td>
<td>243.156</td>
<td>247.000</td>
<td>247.000</td>
<td>247.000</td>
<td>247.000</td>
</tr>
<tr>
<td>E-n22-k4</td>
<td>349.972</td>
<td>350.522</td>
<td>354.492</td>
<td>359.728</td>
<td>364.828</td>
<td>369.758</td>
<td>373.875</td>
</tr>
<tr>
<td>E-n23-k3</td>
<td>529.884</td>
<td>531.763</td>
<td>535.381</td>
<td>538.818</td>
<td>542.955</td>
<td>539.614</td>
<td>560.000</td>
</tr>
<tr>
<td>E-n30-k3</td>
<td>448.740</td>
<td>449.799</td>
<td>451.303</td>
<td>454.776</td>
<td>457.725</td>
<td>458.560</td>
<td>474.632</td>
</tr>
<tr>
<td>E-n31-k7</td>
<td>363.304</td>
<td>363.565</td>
<td>368.349</td>
<td>370.019</td>
<td>367.954</td>
<td>372.795</td>
<td>377.800</td>
</tr>
<tr>
<td>E-n33-k4</td>
<td>784.443</td>
<td>785.808</td>
<td>793.502</td>
<td>798.183</td>
<td>801.952</td>
<td>805.372</td>
<td>818.407</td>
</tr>
<tr>
<td>E-n51-k5</td>
<td>499.433</td>
<td>499.641</td>
<td>500.508</td>
<td>504.063</td>
<td>503.915</td>
<td>506.113</td>
<td>517.098</td>
</tr>
</tbody>
</table>

To the best of our knowledge, the compact formulation with the strongest bound is currently \( F_3 \) as presented in [3]. There is no dominance between the \( p \)-step formulation and \( F_3 \). When we strengthen the \( p \)-step formulation by taking the union of the \( p \)-step formulation and \( F_3 \), the resulting formulation is the new strongest compact formulation, for every choice of \( p \).

We intend to incorporate the column generation algorithm in a branch-price-and-cut algorithm. We want to determine for which value of \( p \) a branching algorithm has the lowest computation time. In particular we want to see if this is necessarily for \( p = 1 \) or \( p = n + 1 \), or whether the lowest computation time can occur for other values of \( p \). Finally, note that the concept of \( p \)-steps can straightforwardly be applied to any type of routing problem.

References


Variable fixing based on two-arc sequences in branch-price-and-cut algorithms

Guy Desaulniers
Department of Mathematics and Industrial Engineering, and GERAD
Polytechnique Montréal, Montréal, Canada
Email: Guy.Desaulniers@gerad.ca

Timo Gschwind, Stefan Irnich
Chair of Logistics Management, Gutenberg School of Management and Economics
Johannes Gutenberg University Mainz, Mainz, Germany

1 Introduction

Branch-price-and-cut (BPC) is the leading exact methodology for solving many vehicle routing problems (VRPs) including the capacitated VRP (CVRP) and the VRP with time windows (VRPTW) [1, 2, 3]. A BPC algorithm is a branch-and-bound algorithm in which the lower bounds are computed by column generation and cuts are added dynamically to strengthen the linear relaxations. Column generation is iterative and solves, at each iteration, a restricted master problem (RMP) and one or several pricing problems. For most VRPs, the pricing problem is an elementary shortest path problem with resource constraints (ESPPRC) which can be solved by a labeling algorithm (see [4]).

State-of-the-art BPC algorithms are complex and include various acceleration techniques such as bidirectional labeling, \(ng\)-route relaxation, limited-memory subset row cuts, variable fixing by reduced costs, and route enumeration. In this paper, we focus on variable fixing. The procedure proposed by Irnich et al. [5] allows to fix to 0 the variables associated with all routes containing at least one arc from a subset of arcs determined according to the dual solution of a linear relaxation. This is equivalent to removing the arcs in this subset from the network used to generate routes. As suggested in [2], this fixing criterion can be refined to consider also the state of a resource (e.g., the load) before traversing an arc. With these techniques, both papers report fixing to 0 the flow on more than 90% of the arcs for certain instances, yielding a significant speed up of the computational time.

In this paper, we extend the procedure of Irnich et al. [5] by considering sequences of two arcs, i.e., under certain conditions, we can fix to 0 the variables associated with routes traversing a sequence of two arcs. This strategy is less straightforward than variable fixing based on a single arc because it requires modifying the labeling algorithm used for pricing.

2 Route variables and route generation

When using a BPC algorithm for solving a VRP, the problem is modeled using an extended formulation that often contains binary route variables. Let \(P\) be the set of feasible routes and denote by \(x_p\) the route variable associated with route \(p \in P\). This variable is equal to 1 if route \(p\) is selected in the solution and 0 otherwise. Let \(N\) be the set of customers
(or requests or locations) that are subject to elementarity requirements, i.e., that cannot be visited more than once in a route.

In practice, there is a huge number of variables \(x_p\), which are generated dynamically by solving an ESPPRC pricing problem at each column generation iteration. The ESPPRC is defined on a network \(G = (V, A)\), where \(V\) is the node set including a source and a sink node, and \(A\) is the arc set. \(G\) encodes all feasible routes as paths from the source to the sink node and the costs of the arcs in \(A\) are such that the sum of the costs of the arcs forming a feasible route (path) is equal to the reduced cost of the corresponding route variable. To solve the ESPPRC, a labeling algorithm is often used. In such an algorithm, a partial path from a source node to a given node \(j\) is represented by a label \(E = (T^{\text{cost}}, (T^r)_{r \in R}, U)\), where \(T^{\text{cost}}\) is the (reduced) cost of this path, \(R\) is the set of constrained resources (such as load and time), \(T^r\) is the value of resource \(r \in R\) at node \(j\), and \(U\) is the subset of the unreachable customers, i.e., those that cannot be visited anymore because it was already visited or because of the resource constraints. Starting from an initial label at the network source node, labels are extended forwardly in the network using resource extension functions. To avoid enumerating all feasible paths, a dominance rule is applied to discard labels that cannot yield an optimal path.

### 3 Variable fixing

Variable fixing by reduced cost is based on the following well-known result.

**Proposition 1.** Let \(UB\) be an upper bound on the optimal value of an integer linear program, \(\pi\) a dual solution to its linear relaxation that provides a lower bound \(LB(\pi)\), and \(\bar{c}_p(\pi)\) the reduced cost with respect to \(\pi\) of an integer variable \(x_p, p \in P\). If \(\bar{c}_p > UB - LB(\pi)\), then \(x_p = 0\) in any optimal solution.

Setting \(x_p = 0\) is difficult in a BPC algorithm because one must also forbid the generation of route \(p\) by the pricing problem. Irnich et al. [5] proposed to rather fix to 0 subsets of variables as follows. Let \(a \in A\) be an arc and \(P_a \subseteq P\) the subset of feasible paths (routes) containing \(a\). If \(\bar{c}_p(\pi) > UB - LB(\pi)\) for all \(p \in P_a\), then one can set \(x_p = 0, \forall p \in P_a\), and remove \(a\) from \(A\) to avoid generating any path in \(P_a\). For many VRPs, \(\bar{c}_p(\pi) = \bar{c}_a(\pi) + \bar{c}^{o \rightarrow i(a)}(\pi) + \bar{c}^{j(a) \rightarrow d}(\pi)\), where \(\bar{c}_a(\pi)\) is the cost of arc \(a\) in the pricing problem and \(\bar{c}^{o \rightarrow i(a)}(\pi)\) (resp. \(\bar{c}^{j(a) \rightarrow d}(\pi)\)) is the sum of the costs of the arcs in \(p\) from the source node \(o\) to the tail node \(i(a)\) of \(a\) (resp. the head node \(j(a)\) of \(a\) to the sink node \(d\)). In this case, computing a resource-constrained shortest path from \(o\) to \(i(a)\) of cost \(\bar{l}^{o \rightarrow i(a)}(\pi)\) and another one from \(j(a)\) to \(d\) of cost \(\bar{j}^{(a) \rightarrow d}(\pi)\) can be used to compute a lower bound \(\bar{l}_a(\pi) = \bar{c}_a(\pi) + \bar{l}^{o \rightarrow i(a)}(\pi) + \bar{j}^{(a) \rightarrow d}(\pi)\) on the reduced cost \(\bar{c}_p(\pi)\) of all \(p \in P_a\). Note that \(\bar{l}^{o \rightarrow i(a)}(\pi)\) and \(\bar{j}^{(a) \rightarrow d}(\pi)\) are lower bounds and can, thus, be computed using relaxed paths such as \(mg\)-paths that were used for our tests (neighborhoods of size 10). Note also that the lower bounds \(\bar{l}^{o \rightarrow i(a)}(\pi)\) (resp. \(\bar{j}^{(a) \rightarrow d}(\pi)\)) for all \(a \in A\) can be computed at once by solving a possibly relaxed ESPPRC from \(o\) (resp. \(d\)) to all other
nodes using a forward (resp. backward) labeling algorithm. Furthermore, these bounds can be made specific for each arc \(a \in A\) (and not for each node \(i \in V\)) if a pair of forward and backward shortest paths yielding a feasible path when concatenated with \(a\) is selected. Once a linear relaxation is solved, the lower bounds \(\bar{l}_a(\pi), a \in A\), are computed and, for each arc \(a \in A\) such that \(\bar{l}_a(\pi) > UB - LB(\pi)\), \(a\) is removed from \(A\) and \(x_p\) is set to 0 for all routes \(p \in P_a\).

Let \(S\) be the set of sequences of two arcs \((a_1, a_2) \in A^2\) that can be traversed consecutively by the same vehicle in a feasible route. Using the same lower bounds \(\bar{l}^{o\rightarrow i}(a)(\pi)\) and \(\bar{l}^{i\rightarrow d}(a)(\pi), a \in A\), as above, we can compute for each sequence \((a_1, a_2) \in S\), a lower bound \(\bar{l}_{(a_1, a_2)}(\pi)\) on the reduced cost \(\bar{c}_p(\pi)\) of all feasible paths traversing \(a_1\) and \(a_2\) consecutively:

\[
\bar{l}_{(a_1, a_2)}(\pi) = \bar{c}_{a_1}(\pi) + \bar{c}_{a_2}(\pi) + \bar{c}^{p\rightarrow i}(a_1)(\pi) + \bar{c}^{p\rightarrow d}(a_2)(\pi).
\]

Let \(S^* \subseteq S\) be the subset of two-arc sequences \((a_1, a_2) \in S\) such that \(\bar{l}_{(a_1, a_2)}(\pi) > UB - LB(\pi)\). According to Proposition 1, we can set \(x_p = 0\) for all routes \(p \in P\) traversing a two-arc sequence \((a_1, a_2) \in S^*\).

Unlike the single-arc case, it is not possible to remove two-arc sequences directly in network \(G\). The labeling algorithm must be modified. The label definition is extended to include two additional components denoted \(L\) and \(F\), i.e., a label is given by \(E = (T^{\text{cost}}, (T^r)_{r \in R}, U, L, F)\), where \(L\) is the last arc in the corresponding partial path and \(F = \{a \in A \mid (L, a) \in S^*\}\) is a subset of the arcs along which this label cannot be extended because it would violate a constraint on a two-arc sequence in \(S^*\). In the extension phase of the labeling algorithm, a label cannot be extended along an arc \(a\) if \(a \in F\). Given these additional restrictions, the dominance rule stated above needs to be adapted: For two labels \(E_k = (T_k^{\text{cost}}, (T_k^r)_{r \in R}, U_k, L_k, F_k), k = 1, 2,\) label \(E_1\) dominates label \(E_2\) if \(T_1^q \leq T_2^q\) for all \(q \in \{\text{cost}\} \cup R\) and \(U_1 \subseteq U_2\), and \(F_1 \subseteq F_2\) and at least one of these relationships is strict.

It is possible to be finer in the dominance by considering that a label can dominate another one only for a subset of its possible extensions. Let \(\delta^+(a_1) = \{a_2 \in A \mid (a_1, a_2) \in S, j(a_1) = i(a_2)\}\) be the subset of arcs that can immediately follow arc \(a_1 \in A\) in a route. If \(T_1^q \leq T_2^q\) for all \(q \in \{\text{cost}\} \cup R\) and \(U_1 \subseteq U_2\), then \(F_2\) can be replaced by \(F_2 \cup (\delta^+(L_1) \setminus F_1)\) in label \(E_2\). This corresponds to additionally forbidding the extension of \(E_2\) along any arc along which \(E_1\) can be extended. Observe that, if arc-sequence restrictions were not considered, \(E_1\) would dominate \(E_2\) and \(E_1\) would possibly be extended (if not dominated) along all arcs in \(\delta^+(L_1)\). Otherwise, \(E_1\) (resp. \(E_2\)) would possibly be extended along all arcs in \(\delta^+(L_1) \setminus F_1\) (resp. \(F_1 \setminus F_2\)). Overall, the number of possible extensions is, thus, reduced when considering sequence restrictions.
4 Preliminary computational results and discussion

A very preliminary version of the proposed BPC algorithm has been implemented. Our tests were performed on the Solomon’s VRPTW benchmark instances with 25, 50 and 100 customers for which upper bounds were computed by the hybrid genetic algorithm in [6]. On the 83 instances that were not solved at the root node, we report the following results obtained after solving the root node. On average 68.2% of the arcs can be eliminated based on the single-arc criterion, resulting in the removal of 85.2% of the two-arc sequences. Applying the two-arc sequence criterion allows the elimination of 23.7% of the remaining sequences on average. For the moment, this additional variable fixing yields shorter computational times only for certain instances. However, we believe that the algorithm can be improved and that removing these arc sequences will have a substantial impact on the total computational time for most instances. Indeed, the number of forbidden extensions resulting from these two-arc sequences should speed up the labeling algorithm significantly. As shown in [5], variable fixing can also improve the lower bounds throughout the search tree and, thus, reduce the total number of search nodes. Finally, forbidding arc sequences will also help the route enumeration procedure. Indeed, it will reduce the time required to enumerate the routes, possibly the total number of enumerated routes and the time required to solve the resulting mixed integer program.

At the conference, we will present complete results for the VRPTW and the electric VRPTW.

References


1 Introduction

Order picking, i.e., the process of retrieving articles from their storage locations according to customer orders, is one of the most cost-intensive operations in warehousing [1]. An effective order-picking process can, therefore, significantly improve the overall performance of a warehouse. Within the order-picking process, the traveling of the pickers is a crucial activity as it is the main factor responsible for the overall picking time [2].

Given a set of customer orders each comprising one or more items, the order batching problem (OBP) consists of designing a set of picking batches such that each order is assigned to exactly one batch, all batches satisfy the capacity restriction of the pickers, and the total traveled distance by the pickers is minimal. The storage locations of the individual items as well as the routing strategy used by the pickers to traverse the warehouse in the picking process are assumed to be fixed a priori. The standard variant of the OBP considers a rectangular warehouse with parallel aisles of equal length and width. Cross aisles at the front and at the back of the warehouse connect the vertical aisles. All pickers start and end their picking routes at a common depot located in front of the leftmost vertical aisle.

In order to collect the items of a batch, pickers traverse the warehouse using a predefined routing strategy. For rectangular warehouses, the strategies s-shape (or traversal), midpoint, return, largest gap, composite, and optimal have been proposed in the literature. Since the simpler heuristic routing strategies are typically more intuitive for the pickers and exhibit less risk of in-aisle congestion than an optimal routing, they are often encountered in practice.

From a computational point of view, the OBP is a challenging problem. It has been shown to be \( \mathcal{NP} \)-hard in general [3]. An overview of heuristic approaches is given in the recent work [4]. Literature on exact approaches to the OBP is scarce and restricted to the branch-and-bound algorithm in [5], a branch-and-price approach for a special case of
the OBP, in which all orders comprise the same number of individual items, in [3] (both papers computing optimal picking routes), and MILP formulations for s-shape, midpoint, and return strategies solved by a commercial solver in [6]. Recently, an exact approach for the same routing strategies based on a set-partitioning formulation, cut-and-column generation, and batch enumeration was suggested in [7].

In this paper, we propose a full-fledged branch-price-and-cut (BPC) algorithm for the OBP. We consider s-shape, return, midpoint, and largest gap strategies. The CG pricing problems are modeled as shortest path problems with resource constraints (SPPRC) and solved by a dynamic-programming labeling algorithm. Several enhancements typically utilized in CG approaches to vehicle routing problems are employed to speed up the CG process. Preliminary computational results indicate the applicability of our approach.

2 Branch-Price-and-Cut Algorithm

To formulate the OBP as a set-partitioning problem, let \( O = \{1, \ldots, n\} \) be the set of customer orders. Each order \( o \in O \) comprises a set \( I_o \) of individual items. A sufficiently large number of pickers with an identical capacity \( Q \) is available to perform the picking routes. The capacity consumption of an order \( o \in O \) is \( q_o \). Note that the capacity consumption of the individual items is not relevant, since splitting of orders is not allowed. We denote by \( \Omega \) the set of all feasible batches. Binary parameters \( a_{ob} \) indicate if order \( o \in O \) is contained in batch \( b \in \Omega \). The distance needed to pick all individual items \( i \in O_b := \bigcup_{o \in O : a_{ob} = 1} I_o \) of a batch \( b \in \Omega \) is given by the function \( c(b) = c_b \) that determines the travel distance depending on the chosen routing strategy. Note that the distance function \( c_b \) is not separable in the orders \( o \in O_b \). Finally, let \( \lambda_b \) by binary decision variables equal to 1 if batch \( b \in \Omega \) is chosen and 0 otherwise. Then the OBP can be formulated as follows:

\[
\begin{align*}
\min & \quad \sum_{b \in \Omega} c_b \lambda_b \tag{1a} \\
\text{s.t.} & \quad \sum_{b \in \Omega} a_{ob} \lambda_b = 1 \quad \forall o \in O \tag{1b} \\
& \quad \lambda_b \in \{0, 1\} \quad \forall b \in \Omega \tag{1c}
\end{align*}
\]

The objective (1a) minimizes the total traveled distance, constraints (1b) ensure that all orders are picked exactly once.

Since the number of batches in \( \Omega \) is typically too large, we solve formulation (1) with a BPC algorithm, which is a branch-and-bound algorithm that computes the bounds by CG strengthened with additional valid inequalities. Initializing model (1) with a subset of batches, the CG pricing problem generates negative-reduced cost batches dynamically. The reduced cost of a batch is given by \( \bar{c} = c_b - \sum_{o \in O_b} \pi_o \), where \( \pi_o \) is the dual price associated with constraints (1b).
We model the pricing problem as an SPPRC as follows: Let $G = (V, A)$ be a digraph with $n + 1$ vertices $V = \{0, \ldots, n\}$ and $2n$ arcs $A$. For each order $o \in O$, there are two parallel arcs $e_{o1}$ and $e_{o0}$ connecting vertices $o - 1$ and $o$ and indicating the inclusion or not, respectively, of order $o$. In general, every $0$-$n$-path in $G$ defines a batch and every $0$-$i$-path with $i \leq n$ defines a partial batch. The solution of the pricing problem is, thus, equal to finding a capacity feasible minimum-cost path in $G$. We solve the SPPRC with a dynamic-programming labeling algorithm. Denote by $q(L)$ and $\bar{c}(L)$ the capacity consumption (or load) and the reduced cost of a label $L$ representing a partial batch. The key problem that has to be dealt with in the labeling algorithm is the non-separability of the distance function, which makes invalid the standard dominance relation between two labels $L_1$ and $L_2$, i.e., a direct comparison of their load and reduced-cost components. To overcome this issue and to obtain a valid dominance rule, we derive a penalty function $p(L_1, L_2)$ that provides an upper bound on the additional distance $L_1$ has to cover compared to $L_2$ for any possible extension of $L_2$ to the sink vertex $n$. Clearly and analog to the travel distance function, $p(L_1, L_2)$ depends on the chosen routing strategy. Then, $\bar{c}(L_1) + p(L_1, L_2) \leq \bar{c}(L_2)$ constitutes a valid dominance relation for the cost component.

To speed-up the CG process, we use the following acceleration strategies: (i) bounding, (ii) bidirectional labeling with a dynamic half-way point, (iii) different pricing heuristics using either a relaxed dominance rule or premature stopping of the labeling process. Furthermore, we add subset row inequalities with subsets of cardinality three to strengthen formulation (1). They are handled in the pricing problem using the standard mechanism.

To obtain integer solutions, we use a hierarchical branching scheme that first branches on the number of pickers, if fractional. Ryan and Foster branching, i.e., branching on whether or not two orders are in the same batch, finally ensures integrality. The latter branching decisions are incorporated into the pricing problem as follows. We group together the orders that are affected by some mutual branching decisions. All orders of each such group are then aggregated and represented by a single vertex in the pricing problem digraph $G$. For each (branching-decision) feasible combination of these orders, one arc entering this vertex is created.

3 Preliminary Computational Results and Upcoming Work

Preliminary computational experiments were conducted using the benchmark set introduced in [7]. It comprises instances from 20 to 100 orders (in steps of ten). The pickers’ capacity is $Q = 24, 36, 28$. Table 1 summarizes our results (own) in comparison to the integer CG approach of [7] (M&O). It reports the number of instances solved to proven optimality within the time limit (one hour for own, two hours for M&O).

Upcoming research includes the use of dual-optimal inequalities [8] to stabilize the CG
process. Moreover, we observed that our BPC algorithm suffered from severe symmetry issues in the OBP instances we tackled. This was especially true for the s-shape routing strategy where the creation of several tens of thousand branch-and-bound nodes in one hour of computation time was not uncommon. We, therefore, believe that the use of (i) an aggressive strong branching strategy, (ii) other branching rules, or (iii) different selection strategies for the order pairs in the Ryan and Foster branching could significantly improve the overall performance of our BPC algorithm.

### References


1 Problem Description

We introduce and solve a real-world, constrained 2-d packing problem which arises in the loading of roll-on-roll-off vehicle ferries. The objective is to minimize penalties resulting from failing to load vehicles that have booked tickets for the crossing. Vehicle ferries transport private vehicles and commercial freight, and the variety of vehicle sizes motivates the consideration of complex 2-d packing arrangements. On departure day the loading process begins shortly before the scheduled departure time of the ferry. Vehicles have been booked in advance and arrive at random times prior to departure where they are parked in lanes on the dockside (or yard) prior to loading. Any vehicles arriving late are unable to board. The allocation of a vehicle to one of the N lanes is made on arrival at the ferry terminal and is based on the vehicle type and the yard policy being used. The use of lanes constrains the packing problem by only allowing vehicles that are at the front of a lane to be loaded next. Furthermore, the orders of the vehicles within the lanes are also uncertain due to the random arrival times, this two can make an optimal packing solution unattainable. As an example, the optimal packing solution may require all of the large vehicles to be loaded first but, if it is impossible to reserve a lane for these vehicles, they
may all be parked at the back of the lanes. In general if there are not enough yard lanes to keep all vehicle types in separate lanes then the optimal ferry utilisation will be reduced and will also be a function of uncertain arrival times. The result is vehicles being left off the ferry. Vehicles that cannot be loaded onto the ferry are refunded and compensated (referred to collectively as a penalty), which subtracts from the ticket revenue generated in the selling season.

2 Problem Formulation

The operational revenue (ticket revenue minus penalties) can be optimised by maximising the efficiency of the vehicle ferry loading process, which is achieved by improving the packing solutions and the yard policy. A way of minimising penalties is by only selling tickets to vehicles that can definitely be packed, irrespective of their arrival order. This approach minimises penalties but can be too conservative with respect to revenues. Maximising ticket revenues and minimising penalties are conflicting objectives. We formulate this problem as a two-stage stochastic optimisation problem. In the first stage arrival orders are uncertain and the task is to determine a yard policy for allocating vehicles to lanes when they arrive. The objective is to minimise penalties due to failing to load vehicles. The second stage considers a realisation of the random arrival process, in which vehicles have been allocated to terminal lanes using the yard policy from the first stage. The second stage recourse problem is that of resolving the packing problem for the realised set of vehicle queues such that penalties are minimised. It is convenient for later discussion to note that minimising penalties is equivalent to maximising operational revenue.

To address arrival order uncertainty in the first stage we consider a set of random arrival scenarios (S) over which to maximise operational revenue. We propose a maximin formulation which maximises the operational revenue that can be guaranteed in all of a specified number of those scenarios (subset size). This is calculated based on a set that we define as the intersection vehicle mix of the vehicles that are loaded in each of the individual arrival scenarios. A basic definition of an intersection vehicle mix is the minimum numbers of each vehicle type from each vehicle mix—hence the ‘min’ in ‘maximin’. It is the intersection vehicle mix that can definitely be packed in every scenario. The subset size can be thought of as a risk parameter, which controls the risk of incurring high penalties, and is set by a decision maker. The intersection vehicle mix of the maximin objective can also be used to determine whether the ferry’s capacity has been oversold in the selling season. In the event of overselling it is the vehicles that are not in intersection vehicle mix that should be pre-emptively refunded or assigned to a different departure time assuming that this pre-emptive action negates the risk of a financial penalty. We demonstrate in experiments that higher operational revenues can be achieved using the
proposed maximin formulation compared to those achieved when maximising the expected operational revenue. This result holds regardless of whether or not pre-emptive refunds or departure time reassignment actions are implemented. The proposed formulation offers a number of levers for avoiding over-conservative (low-risk low-revenue) solutions. The subset size risk parameter can be lowered, and in experiments we demonstrate scenarios where this is beneficial.

3 Packing Methodology

On a roll-on-roll-off ferry, vehicles enter at one end of the ferry and exit at the other end. Due to the limited space for vehicle manoeuvres vehicles can only be parked in positions that they can drive into from the entrance. Therefore the packing problem that we address in this work is: 1) constrained by the time required to complete the loading process, meaning that any proposed packing arrangements have to be easy to implement by human loaders; 2) constrained by the orders of the terminal lanes, as discussed earlier; and 3) is also constrained by the manoeuvrability of vehicles in that they must be able to drive into their parking positions from the entrance. This final constraint means that the packing order has to be taken into account.

We propose a Sequential Guillotine Cut Knapsack Packing methodology (SGCKS) for 2-d rectangle packing that is based on packing rows and columns of rectangles (vehicles) at a time referred to as (horizontal and vertical) cuts where the vehicles are selected from those at the front of the terminal lanes. This approach addresses the three constraints listed above. The SGCKS packing methodology is presented first and we then introduce a slight relaxation of this methodology (General Packing or GP) that allows for lateral and longitudinal compacting of rectangles, which recovers wasted space. A lower bound based on a relaxation of the packing formulation is introduced, which shows that the SGCKS solutions are close to optimality. The SGCKS methodology has the practical benefit that the packing and yard policy solutions are intuitive and easy to implement by human loaders and yard personnel. According to the improved cutting and packing typology of Wascher et al. According to the improved cutting and packing typology of Wascher et al. [1] the packing problem considered in this paper is related to the “2-d rectangular Single Knapsack Problem (2DSKP)”, with additional constraints on the availability of the small items (vehicles).

4 Solution Approach

To solve the first stage problem we propose an iterative metaheuristic that iterates between packing solution optimisation whilst fixing the yard policy and yard policy optimisation whilst fixing the packing solution. The objective of packing iterations is to find packing
solutions for each arrival scenario that improve the operation revenue objective. Both the yard policies and packing solutions for each arrival scenario are encoded as integer strings where each integer encodes size and orientation information. The SGCKS and GP methodologies are well suited to this iterative solution approach because they have the feature that searching the space of yard policies also searches the space of packing arrangements even for a fixed packing integer solution string. Within the iterative metaheuristic a variety of search neighbourhoods are used including: mutation; orientation mutation; size mutation; integer position mutation; and randomised constructive heuristics. Experiments were carried out to identify the best probabilities for applying these neighbourhoods.

5 Results and Conclusions

In experiments, the GP methodology achieved an average optimality gap of 2.71% compared with the calculated lower bound, based on 300 generated 2-d rectangle packing problems. Average solutions times were around 10 seconds for each problem. Experiments were also carried out to assess the impact of varying the number of arrival scenarios and the subset size in the yard policy optimization formulation. These experiments were repeated for three demand scenarios: 1) vehicle types with fully nested sizes (Russian Doll); 2) vehicle types with some nested sizes; and 3) vehicle types with no nested sizes (increasing-length decreasing-width). It was found that in cases 2 and 3 the approach led to over-conservative solutions due to the reduced size of the vehicle mix intersections in these cases. However reducing the subset size alleviates this problem. For the full nested vehicle size case the highest revenues were achieved by setting the subset size equal to the uncertainty set size. The proposed minimax formulation attained higher revenues than an equivalent formulation based on maximising the expected operational revenue. The reason for this result is that the intersection vehicle mix reduces the overfitting of the packing solutions for each first stage arrival scenario as it focusses on the packing of a single vehicle mix in each scenario.

The proposed SGCKS also has a broad applicability in general packing problems; for example, similar problems that occur in the loading of air freight, where there is some uncertainty in the arrival process.

References

Railroads move large quantities of a broad variety of commodities over long distances in a cost-effective and environmentally sustainable manner. They are thus a key element of the world-wide intermodal transportation network, which displays a steady traffic growth. Efficient and profitable railroad activities require adequate planning of operations and resources. These planning processes are complex due in large part to the interactions among the main components and goals of the system, e.g., yards, lines, trains, blocks of cars, economic profitability, resource utilization, and customer satisfaction.

We focus on the Block & Car Tactical Planning problem (BCTP) for intermodal rail for which, according to our best knowledge, no adequate methodology exists. We propose a new scheduled service network design (SSND) model for the BCTP that accounts for the characteristics of intermodality, namely, the demand expressed in terms of containers of various types and their assignment to multi-platform double-stack cars of different types. This container-to-car assignment requirement adds a new dimension to the train blocking problem addressed in the literature, and a new design layer to the classical car-to-block and block-to-train design decisions of the SND formulations. We briefly describe the problem (Section 1) and the model (Section 2), and sum up the presentation plan in Section 3.

1 Problem Description

Rail freight is moved by trains made up of blocks of cars. Cars are classified (sorted) in yard terminals and assigned to blocks. A block is a group of cars, with possibly different origins and destinations, that move as a single unit between a pair of yards, without cars being handled individually when transferred from one train to another at intermediate yards. Blocking thus aims to take advantage of economies of scale and reduce yard handling costs. A block is moved by a sequence of trains, while a car can be moved by a sequence
of blocks between its origin and destination yards. The classical train blocking problem then consists in selecting the blocks to build and assigning cars to blocks. A number of studies in the literature address this case, e.g., [1, 3], but none accounts for the intermodal challenges, notably the need to explicitly account for the loading of containers on cars.

About 90% of the containers used worldwide are 20 or 40 feet, while longer units, e.g., 53 feet, are also used in the North American market. The origin-to-destination, OD, demand is then defined as a number of containers of particular type (depending on, e.g., size and transported goods), OD pair, arrival time at origin, and due time at destination. This definition is different from the classical one in numbers of loaded cars.

Differences exist in terms of equipment also. The railroad uses a heterogeneous fleet of intermodal cars, different from other rail cars, to move the demand. Each car has one to five platforms, which may be double- (two container slots) or single-stacked (one slot). The matching of containers to slots is an important issue since, even though all slots are not usable for all container types (e.g., 40-foot platforms cannot take 53-foot containers in the bottom slot), the multitude of car and container types yields a large number of ways to load containers that must respect several loading rules [2].

Given a train schedule, the goal of the BCTP is to determine a blocking plan and schedule that minimizes the total operation costs to deliver the container demand. Particular challenging issues in addressing the BCTP are 1) simultaneously considering three consolidation processes, containers to cars, cars to blocks, and blocks to trains, 2) differentiating car and container types, and 3) representing in a computationally efficient way the assignment of containers to cars within a tactical blocking model.

2 Problem Description and Modeling

We propose a model that is based on a cyclic four-layer space-time network representation, illustrated in Figure 1. This is a tactical problem and the plan is defined over a given schedule length (e.g., a week) and is assumed to be repeated over a planning horizon. Since intermodal traffic shares the network with trains moving other types of cargo, we take the train schedule as given. Different from most studies in the literature [3], this allows us to use a continuous-time network representation. Moreover, we do not have any design decisions related to the train (top) layer.

The arrival and departure times of each scheduled intermodal train at the terminals on its route are known and yield the corresponding arrival and departure nodes, defining the time structure of the entire network. We model train activities through waiting arcs representing the time trains spend at terminals and moving arcs between terminals. The train features, e.g., power and maximum length, provide the capacity of the moving arcs.

The block layer is illustrated below the train layer. A block is defined by a particular
time-dependent OD pair and a path made up of movements on train-moving arcs and activities in yards (virtual paths within the block layer are obtained by projecting the train-moving arcs on it). We model building new blocks, transferring blocks from one train to another, and dismantling blocks at their destinations through, respectively, Blocks-Build intra-layer arcs and three sets of inter-layer arcs between the block and train layers: Block-Attach (attaching a block to a train), Block2Dismantle (a block at destination is disconnected from the train and the cars are “freed”), and Block-Transfer (blocks are disconnected from a train and attached to the next one in their route).

Demand enters and exists the system through the container layer, which appears at the bottom of the figure. Demand is defined by its origin terminal, time at the origin terminal, destination terminal, due-date at destination, container type, number of containers of that type, and a penalty cost for late arrival. A set of blocks that can transport it, i.e., with same origin and destination is also defined. Upon arrival, containers wait on Container-Waiting arcs until the selected block. The inter-layer Containers-Loaded arcs support the container-to-car-to-block assignment model we propose, which is based on the relations among the lengths of blocks, platforms, and the two main container types (40 and 53 feet). The Containers-at-Destination inter-layer arcs, with possibly late-arrival penalty costs, support the flow of containers at destinations, unloaded from cars out of the dismantled blocks, to a demand-out super sink. Finally, the car layer, situated between the container and block layers, models the hooking of cars to blocks (Cars2Blocks links) and the release of cars at the block destinations (Cars at Destination links).

We propose a mixed integer linear programming (MILP) formulation with four groups of decision variables: (i) Block selection, binary variable equal to 1 if a block is selected, and 0 otherwise; (ii) Container flow distribution representing the number of containers of
each compatible demand on each block; (iii) Container and platform-on-block distribution standing for the total number of containers of each type and the associated number of platforms (cars) of each type on each block.

The objective function minimizes the total cost of the system over the planning horizon. It encompasses the cost of selecting, operating and transferring blocks, the time-related costs for containers and cars idling at train stops, as well as penalties for late arrival of demand and train-capacity overload. There are flow conservation constraints in the four layers, container-to-car assignment constraints yielding the number of platforms to move, linking constraints related to the capacities of blocks and trains, and yard capacity constraints. The latter are defined as bundle constraints regarding the maximum length of blocks that can be built and dismantled during a given time interval at a given terminal.

3 Conclusion

A large set of experiments was performed using data from a major North-American railroad. Realistically-sized and defined instances were solved exactly within rather short computing times. We analyzed the impact on the design of the block & car plan and the computational difficulty of the resulting instance, of several problem characteristics, e.g., number of trains, number of intermediate stops of these services, demand distribution in space and time, relative value of the penalties relative to the operational costs, possibility to split the demand flows among several blocks, and the inclusion of extra trains, with high costs. We will present and discuss the problem, the container-to-car assignment solution we propose and the BCTP model, the experimental setting and the numerical results and analyzes obtained.

We gratefully acknowledge the close collaboration with the Canadian National Railway Company (CN). This research is supported by the CN Chair in Optimization of Railway Operations, U. de Montréal, and the Natural Sciences and Engineering Research Council of Canada through its Collaborative Research & Development and Discovery grant programs.

References


The Hub Location and Arc Pricing Problem

Maria Battarra  
School of Management,  
University of Bath, Bath, United Kingdom

Antonio M. Rodríguez-Chía  
Departamento de Estadística e Investigación Operativa  
Universidad de Cádiz, Cádiz, Spain

Güneş Erdoğan  
School of Management,  
University of Bath, Bath, United Kingdom  
Email: G.Erdogan@bath.ac.uk

1 Introduction

In this paper we consider a hub location problem that involves two levels of decision makers acting non-cooperatively. The upper level decision maker (leader) tries to maximize the revenue by locating $q$ hubs and providing transportation service between the hubs at a lower price than the cost of direct transportation. The lower level decision makers (followers) aim to send their commodity from a source to a destination at minimum cost, and accordingly determine the routing of their commodities based on the prices of the hub arcs and direct transportation arcs. The followers do not use the hub service unless it can transport their commodity at a price that is cheaper by a factor of $\gamma$. An application arises when a cargo company decides to switch from door-to-door service to depot-to-depot service, and aims to find the optimal locations of the depots as well as the price between any two depots.

Hub location problems have been studied extensively, with many variants based on the number of hubs to transport commodities, hub capacities, and objective function type (minisum or minimax). We refer the interested reader to the book of Contreras [1] for a comprehensive review. Joint hub location and pricing models, on the other hand, have not received much attention. Lüer-Villagra and Marianov [2] have studied a competitive hub location and pricing problem, where two hub networks compete to maximize their profits, with the existing hub network using mill pricing (i.e., a common discount factor $\alpha$ is applied to the original cost of the arcs to determine the cost of traversing the hub arcs) and the entrant using arc-based pricing. The most relevant study is [3], where the
authors focus on the problem of deciding which arcs of a given network to invest on and subsequently apply pricing to. In the following sections, we provide the formal definition and a formulation for our problem, initial results, and preliminary conclusions.

2 Formal definition and formulation

Consider a complete directed graph $G = (V, A)$ with vertex set $V$ and arc set $A$, and a set of commodities $K$ to be transported. We define $t_k$ as the total amount of commodity $k \in K$ to be routed from its origin $o(k) \in V$ to its destination $d(k) \in V$. In addition, we define $c_{ij} \geq 0$ as the cost of traversing arc $(i, j) \in A$ using the infrastructure network for any commodity, and assume that these costs satisfy the triangle inequality. Each follower uses the shortest path on the network, and may use the hub service when traversing the arcs between the hubs with prices $p_{ij}$. In the rest of this extended abstract, we use the term cost to denote the fee of using the infrastructure arcs, and price to denote the fee of using the hub arcs. A follower uses the hub arcs only if it is possible to transport its commodity at a cost that is cheaper by a factor of $\gamma \in (0, 1]$ with respect to using the existing infrastructure arcs, and they become compliant, i.e. use a shortest path with maximal profit for the leader. The problem consists of finding the optimal locations of $q$ hubs, and the prices $p_{ij}$ between the hubs to maximize the revenue of the leader.

For each $i \in V$, we define $\delta^+_i$ as the set of outgoing arcs from node $i$ and $\delta^-_i$ as the set of incoming arcs. For the sake of brevity, we write $d_{ij}^k$ to denote the cost of using arc $(i, j) \in A$ to transport commodity $k \in K$ without using the hub network, where $d_{ij}^k = \gamma \times c_{ij}$ if $i = o(k)$ and $j = d(k)$, and $d_{ij}^k = c_{ij}$ otherwise. In addition, we define

$$b_i^k = \begin{cases} 
1, & \text{if } i = o(k), \\
-1, & \text{if } i = d(k), \\
0, & \text{otherwise}.
\end{cases}$$

For any $i \in V$, let $x_i$ be equal to 1 if a hub is located at node $i$, and 0 otherwise. In addition, let $p_{ij}$ be the price of using the hub network for crossing the arc $(i, j) \in A$. The decision variables stated so far correspond to the decisions of the leader. Finally, for any $(i, j) \in A$ and $k \in K$, let $y_{ij}^k$ be equal to 1 if commodity $k$ is transported on arc $(i, j)$ without using the hub network, and 0 otherwise. Similarly, for any $(i, j) \in A$ and $k \in K$, let $w_{ij}^k$ be equal to 1 if commodity $k$ is transported on arc $(i, j)$ using the hub network, and 0 otherwise. We emphasize that the transportation variables are stated as $\hat{y}_{ij}^k, \hat{w}_{ij}^k$ and $\bar{y}_{ij}^k, \bar{w}_{ij}^k$ for the leader and the followers, respectively. A formulation of our problem can then be stated as:
maximize \[ \sum_{k \in K} \sum_{(i,j) \in A} t_k p_{ij} \hat{w}_{ij}^k \] (1)

subject to \[ \sum_{i \in V} x_i \leq q \] (2)
\[ p_{ij} \geq c_{ij}(1 - x_i) \quad \forall (i,j) \in A, \] (3)
\[ p_{ij} \geq c_{ij}(1 - x_j) \quad \forall (i,j) \in A, \] (4)
\[ \hat{w}_{ij}^k \leq x_i \quad \forall (i,j) \in A, k \in K, \] (5)
\[ \hat{w}_{ij}^k \leq x_j \quad \forall (i,j) \in A, k \in K, \] (6)
\[ \sum_{j \in \delta^+_i} (\hat{y}_{ij}^k + \hat{w}_{ij}^k) - \sum_{j \in \delta^-_i} (\hat{y}_{ji}^k + \hat{w}_{ji}^k) = b_i^k \quad \forall i \in V, \forall k \in K, \] (7)
\[ x_i \in \{0, 1\} \quad \forall i \in V, \] (8)
\[ \hat{y}_{ij}^k, \hat{w}_{ij}^k \in \{0, 1\} \quad \forall (i,j) \in A, k \in K, \] (9)
\[ \sum_{(i,j) \in A} (d_{ij} \hat{y}_{ij}^k + p_{ij} \hat{w}_{ij}^k) = \min \left\{ \sum_{(i,j) \in A} (d_{ij} \hat{y}_{ij}^k + p_{ij} \hat{w}_{ij}^k) \right\} \forall k \in K. \] (10)

The objective function (1) aims to maximize the profit of the leader. Constraint (2) sets the maximum number of hubs to be located. Constraint sets (3) and (4) state that the price of an arc cannot be less than the cost of direct transportation unless both ends of the arc are hubs. Constraint sets (5) and (6) enforce both vertices at the end of an arc to be hubs if a commodity is transported on the arc using the hub network. Constraint set (7) states the flow conservation for the commodities. Constraint sets (8) and (9) require the hub and transportation decisions to be binary. Finally, constraint set (10) ensures that each commodity is transported at minimum cost.

The formulation presented above is nonlinear due to the objective function (1) as well as the bilevel programming constraints (10). To linearize this formulation, we define \[ z_{ij}^k = p_{ij} \hat{w}_{ij}^k, \forall (i,j) \in A, k \in K, \] and add the necessary linearization constraints. We also define the dual variables \[ \lambda^k_i, \] associated with constraint \[ i \in V \] for commodity \[ k \in K \] within (10), and apply the standard inner dualization method for bilevel programming. We skip the resulting linearized formulation due to the page limit.

### 3 Initial computational results

We have implemented the linearized formulation using the callable library of CPLEX 12.7.1 and C++, and conducted experiments on workstations with two Intel E5-2650 v2...
CPUs running at 2.60 GHz and 64GB RAM. We have generated random instances with $|V| \in \{10, 15, 20\}$, $q \in \{3, \ldots, \lceil |V|/3 \rceil \}$, $\gamma \in \{0.9, 0.925, 0.95, 0.975, 1\}$, and $|K| = \sigma \times |V|^2$ for the sparsity coefficient $\sigma \in \{0.2, 0.5, 0.8\}$. The linearized formulation, strengthened with valid inequalities as well as a hybrid metaheuristic to generate initial solutions, was able to solve 1468 out of 1500 instances for $|V| = 10$ and 15 within 4 hours. The results for $|V| = 20$ are summarized in Table 1.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$\sigma$</th>
<th>Average final gap</th>
<th>Average CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.2</td>
<td>0.00%</td>
<td>40.10</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.00%</td>
<td>274.46</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.00%</td>
<td>1,209.75</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>0.00%</td>
<td>41.67</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.00%</td>
<td>500.14</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.34%</td>
<td>4,816.98</td>
</tr>
<tr>
<td>5</td>
<td>0.2</td>
<td>0.00%</td>
<td>39.35</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.00%</td>
<td>1,282.99</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.66%</td>
<td>7,717.32</td>
</tr>
<tr>
<td>6</td>
<td>0.2</td>
<td>0.00%</td>
<td>38.33</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.12%</td>
<td>2,608.45</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>2.49%</td>
<td>13,788.93</td>
</tr>
<tr>
<td>7</td>
<td>0.2</td>
<td>0.00%</td>
<td>35.16</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.08%</td>
<td>3,537.40</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>1.29%</td>
<td>10,494.07</td>
</tr>
</tbody>
</table>

4 Conclusion

The strengthened linearized formulation is observed to be able to solve problem instances with nontrivial sizes. In future work, we aim to solve instances with larger sizes, and test the formulation on benchmark instances from the literature. In addition, we aim to analyze the case with mill pricing.

References


The Covering Location Problem with Interconnected Facilities

Marilène Cherkesly
Department of management and technology,
ESG UQÀM, Montreal, Canada
Email: cherkesly.marilene@uqam.ca

Mercedes Landete
Departamento de Estadística, Matemáticas e Informática,
Universidad Miguel Hernández de Elche, Elche, Spain

Gilbert Laporte
Department of Management Sciences,
HEC Montréal, Montreal, Canada

1 Introduction

The purpose of this paper is to introduce, model and solve a location problem with interconnected facilities. In classical location problems, the number of located facilities ($p$) is a variable, facility fixed costs are imposed and the objective is to minimize the total access cost of the users to the facilities. This problem is similar to the plant location problem. When the number of facilities is a given value $p$, then the problem is similar to the classical $p$-median problem. The covering location problem with interconnected facilities (CPIF) differs from classical location problems in the sense that the facilities must be interconnected. More precisely, two facilities are interconnected whenever the distance between them does not exceed a given limit $r$ (see Figure 1). This problem is rooted in the work of [3] who were concerned with the design of a connected network linking forest

![Network showing a set of interconnected facilities and an allocation of users.](image)

Figure 1: Network showing a set of interconnected facilities and an allocation of users.
fire-fighters who must extinguish fires in several forest parcels. The fire-fighters communicate between themselves and a central office (a fixed root) by means of short-range radios (walkie-talkies). This problem arises, for example in remote regions where mobile phone connections are not available. The range of the radio connection defines the value of \( r \). Because the designed network is connected, messages can be relayed between its nodes.

Related problems in which the facilities are interconnected are the tree of hubs location problem [2] and network design problems where the underlying connected network is not necessarily a tree [1]. In these problems, the cost of the network is accounted for in the optimization model. In contrast, in this paper, the cost of the edges linking the facilities is not considered; all that matters is that these facilities form a connected network.

Our aim is to solve the CPIF. To this end, we first introduce in Section 2 a mathematical model. Since this models can only be solved exactly for moderate sizes by commercial solvers we will develop a GRASP metaheuristic in Section 3 in order to solve larger instances. This will be followed by computational results in Section 4.

## 2 Mathematical model

The CPIF is defined in an undirected graph \( G = (N, E) \), where \( N \) is the node set and \( E \) is the edge set. The node set \( N \) is the union of a set of potential facility sites \( I \) including a root node 0, and a set of users \( J \). The set \( E \) is made up of all edges connecting two nodes \( I \) with a distance \( r \) of each other, as well as all edges connecting nodes of \( I \) and \( J \). Let \( d_k \) denote the demand of user \( k \in J \). Locating a facility at node \( i \) incurs a fixed cost \( g_i \). The cost of allocating users \( k \in J \) to a facility located at \( i \in I \) is equal to \( c_{ki} \).

In this formulation, two reverse arcs \((i, j)\) and \((j, i)\) are associated with edge \([i, j] \in E\). For each \( i \in I \), let \( y_i \) be a binary location variable equal to 1 if and only if a facility is located at site \( i \). For each \( i, j \in I, i \neq j \), let \( f_{ij} \) be non-negative flow variables. For each \( k \in J \) and \( i \in I \), the binary variable \( w_{ki} \) indicates whether user \( k \) is allocated to facility \( i \) or not. The problem can be modeled as:

\[
\begin{align*}
\text{(CPIF)} & \quad \text{minimize} & & \sum_{i \in I} g_i y_i + \sum_{k \in J} \sum_{i \in I} d_k c_{ki} w_{ki} \\
& & & y_0 = 1 \\
& & & w_{ki} \leq y_i \quad k \in J, i \in I \\
& & & y_k + \sum_{i \in I} w_{ki} = 1 \quad k \in J \cap I \\
& & & \sum_{i \in I} y_i = p \\
& & & \sum_{i \in I} w_{ki} = 1 \quad k \in J \setminus (J \cap I) \\
& & & \sum_{i \in I \setminus \{0\}} f_{0i} = \sum_{i \in I} y_i - 1
\end{align*}
\]
\[
\sum_{i \in I \setminus \{j\}} f_{ij} - \sum_{i \in I \setminus \{j\}} f_{ji} = y_j \quad j \in I \quad (7)
\]
\[
f_{ij} \leq M y_i \quad i, j \in I \quad (8)
\]
\[
f_{ij} \leq M y_j \quad i, j \in I \quad (9)
\]
\[
f_{ij} = 0 \quad i, j \in I : i \neq j, c_{ij} > r \quad (10)
\]
\[
w_{ki} \in \{0, 1\} \quad k \in J, i \in I \quad (11)
\]
\[
y_i \in \{0, 1\} \quad i \in I, \quad (12)
\]
\[
f_{ij} \geq 0 \quad i, j \in I : i \neq j. \quad (13)
\]

Constraint (1) ensures that the fixed root is open. From constraints (2), users are assigned to open facilities, and, from constraints (3), users at sites where facilities are open are not allocated to other facilities. Constraint 4 restricts the number of facilities to be equal to \(p\). Constraints (5) guarantee that each user is assigned to an open facility. Constraint (6) establishes that the root node sends \(\sum_{i \in I} y_i - 1\) units of flow, i.e., the number of located facilities. Constraints (7) impose the amount of flow entering a node of the tree to be one unit larger than the flow going out of the same node. Constraints (8) and (9) state that both ends of any edge of the tree are open facilities, where \(M\) is a large number equal to \(p - 1\). Constraints (10) state that both ends of any edge of the tree are interconnected within a distance \(r\). Finally, constraints (11)–(13) are the domain constraints.

### 3 Metaheuristic

The proposed metaheuristic is a local search algorithm. An initial solution is generated by means of a GRASP [4]. The cost evaluation for each potential facility location is based on a savings criterion. Each solution undergoes a local search phase to first minimize the total assignment cost (\(\sum_{k \in J} \sum_{i \in I} d_{ki} c_{ki} w_{ki}\)) and then the total location costs (\(\sum_{i \in I} g_i y_i\)).

Three local search are developed for the problem: an exchange operator, a removal operator and an insertion operator. The exchange operator selects a located facility in the current solution and exchanges it with another facility that is currently not in the current solution. The removal operator aims to decrease the number of facilities by removing a facility currently located. The insertion operator aims to increase the number of facilities by inserting a facility not currently located. Once this local search phase is completed, it is necessary to perturb the solution as the algorithm often yields a local optimum. The perturbation operator randomly selects \(\eta\) located facilities while ensuring that the remaining solution is feasible, and then adds \(\eta\) new facilities to the remaining solution. The process is repeated until no additional improvement is possible.
4 Computational results

The proposed metaheuristic is tested on two classes of instances. The first class of instances comprises 40 instances with 100 to 300 nodes. Those instances have no fixed facility location cost ($g_i = 0, \forall i \in I$), but there is a maximum number of facilities ($p \leq 100$). We have tested values for the maximal distance between two interconnected facilities ($r$) from 25 to 150. The second class of instances comprises 29 instances with 100 nodes. In contrast with the first class, those instances have a fixed facility location cost ($g_i = 3000, \forall i \in I$), and there is no limit on the maximum number of facilities. For those instances, we have tested values of $r$ from 500 to 10,000. Tables 1 and 2 present initial computational results.

We report the maximal distance ($r$), the average computing time in seconds taken by CPLEX to solve the problem to optimality ($t^*$) and taken by our metaheuristic ($t^{grasp}$), and the average deviation in percentage on the optimal solution for our metaheuristic ($\Delta z^*$). The initial computational results show that the proposed metaheuristic is faster than CPLEX and produces good solutions (less than 1% from an optimal solution). In addition, all instances are solved with the metaheuristic whereas only 50% and 98% of instances of class C1 and C2 are solved with CPLEX, respectively. Detailed computational results will be reported at the conference.

References


A Lagrangian-based Decomposition Method on a New Formulation of the Capacitated Concentrator Location Problem

Massimo Di Francesco¹, Manlio Gaudioso², Enrico Gorgone¹, and Ishwar Murthy³

¹Dipartimento di Matematica e Informatica, Università di Cagliari, 09124 Cagliari, Italy
²Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria, 87036 Rende, Italy
³Indian Institute of Management Bangalore, 560076 Bangalore, India, Email: ishwar@iimb.ac.in

1 Introduction

The Capacitated Concentrator Location Problem (CCLP) is a classic subject of network design and has relevant applications in areas such as computer network, logistics, health systems etc. However, CCLP is of particular interest in freight transportation. The problem is best described as the optimal design of a layered network where a central node must be connected to a set of final nodes through a set of satellite nodes (the concentrators in the CCLP parlance) [1]. A relevant application in the field of freight transportation is the shipping of pallets to a port, which represents the central node. Consider a finite set of customer locations \( N \), each of which represents an origin node for pallets. A specified number of pallets \( d_j \) has to be transported to the port from each origin node \( j \). As opposed to transporting the required pallets from each origin directly to the port, there is a benefit to strategically placing warehouses at locations where the pallets are aggregated and packed in containers, which are shipped to the port. These warehouses are referred to as concentrators. The set of potential concentrator locations is \( M \), each having a capacity \( C_i \). The cost of setting up a concentrator at location \( i \in M \) is \( f_i \). The cost of transporting \( d_j \) pallets to concentrator \( i \) from destination \( j \) is \( c_{ij} \). The decision problem is to determine which concentrator locations to set up, and then to assign each origin in \( N \) to exactly one concentrator that is set up in a manner so as to minimize the total cost, while respecting the capacity restrictions of each concentrator.

This problem can be modelled as an integer programming problem consisting of two sets of binary decision variables. One set of binary variables is \( w_i \) which is equal to 1 if a concentrator is set up in location \( i \), 0 otherwise. The other set of variables are \( x_{ij} \), where \( x_{ij} = 1 \) if customer location \( j \) is assigned to concentrator \( i \), 0 otherwise. Using these two sets of variables, one can adopt a well-known formulation which can be found in [4, 5].

The literature on CCLP is extremely wide and diverse methodologies, both exact and approximate, have been applied. Among the others we mention the Lagrangian relaxation approach [2] and the discretization method [5]. In the talk, starting from a formulation introduced in [5], we describe a reformulation of classic CCLP whose relaxation is stronger
than the classic formulation and which is suitable for a Lagrangian-based decomposition scheme.

2. A New Disaggregated Formulation of CCLP and its Lagrangian Decomposition

We now introduce a new formulation of CCLP that induces the notion of cardinality associated with each concentrator that is set up. Specifically, let \( k_i \) denote the number of customer locations that are assigned to concentrator location \( i \). Further let \( K_i \) denote the maximum number of customer locations that can be assigned to \( i \). This number can be determined easily by using the notion of bin packing, whereby customer locations are sorted by \( d_j \). Given \( C_i, K_i \) can be obtained easily.

Two types of decision variables are defined in the proposed formulation:

\[ y_{ik} = 1 \text{ if } k_i \text{ number of customer locations are assigned to concentrator } i, \]
\[ = 0 \text{ otherwise.} \]

and

\[ z_{jk} = 1 \text{ if customer location } j \text{ is assigned to concentrator } i \text{ having } k_i \text{ customer locations assigned to it,} \]
\[ = 0 \text{ otherwise.} \]

Given the above decision variables, the new formulation is:

\[
\begin{align*}
\text{(P_{z,y})} \quad & \text{Minimize} & \sum_{i \in M} \sum_{j \in N} \sum_{k=1}^{K_i} c_{ij} z_{jk} + \sum_{i \in M} \sum_{k=1}^{K_i} f_i y_{ik} \\
\text{s.t.} & \quad & \sum_{k=1}^{K_i} z_{jk} = 1 & \forall j \in N \\
& & z_{jk} \leq y_{ik} & \forall i \in M, j \in N, k_i = 1, \ldots, K_i \\
& & \sum_{j \in N} d_j z_{jk} \leq C_i y_{ik} & \forall i \in M, k_i = 1, \ldots, K_i \\
& & \sum_{j \in N} z_{jk} = k_i y_{ik} & \forall i \in M, k_i = 1, \ldots, K_i \\
& & \sum_{k=1}^{K_i} y_{ik} \leq 1 & \forall i \in M \\
& & z_{jk}, y_{ik} \in \{0, 1\} & \forall j \in N, \forall i \in M
\end{align*}
\]

In the above formulation, (2) enforces the condition that each customer location is assigned to exactly one concentrator location with a cardinality of \( k \). Constraints (3) enforces the condition that if a concentrator with a given cardinality is not set up, then the corresponding
assignment of customer location \( j \) to that concentrator-cardinality combination is not possible. Constraints (4) enforces the capacity constraints on each concentrator with a given cardinality. Constraints (5) specifies the cardinality for each concentrator, while (6) enforces the requirement that at most one cardinality type can be specified for each concentrator.

In this study, we first show that the lower bounds obtained from the LP relaxation of \((P_{xyz})\) is stronger than that obtained from the traditional formulation as described in [4] and [5]. Further, we also describe some inequalities that are unique to \((P_{xyz})\), which further strengthens the formulation when they are added selectively. Finally, we present a Lagrangian relaxation scheme by dualizing constraints (2) and (5) with Lagrangian multipliers \( \pi_i \) for \( i \in \mathbb{N} \) and \( \theta_{ik} \) for \( i \in \mathbb{M} \) and \( k_i = 1, \ldots, K_i \), respectively. Then, by representing the multipliers in vector compact form as \( \pi \) and \( \theta \), respectively, we obtain the Lagrangian relaxation subproblem \( \text{LR}(\pi, \theta) \) with objective function

\[
\hat{v}(\pi; \theta) = \text{Minimize} \quad \sum_{i \in \mathbb{M}} \sum_{j \in \mathbb{N}} \sum_{k_i = 1}^{K_i} \hat{c}_{ijk} z_{ijk} + \sum_{i \in \mathbb{M}} \sum_{k_i = 1}^{K_i} \hat{f}_{ik} y_{ik} \tag{8}
\]

s.t.

\[
\sum_{i \in \mathbb{M}} d_j z_{ijk} \leq C_i y_{ijk} \quad \forall i \in \mathbb{M}, k_i = 1, \ldots, K_i \tag{9}
\]

\[
\sum_{i \in \mathbb{M}} y_{ik} \leq 1 \quad \forall i \in \mathbb{M} \tag{10}
\]

\[
z_{ijk}, y_{ik} \in [0, 1] \tag{12}
\]

where \( \hat{c}_{ijk} = c_{ij} + \pi_j + \theta_{ik} \) and \( \hat{f}_{ik} = f_i - k \theta_{ik} \).

Following the same approach of [3], we define, for each \( (i; k_i) \), the binary knapsack problem:

\[
g_{i,k_i}(\pi, \theta) = \text{Min} \sum_{j \in \mathbb{N}} \hat{c}_{ijk} z_{ijk} \tag{13}
\]

s.t.

\[
\sum_{i \in \mathbb{M}} d_j z_{ijk} \leq C_i \tag{14}
\]

\[
z_{ijk} \in [0, 1] \quad \forall j \in \mathbb{N} \tag{15}
\]

And thus the problem (8)-(12) can be equivalently written as:

\[
\sum_{i \in \mathbb{M}} \min_{k_i = 1, \ldots, K_i} \left( \min_{y_{ik} = 0, \ldots, K_i} \left( g_{i,k_i}(\pi, \theta) + \hat{f}_{ik} y_{ik} \right) \right)
\]

where the inner problem can be solved by branching on the single variable \( y_{ik} \). We propose a Lagrangian heuristic which operates by maximizing function \( \hat{v}(\pi; \theta) \) by means of the subgradient method and by repairing, at each iteration of such method, the solution, possibly
infeasible, of the relaxed problem. At the end of such heuristic the best solution found is returned.

References

1 Introduction

We consider a vehicle routing problem (VRP) in which the demands of the customers are stochastic. This problem is known as the vehicle routing problem with stochastic demands (VRPSD). In the VRPSD, each vehicle is allowed to perform replenishment trips to the depot, in order to increase the total capacity available for a route. These restocking trips might be reactive, i.e., triggered by a stockout at some customer, or preventive, i.e., performed in anticipation of a probable stockout later in the route. It is well-known that, given an a-priori route, optimal restocking decisions can be computed with a stochastic dynamic programming algorithm. We consider the VRPSD under optimal policy, which calls for the identification of a set of routes visiting all customers with minimum total expected cost, considering that optimal restocking decisions are always made.

Over the years, many exact algorithms have been developed for the VRPSD under the reactive, or detour-to-depot, policy. Under this policy, the vehicle is not allowed to perform preventive restocking trips, even if there is a high stockout probability at some customer located far from the depot. Recently, the first approaches addressing the VRPSD under optimal restocking have been proposed. While these efforts are noteworthy, the distributional assumptions on the customer demands render them unsuitable for most situations one might encounter in practice.

We close this gap in the literature by designing a branch-cut-and-price algorithm for the VRPSD. The main component of our algorithm is an efficient labeling procedure for pricing profitable columns. This procedure leverages on the stochastic dynamic programming algorithm for route cost evaluation. Exact and approximate dominance rules are proposed.
and applied to control the combinatorial growth of labels, together with two reduced cost bounding procedures.

In summary, in this talk we present the following contributions: (1) the first algorithm for solving the VRPSD under optimal restocking and general demand distributions is proposed. This algorithm allows us to solve, for the first time, literature instances with no adaptation, i.e., by just considering the customer demand value as the parameter of a probability distribution. Instances of moderate size (up to 75 nodes) can be solved in less than 5 hours, and larger instances (up to 148 nodes) can be solved in long-runs of the algorithm; (2) by measuring the value of the stochastic solution of all literature instances solved, we determine how much better the optimal solution is, as compared to the solution obtained by solving the deterministic equivalent. Indeed, by taking stochasticity into consideration, one is able to improve the expected solution cost by up to 9.5%. This simple measure has never been computed for the VRPSD, in part because the non-literature instances solved by current methods have unknown deterministic optima; (3) we measure the value of allowing load factors greater than 1, meaning that the total expected demand in a route is allowed to exceed the capacity of the vehicle. This scenario has been largely ignored by researchers, even though, as we show, it is possible to improve significantly the objective value by allowing load factors slightly higher than 1.

2 Related Literature

For brevity, we only mention the approaches for the VRPSD which assume optimal restocking. The first approach in the literature is the one from [1]. The proposed algorithm, based on the integer L-shaped method, could solve relatively large instances of the problem. Unfortunately, the main method for bounding the restocking cost assumed the customer demands to be identically distributed. This assumption is lifted in [2], who developed bounds that are valid for the case of discrete and bounded distributions. Even though, the demand distributions used in the experiments are still simplified triangular distributions, with only up to 5 possible values per customer. When a single vehicle is considered, the problem can be modeled using Markov decision theory. The unified mixed-integer model presented in [3] computes, simultaneously, optimal tours and restocking policies. Unfortunately, the dimension of the model grows polynomially with the capacity of the vehicle. For this reason, only simplified instances can be solved.

3 Branch-Cut-and-Price Algorithm

Our branch-cut-and-price algorithm separates cuts defined over the variables of an arc-based compact formulation, and computes the linear bound by applying column generation on a path-based extended formulation. In branch-and-price approaches for vehicle routing
problems, the main difficulty is usually the solution of the pricing problem. In our case
this difficulty is magnified, since the cost of a route is computed by stochastic dynamic
programming. The main component of our algorithm is the set of procedures, dominance
rules and reduced cost bounds that enables the efficient solution of the pricing problem.

3.1 \textit{ng}-Route Relaxation and Label Dominance.

If measures to control the combinatorial growth of labels are not taken, solving the pricing
problem becomes a computationally intractable task. In order to propose label dominance
rules, we relax the route elementarity constraint. The \textit{ng}-route relaxation [4] is generally
regarded as the best route relaxation for using in a branch-and-price context, since it can
be efficiently implemented, and its impact on the linear bound is relatively small [5]. When
elementarity is relaxed, we are able to propose exact and heuristic dominance rules. By
applying solely heuristic dominance, we transform the whole approach in a heuristic. In
fact, our results show that such heuristic could find the optimal solution in all instances
where the optimum is known.

3.2 Reduced cost bounds.

Reduced cost bounds, or completion bounds, are lower-bounds computed on the reduced
cost of all \textit{ng}-feasible route extensions from a particular label. These bounds are useful
for accelerating the solution of the pricing problem, since a label can be safely discarded if
it has a nonnegative completion bound. In this talk, we discuss the two following bounds:

\textbf{Forward RCSP bound.} By the triangle inequality, the expected cost of a route
is bounded from below by its a-priori cost. The RCSP bound explores this property,
computing a completion bound by underestimating the contribution of the expected route
cost to the reduced cost of a column.

\textbf{Knapsack bound.} Another completion bound can be calculated by solving a knap-
sack problem, where the capacity of the knapsack is given by the remaining load of the
vehicle, and the values of the knapsack items given by the dual values of the partitioning
constraints and rounded capacity cuts.

4 Computational Results

We first performed experiments on several literature instances. In addition, we compare
results with [1] and [6], where the detour-to-depot policy was assumed. We also report
results of the experiments with different load scenarios and results obtained in long-runs
of the algorithm on larger instances.

We tested the algorithm on instances of the sets \textit{A}, \textit{E}, \textit{P}, and \textit{X} of the \textsc{CVRPLIB} [7]. The
demands of the customers were assumed Poisson distributed, with the parameter as given
by the deterministic demand values in the instances. The time limit was set to 5 hours. The largest instance that could be solved to optimality contains 75 nodes.

The instances proposed by [1] exhibit a high ratio of average number of customers per vehicle, a characteristic that favors pure branch-and-cut solution methods. We tested the branch-cut-and-price algorithm on the 32 instances with up to 51 nodes. Differently from [1], we do not use bounding procedures tailored for the case of identically distributed demands. Still, we were able to close 4 instances that could not be solved by their integer L-shaped algorithm. Interestingly, the value of the stochastic solution in these instances is well below the ones obtained in the literature instances. We discuss a few intuitive explanations for this result.

In [6], several literature instances were solved assuming the detour-to-depot policy, and Poisson distributed demands. By comparing the optimal solution values, we were able to quantify the cost savings when adopting the optimal policy.

Relaxations of the load constraint, even if slight, have the undesirable effect of making the problem more difficult. We experimented only with the instances that could be solved relatively fast for a load factor of 1. For each instance, we experimented with 6 additional load factors ranging from 1.05 to 1.50. Finally, long-run experiments were performed, which enabled the solution of VRPSD instances of up to 148 nodes.

5 Conclusions

We considered the VRPSD under optimal restocking, and proposed the first algorithm for solving the problem under general distributional assumptions. Computational experiments performed on several literature instances confirmed the effectiveness of the branch-cut-and-price approach. Instances of moderate size (up to 75 nodes) could be solved in reasonable time (up to 5 hours), and larger instances (up to 148 nodes) could be solved in long-runs of the algorithm.

Our results indicate that the value of the stochastic solution (VSS) is higher in instances with a low average number of customers per vehicle. With few exceptions, most instances with a high customers per vehicle ratio (above 12) have a VSS near, or in many cases equal to, zero. Therefore, solving the stochastic problem (as opposed to solving the deterministic equivalent) makes more practical sense in instances with a low number of customers per vehicle. In these instances, gains of up to 9.5% in the solution cost might be attained when stochasticity is explicitly considered.

Finally, we performed a range of experiments assuming load factors greater than 1. The VRPSD has been traditionally studied under the assumption of a fixed load factor of 1, even when optimal restocking is adopted. By allowing slightly higher load factors, it is possible to improve the objective value significantly.
References


Branch and Price for Probabilistic Routing

Felipe Lagos*, Mathias A. Klapp†, Alejandro Toriello*

*Georgia Institute of Technology, Atlanta, Georgia, USA
†Pontificia Universidad Católica, Santiago, Chile
Email: atoriello@isye.gatech.edu

1 Introduction and Problem Statement

The Vehicle Routing Problem (VRP) and its variants are widely studied within the OR and transportation science communities. A priori or probabilistic VRPs [1] are particular versions of VRPs with uncertain parameters in which route plans are specified before uncertainty is realized and only modified via simple recourse rules for any realization of the uncertainty. Probabilistic VRP models are fundamental building blocks for analyzing and solving more complex dynamic and stochastic VRPs, and are also natural for problem settings where it is infeasible or impractical to plan optimal routes a posteriori.

We study a novel branch-and-price approach for the a priori VRP with probabilistic customers (VRP-PC). In this problem, the subset of customers requiring service is random and follows a known probability model. In the first stage, the decision maker plans a set of vehicle routes dispatched from the depot and visiting all potential customer locations. Vehicles are dispatched in the second stage after observing the subset of customers requiring service. The second-stage recourse rule modifies first-stage routes by skipping locations without a service requirement, maintaining the established sequence of each route for the visited customers. The objective is to minimize the expected vehicle travel cost, accounting for this recourse rule.

Let $C = \{1, \ldots, n\}$ be a set of customer locations, and let $N = C \cup \{0, n + 1\}$, so that routes start at 0 and end at $n + 1$, both identified with the depot. Each customer $i \in C$ independently requires service with probability $p_i \in (0, 1]$. When requiring service, customer $i \in C$ has integer demand $d_i$ that must be served by a single vehicle route, where the vehicle capacity is $q \geq \max_i d_i$; the vehicle capacity constraint (and any other route constraint) must be satisfied with certainty, regardless of customer realizations. Traveling from $i$ to $j$ costs $c_{ij} \geq 0$, and these costs satisfy the triangle inequality. We can accommodate any other constraints amenable to column generation, such as service time windows, route duration limits, etc., but we do not specify them here for brevity.

Let $R$ be the set of feasible routes; each $r \in R$ is a path in $N$ starting at 0, ending at
\( n + 1 \), and satisfying vehicle capacity and any other route constraints. If \( n_r \) is the number of customer visits in \( r \), we use \( r(i) \) to denote the node in the \( i \)-th position, with \( r(0) = 0 \) and \( r(n_r + 1) = n + 1 \). Since the vehicle skips customers that do not realize, the expected cost of route \( r \) is given by

\[
E(c_r) = \sum_{k=0}^{n_r-1} \sum_{i=0}^{n_r-k} c_{r(i),r(i+1+k)}P_{r(i),r(i+1+k)} \prod_{j=1}^{k} (1 - p_{r(i+j)}) =: \sum_{k=0}^{n_r-1} Y_r^k, \tag{1}
\]

where \( p_0 = p_{n+1} = 1 \), and the term \( Y_r^k \) groups all costs associated with arcs that skip \( k \) customers. As is common in VRP set partitioning relaxations, we may consider routes that repeat customers (possibly also eliminating cycles up to some length); definition (1) still applies if we define \( c_{r(i),r(j)} = 0 \) whenever \( r(i) = r(j) \). A set partitioning relaxation for the VRP-PC is then given by

\[
\min \sum_{r \in R} y_r E(c_r) \text{ s.t. } \sum_{r \in R} \alpha_i^r y_r = 1, \quad i \in C, \tag{2}
\]

where \( \alpha_i^r \) indicates the number of times \( r \) visits \( i \), which may be greater than 1 if we allow repeat visits.

## 2 Column Generation and Algorithm

The columns of (2) are difficult to price exactly with dynamic programming, as they require recording the entire partial route. Consider instead the expected cost under-approximation

\[
E_\ell(c_r) := \sum_{k \leq \ell} Y_r^k \leq E(c_r),
\]

which only considers arcs that skip \( \ell \) customers or fewer. Define \( \hat{c} := \max_{i,j} c_{ij}, \hat{p} := \max_i p_i, \bar{p} := \min_i p_i; \) for any \( y \) feasible in (2), we prove the guarantee

\[
\sum_{r \in R} y_r E(c_r) \leq \sum_{r \in R} y_r E_\ell(c_r) + \hat{c} \hat{p}^2 \sum_{k=\ell+1}^{n-1} (n-k+1)(1-\bar{p})^k =: \sum_{r \in R} y_r E_\ell(c_r) + \delta_\ell. \tag{3}
\]

For a given \( \ell \), the \( \delta_\ell \) value provides an a priori guarantee on the tightness of the approximation given by \( E_\ell \) in (2); furthermore, if we replace \( E \) with \( E_\ell \) in (2) and let \( R \) include all routes that allow repeat visits but exclude cycles of length \( \ell \) or shorter, we can price columns via a modification of the resource-constrained shortest path problem with \( \ell \)-cycle elimination [2].

For a given \( \ell \geq 0 \), define algorithm UCA as the solution of (2) with \( E_\ell \) replacing \( E \), and with \( R \) excluding \( \ell \)-cycles but otherwise allowing repeat visits.

**Theorem 1.** For any \( \ell \), UCA returns a lower bound for (2). For any additive optimality tolerance \( \epsilon > 0 \), the solution returned by UCA is within this gap for a large enough \( \ell \) satisfying \( \delta_\ell \leq \epsilon \). By embedding UCA in a branch-and-price framework, the integer solution returned by this algorithm is within an additive optimality gap of \( \delta_\ell \).
To speed up solution time, UCA can be warm-started with a lower value $\ell_0 < \ell$, pricing out routes that allow shorter cycles and using the coarser cost approximation $E_{\ell_0}$, and the parameter can be gradually increased until reaching $\ell$. Under these conditions the value returned by UCA may no longer be within $\delta_\ell$ of (2) (since we potentially include some routes with shorter cycles), but the guarantee on the integer solution still holds. Note also that the gap guarantee provided by $\delta_\ell$ is conservative, and we can obtain a tighter a posteriori gap by evaluating the solution with the two objectives, $E_\ell$ and $E$.

We also propose a second algorithm, FCA, in which we price out columns using $E_{\ell_0}$ but include them using the true expected cost $E$. This algorithm is not guaranteed to yield a lower bound on (2), but a similar gap guarantee holds for an integer solution obtained with FCA embedded in a branch-and-price framework. We omit the details here for brevity.

### 3 Results

To test our algorithms, we used instances of the VRP-PC with time windows based on the Solomon benchmark set [3]. We created 1,710 total instances with $n \in \{15, 25, 40\}$ and customer probabilities uniformly set to $p \in \{0.5, 0.7, 0.9\}$. For each instance, we ran the branch-and-price algorithm twice, using either UCA or FCA at each node, with values $\ell_0 = 0$ and $\ell \in \{0, \ldots, 4\}$. We used CPLEX 12.4 running in the Georgia Tech ISyE cluster, and set a 6-hour time limit per run.

Table 1 shows averages of the highest $\ell$ parameter for which we were able to get an integer solution via branch-and-price within three given relative gaps, 0% (provably optimal), 5% and 10%. (The relative gap here is with respect to the approximate objective $E_\ell$ and the best bound from the branch-and-bound tree.) Evidently, the problems become easier as $p$ grows, and more difficult as $n$ grows.

<table>
<thead>
<tr>
<th>Customers</th>
<th>Probability</th>
<th>UCA</th>
<th>FCA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0%</td>
<td>5%</td>
<td>10%</td>
</tr>
<tr>
<td>15</td>
<td>0.5</td>
<td>3.45</td>
<td>3.83</td>
<td>3.95</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>3.67</td>
<td>3.9</td>
<td>3.97</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>3.87</td>
<td>3.9</td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td>0.5</td>
<td>2.37</td>
<td>3.08</td>
<td>3.68</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>2.45</td>
<td>3.11</td>
<td>3.74</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>2.71</td>
<td>3.11</td>
<td>3.58</td>
</tr>
<tr>
<td>40</td>
<td>0.5</td>
<td>1.5</td>
<td>2.19</td>
<td>3.38</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>1.38</td>
<td>2.69</td>
<td>3.25</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.62</td>
<td>3.38</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 1: Average of highest $\ell$ parameter solved within time limit, by algorithm and gap value.

In terms of the true expected cost objective function, we empirically find that either
algorithm’s solution has expected cost within 5% of the optimum, on average, based on the
best bound provided by UCA, regardless of $\ell$, and in many cases this gap is much smaller,
especially for higher values of $\ell$. In addition to $\ell$, the factors influencing the solution’s
quality include $p$, $n$ and the topology of the underlying instance; in particular, instances
with customers clustered together tend to be more difficult compared to instances in which
the customers are more randomly and evenly distributed. For these latter instances, even
at $n = 40$ our solutions are within 0.1% of optimal for $\ell \geq 3$.

Finally, we remark on the structure of solutions for the VRP-PC obtained by our
algorithms. Figure 1 compares two solutions for an instance with $p = 0.5$: The first is
the solution to the deterministic VRP, while the second is the solution obtained by our
UCA algorithm with $\ell = 4$. Although the two only differ by about 3% in expected cost,
we see that the VRP-PC solution groups customers into routes in a very different way,
including several overlapping routes. The solution highlights potential counter-intuitive
operational implications of accounting for uncertainty and probabilistic customers when
planning vehicle routes.

Figure 1: Deterministic (left) and UCA (right, $\ell = 4$) solutions for instance with $p = 0.5$.

References


[2] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and

High-dimensional dependent random travel times in vehicle routing

Stein W. Wallace
Department of Business and Management Science,
NHH Norwegian School of Economics
Email: stein.wallace@nhh.no

Zhaoxia Guo
Business School of Sichuan University, China

Michal Kaut
SINTEF Technology and Society, Trondheim, Norway

1 Dependent random variables

Vehicle routing, in all its variants, is one of the most studied problems in logistics [7, 9, 1, 10, 2]. But for historical as well as numerical reasons, the vast majority of papers are on deterministic problems. Stochastic versions have started to occur, see for example the reviews [9, 3, 8, 11].

We study how to model and handle stochastic travel times (speeds) in two-stage stochastic vehicle routing problems. The first stage is to pick a set of routes for a set of vehicles (the actual requirements of the routes will vary with the chosen VRP). The second stage is to follow these routes (again following rules that will depend on the VRP being studied) taking properly care of the realized speeds. The objective is to minimize overall expected costs, where costs stem from both the routes themselves and the expected travel costs (which may depend on fuel consumption, distance, time spent and customers visited). Contrary to existing literature, we allow these speeds to be stochastically dependent in time and space, that is, the speed on one link in one period will be correlated to speeds on the same link in nearby time periods, as well as speeds on neighboring links in the same time and next period. Hence we are handling a very high-dimensional dependent random vector. We discuss how such vehicle routing problems should be modeled in time and space, how the random vector can be represented, and how scenarios (discretizations) can meaningfully be generated to be used in a stochastic program. We assume that the stochastic vehicle-routing problem is being solved by a heuristic, and focus on the objective function evaluation for any given solution. Numerical procedures are given and tested. As
an example, our largest case, representing Beijing, has 142 nodes, 418 road links and 60
time periods, leading to 25,080 dependent random variables. This is close to the limit of
what we can handle on a standard portable PC due to memory limitations, where memory
operations dominate execution times.

1.1 The underlying network

We chose to model random speeds rather than travel times. This has practical reasons, in
particular that speeds are nicely bounded from above and below, whereas travel times are
not, leading to a simpler process of scenario generation. One major issue when modeling
dependent stochastic speeds in a VRP is whether or not we can operate on the standard
network of customer nodes, as is classically done in deterministic models. We demonstrate
why this, at best, is very cumbersome when handling dependencies, as dependencies aris-
ing from customer node-pairs sharing edges (road links) cannot be properly described in
such a framework. In addition, data is normally (though not always) collected on road
links, not customer node-pairs (for example as part of toll systems or other traffic control
mechanisms). Note that even if all road link speeds were independent, customer node-pair
speeds would not, and that assuming that all customer node-pair speeds are independent
amounts to assuming that no two customer node-pairs ever share a link. Finally, and
maybe most importantly, with discrete times, the distribution of speeds for a given cus-
tomer node-pair would depend on when a trip starts (not just which period it starts in).
This would create enormous difficulties in representing stochastic speeds at customer node
level. This problem does not occur when speeds are by link and time period. Hence it is
necessary to operate on the underlying network (map). All this is in line with [4], and we
follow their logic.

So the setup is that a VRP is solved using a heuristic where one part determines routes
and another finds the costs associated with the routes. Our contribution is the latter part.
A route is a sequence of customers (first-stage decision), and during operations (second-
stage decision), the vehicle has to select a path from one customer to the next, based
on present traffic conditions. This is what [4] calls path selection. Our approach allows
many ways to pick the path, the two most interesting probably being picking the path
with the shortest expected travel time, or the one with shortest travel time given speeds
at the time of making the choice. We do not assume foresight. We then calculate the
distribution of travel times based on the underlying stochastics. The goal of the model
is to find the sequence of customers, but to do so, we need to describe how the vehicles
will operate. In reality, they will probably use some existing service, such as Google, but
that cannot be put into a VRP directly. Using expected speeds or speeds at the time of
decision-making are therefore two possible choices, which might be used in reality, but
more likely are approximations of what will actually be done. There is no such thing as
an optimal set of routes without a reference to how path selections are made.

1.2 Scenario generation

The next major issue to discuss is how to represent the dependent random variables, in particular the correlations. In our largest cases we face over 300 million distinct correlations, and using correlations is not even enough to fully describe a distribution. Clearly, neither calculating nor storing that kind of data sets are feasible. Also, we demonstrate why guessing (at least for academic use) a 25,000-dimensional dependent distribution is close to impossible, and even more so if we require the given distribution to make sense (have meaningful dependencies). And if we don’t have a distribution, we cannot even sample, as sampling requires something to sample from. Our approach, and that is the main contribution of this work, is to show how we can use an existing scenario generation method by [5] to generate scenarios based on a subset of the correlations, and still be able to control the quality of the final solution. The chosen method takes marginal distributions for link speeds plus a subset of the speed correlations, and generates a set of scenarios that have exactly these marginal distributions and correlations. We pick correlations between pairs of speeds on the same link in neighboring time periods, and neighboring links in the same and next period. This is exactly the way a route is set up in a VRP; they follow a sequence of links in time and space totally within this set. Other correlations are not controlled. So although, for any route, the expected travel times is correct, the distribution of travel times can (and generally will) be incorrect due to spurious correlations between speeds on links that are not neighbors. However, we control these errors in our stability test, a test developed from the ideas in [6], adjusted for the enormous dimensions of our problem. We typically achieve an accuracy of 1%, with only fifteen scenarios. Practically, this means that one function evaluation of a heuristic for a stochastic VRP will take about 15 times longer than for the deterministic version of the same problem, once the scenarios are generated.

1.3 Results

We present results for several different VRPs with random speeds. As an example, consider the case from Beijing with 142 nodes, 418 road links and 60 time periods, leading to 25,040 dependent random variables, and where the costs stem from expected driver overtime pay. Within this context we control about 264,000, out of over 300 million, distinct correlations, and end up with an accuracy of 1% with 15 scenarios. This is a case where the VSS (the Value of the Stochastic Solution) is about 10%. Generating the scenario tree (which has about 375,000 entries) took in the order of five hours, and this is a fixed cost for a given map (this is also a reason for using the map rather than just the customer nodes if the customer node set is not stable, but the map is). The increase in CPU time per iteration
of the heuristic is 15-fold for the function evaluation in the stochastic setting, as we have 15 scenarios.

References


Fast prediction of solutions to an integer linear program using machine learning

Eric Larsen, Sébastien Lachapelle
DIRO and CIRRELT, Université de Montréal, Montreal, Canada

Yoshua Bengio
DIRO and MILA, Université de Montréal, Montreal, Canada

Emma Frejinger (corresponding author)
DIRO and CIRRELT, Université de Montréal, Montreal, Canada
Email: emma.frejinger@cirrelt.ca

Simon Lacoste Julien
DIRO and MILA, Université de Montréal, Montreal, Canada

Andrea Lodi
Polytechnique Montréal, Montreal, Canada

February 23, 2018

1 Introduction

Operations research (OR) has been successful in developing methodologies and algorithms allowing to solve efficiently various types of decision problems that can be formalized but are nevertheless too complex or time consuming for humans to process. These methodologies and algorithms are crucial to a huge variety of applications. Machine learning (ML), on the other hand, and deep learning in particular, has had remarkable success in automating tasks that are easy to accomplish but difficult to formalize by humans, for example, image analysis, natural language processing, voice and face recognition. Through this undertaking, ML has developed an array of powerful classification and regression methods that can be used to approximate generic input-output maps.

This paper focuses on a topic at the intersection of ML and discrete optimization. Our objective is to define a ML methodology that allows to predict descriptions of solutions to combinatorial optimization problems. We focus on problems for which a deterministic optimization model is available and nevertheless cannot be used in an application due to (i) imperfect knowledge of problem instances and (ii) computational limits.

The level of detail chosen to describe the solution depends on the application. The
objective function value and the set of values taken by all decision variables correspond respectively to the most global and most detailed descriptions. We focus on predicting intermediate descriptions providing more information than the sole objective function value without supplying the solution in extenso. Due to its discrete nature and possibly high dimensionality, predicting the solution – a structured, classification or discrete-valued regression problem – is a more complex task than predicting the value of the objective function – an unstructured real-valued regression problem.

The proposed methodology is based on the approach known in ML as supervised learning (making use of labeled data) and we can view the problem at hand as a classification or regression problem. It consists of fours steps: (i) Problem instances (input) are sampled from the space relevant to the application and solved using an existing optimization model and a solver. We can in this manner generate as much data as the computational budget allows. (ii) Detailed solutions (output of the solver) are synthesized according to the chosen description. (iii) A ML model-algorithm pair is implemented based on the generated input-output data. (iv) This approximator is used to generate predictions for the solutions of actual problem instances. The predictions are expected to be delivered with high speed, high accuracy, as well as low marginal cost in terms of data, memory and computing requirements. Given the relatively high fixed cost associated with data generation and training of the ML approximator, the goal of a low average cost can be achieved for applications where the predictions must be used with a high frequency. We consider such an application.

The problem of predicting the optimal objective function values of various types of optimization problems has been the focus of many studies, in particular in the literature on reinforcement learning or approximate dynamic programming. We aim to predict description of optimal solutions without reinforcement. [2] consider a context that is similar to ours but they but focus on predicting the objective function value. The literature on predicting descriptions of solutions is however scarce. Closest to our work is [3] that describes the prediction of detailed solutions to the planar Traveling Salesman Problem assuming perfect information at the time of prediction.

A wide range of model-algorithm pairs can be considered. We model the approximation problem at hand both as a classification problem and as a regression problem and compare the predictive performance of low-capacity logistic/linear regression models with that of high-capacity regression/classification deep feedforward neural networks. While the basic idea behind the methodology is simple, some important challenges arise. The most important one being that the information on a subset of the input (problem instances) is unknown at time when solutions are predicted. Instead of solving a stochastic program, we use the deterministic model and a solver to compute a large number of solutions. The training data is hence based on perfect information. Nonetheless, we account for the
stochasticity in uncertain inputs through aggregation. We contribute to the literature by showing that the proposed methodology allows to build an accurate and fast predictor for a description of optimal solutions to an integer linear programming (ILP) formulation of the load planning problem under imperfect information.

2 Application

Our application concerns the decision of accepting or rejecting a container booking on double stack intermodal trains. These booking decisions depend on the so-called load planning (LP) problem: Given a set of containers and a set of railcars, determine the subset of containers to load and the exact way of loading them on a subset of railcars. The objective is to minimize the costs of unloaded containers and empty slots. Due to the combinatorial nature of the LP problem, the effective capacity of railcars depends on the characteristics of the set of containers. Containers are characterized by their length, height, type, content and gross weight. Double-stack railcars are characterized by the number, length, weight holding capacity of their platforms and by their loading capabilities (set of loading patterns describing the possible ways in which containers of diverse lengths can be placed in the lower and upper slots, i.e. positions, of each platform).

The booking application requires, in very short computational time (fraction of a second), to determine if the available railcars have enough capacity to carry fully or partially a set of containers. Container weights, crucial to the LP problem, are unknown at the time of booking. This problem is of practical importance and it features characteristics that make it useful for illustrating the proposed methodology. Indeed, we can use a commercial solver to solve an ILP formulation of the problem [1] in reasonable time (seconds to minutes) under perfect information. Nevertheless, the formulation cannot be used for the application because of the restricted computational budget (fraction of a second) and the unknown container weights.

At the time of prediction, the following features of the LP problem are available: numbers of containers of each type and number of available railcars of each type. The following features of the solution are required: numbers of containers of each type that can be loaded and number of railcars of each type that are used. Hence, a problem-solution map relating the former (input) and the latter (output) sets of features is well-defined. Since exact container weights are unavailable, aggregation over container weights must be performed either before, through or after ML approximation.

3 Training and results

We generated 19 million problem instances with numbers of platforms and containers ranging from 1 to 50 and 1 to 150. We randomly sampled containers from 2 sizes, as well as their weights, and railcars from 10 standardized types. Each problem instance was solved with the ILP solver and the resulting set of problem-solution pairs was divided into
training (64%), validation (16%) and test (20%) sets. Two models were considered for approximating the problem-solution map. First, a multilayer perceptron (MLP), a.k.a. deep feedforward neural network, consisting of 12 input units (size of input), 10 hidden layers of 600 units each with rectified linear units for activation, output layer of 544 units and 12 softmax outputs (size of output). Second, the base model, a feedforward network with identical input and outputs but without hidden layers, a.k.a. logistic regression (LR) or logit. In both cases, inequalities between output and input values for each feature were imposed and aggregation over container weights was performed implicitly. Training of the models was accomplished with pseudo likelihood maximization under the assumption that outputs are conditionally mutually independent given inputs. Optimization was performed with batch stochastic gradient descent and learning rate adaptation was governed by the Adam (adaptive moment estimation) method. Regularization was ensured by early stopping. Training the MLP required approximately 32 minutes on a GPU.

Once the models have been trained, computing the predicted output values for particular input values requires negligible time. Predictive performance was measured with mean absolute prediction error (MAE) overall, with respect to slots only and to containers only. Over the complete test set (3.8M instances), MAE is 5.20 and 3.15 for LR and MLP respectively, meaning that the best model (MLP) only makes an average error of 3.15 slots/containers. MAE with respect to slots is slightly lower (1.21 for MLP and 2.15 for LR) than for containers (1.94 for MLP and 3.05 for LR).

We performed a detailed analysis of the errors incurred over partitions of the test set. We observed that the high-capacity MLP consistently exhibits a superior performance over the range of numbers of platforms and containers: The difference in performance between MLP and LR increases with the complexity of the problem instances. Not unreasonably, the error made in predicting the number of loaded containers is higher when there is excess demand (difference between containers awaiting loading and available slots). Conversely, the error made in predicting the number of used slots is lower when there is excess demand.

References


1 Introduction

Granular search (GS) is one of the most popular speedup techniques in metaheuristics for vehicle routing problems (VRPs). The main idea is to speed up the search by only generating the most promising parts of the neighborhoods in each step of the local search \[1\]. A move is uniquely defined by a so-called generator arc in combination with a neighborhood operator: after applying the move, the generator arc is part of the solution. Sparsification methods are used to sort all arcs according to a certain criterion that indicates the desirability of an arc, e.g., the arc length or reduced cost values of a relaxation of the problem, see, e.g., \[2\]. The resulting list is truncated at a certain point determined by a granularity threshold, and only the remaining entries are used as generator arcs. Thus, the insertion of at least one promising arc in each iteration is guaranteed.

Because the selection of the generator arcs is crucial for the performance of granular search, we investigate machine learning techniques to predict the optimal granularity threshold in each iteration given a specific sparsification method and neighborhood operator. The optimal granularity threshold truncates the list directly after the generator arc leading to the maximal gain. We study this idea on the multi-depot VRP (MDVRP), in which a set of customers have to be optimally served by a vehicle fleet stationed at multiple depots. To estimate the potential of machine learning components, the first aim of our research is to determine the value of perfect information, i.e., investigating the speed
up that can be gained from knowing the optimal granularity threshold in advance.

Section 2 shows the algorithmic framework we use to identify the potential of perfect information. Section 3 gives a description of the computational experiment and showcases the test results. Finally, in section 4, we present our conclusion and next steps.

2 Algorithmic framework

The initial solution is generated by an insertion heuristic. In each iteration, an unrouted customer either forms a new route or is inserted in an existing route the way which increases the objective function the least.

As a framework for our computational experiment a local search with a composite neighborhood $\Theta$ is used, which consists of the operators $2\text{-opt}^*, 2\text{-opt}, \text{relocate}, \text{swap}, \text{string relocate}, \text{string exchange}$ and $\text{string exchange inverted}$. The latter works like the standard string exchange except the two cutoff path segments are inverted before reinsertion. In each iteration of the local search, the generator arc set, in combination with the neighborhood operators, induces the moves. Only feasible moves are considered and the move which leads to the best improvement of the current solution is applied. This process is repeated until a local optimum is reached with respect to the given operators.

3 Computational Experiment

As a benchmark set, we use parts of the benchmark set of [3], which features 14 different instance sizes ranging between 300 and 2,500 customers. We limit our experiment to the instances with an expected number of 50 customers per route. Ten instances per variant are generated, resulting in 140 instances. All tests are run on an Intel Xeon CPU E5-2687W0 processor at 3.10GHz.

Three different algorithmic settings are used to assess the value of perfect information:

**Complete search (CS):** The search step is evaluated using the complete arc list as generator arcs. The best possible gain $g^*_{ij}$ is saved separately for each neighborhood $j \in \Theta$ and the spend times to explore the complete neighborhoods $t_{ij}^{CS}$ are saved.

**Granular search with perfect information (PI):** From the position of the best move in the complete search, we calculate the optimal thresholds $\pi_{ij}^{PI}$ and use them to sparsify the arc set. The times $t_{ij}^{PI}$ of exploring the neighborhood are saved.

**Granular search with fixed threshold (FT):** We evaluate the step using a granular search with a fixed granularity threshold $\pi_{ij}^{FT} = 0.05 \ \forall j \in \Theta$ and store the time spent on the neighborhood exploration $t_{ij}^{FT}$ and the best move’s gain $g_{ij}^{FT}$ for each $j \in \Theta$.

In each iteration $i$ of the local search, we examine each neighborhood and each setting independently. After collecting all statistics, we apply the best move found. For each
Table 1: Speedup factors and gain ratios for instances with an expected number of 50 customers per route and 300, 800 respectively 1600 customers.

<table>
<thead>
<tr>
<th># customers</th>
<th>300</th>
<th>800</th>
<th>1600</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$s^1$</td>
<td>$s^2$</td>
<td>$r$</td>
</tr>
<tr>
<td>2-opt*</td>
<td>53.38</td>
<td>1.23</td>
<td>1.14</td>
</tr>
<tr>
<td>2-opt</td>
<td>71.21</td>
<td>1.62</td>
<td>1.12</td>
</tr>
<tr>
<td>relocate</td>
<td>45.07</td>
<td>1.95</td>
<td>1.06</td>
</tr>
<tr>
<td>swap</td>
<td>362.36</td>
<td>14.43</td>
<td>1.00</td>
</tr>
<tr>
<td>string exchange inverted</td>
<td>309.44</td>
<td>10.67</td>
<td>1.00</td>
</tr>
<tr>
<td>string exchange</td>
<td>315.65</td>
<td>9.03</td>
<td>1.00</td>
</tr>
<tr>
<td>string relocate</td>
<td>62.95</td>
<td>2.89</td>
<td>1.03</td>
</tr>
<tr>
<td>Average</td>
<td>174.29</td>
<td>5.97</td>
<td>1.05</td>
</tr>
</tbody>
</table>

neighborhood $j \in \Theta$, we calculate the speedup factors $s^1_j = \sum_i \frac{i^c_{ij}}{i^{GS}_{ij}}/n$, $s^2_j = \sum_i \frac{i^{FT}_{ij}}{i^{GS}_{ij}}/n$ and the gain ratio $r_j = \sum_i \frac{g^*_i}{g_{ij}}/n$ with $n$ being the number of iterations. By means of these three indicators, we are able to assess the value of perfect information. Speedup factor $s^1$ shows the possible time savings when finding the best move in each iteration of the local search is desired. Speedup factor $s^2$ and gain ratio $r$ facilitate the comparison of perfect information with a conventional GS approach.

Table 1 shows the respective information for the instances with 300, 800 and 1600 customers and an expected value of 50 customers per route, given as averages over the instances per setting. We observe very promising speedup factors $s^1$ between roughly 38 and 585. The speedup factors grow with the size of the problem instances, which shows that perfect information is very valuable in the main application area of granular searches, namely to solve large-scale instances. The speedup factors differ between the investigated neighborhoods: while swap and the string exchange operators show high values, the 2-opt operator exhibit the smallest speedup. The speedup factors $s^2$ range from 1.26 to 26.01. Averages of at least 5.97 indicate the superiority of perfect information over the conventional GS. The gain ratios are close to 1 for many operators. An exception are the two 2-opt operators especially for instances with fewer customers. This could indicate that good 2-opt moves often need long generator arcs to be effective. Hence, a machine learning approach could be particularly suitable for these operators, since there is a lot to gain in comparison to a conventional GS.
4 Summary and conclusion

In summary, the results verify a high potential for using machine learning techniques to predict optimal granularity thresholds. This inevitably gives rise to several subsequent questions, such as the choice of the right features, the generation of a representative test set and the assessment of the moves. Concerning the last point, predicting granularity thresholds that lead to best gains in each iteration can certainly accelerate the speed of the search but do not necessarily improve or match the quality of the final solution found by conventional granular search with a fixed threshold or by completely searching the neighborhoods.

References


1 Introduction

The vehicle routing problem (VRP) aims to service a set of customers at minimal cost through a collection of routes satisfying various conditions such as capacity, duration, or time windows. It is commonplace to tackle the VRP using metaheuristics. The latter often base their local search operators on the well established lexicographic search paradigm. Sequential search (SS) is an alternative paradigm which performs the exploration in a different fashion. It is introduced in [1] using various well known edge- and string-exchange neighborhood operators. At its root, the idea is to accelerate the core of the search thus reminiscing of the introduction of constant time feasibility checks by [2]. The computational results show that, regardless of the tested operator, the paradigm is of interest and also scales well with the growing number of customers. While lexicographic search is able to prune the search tree based on feasibility considerations, SS uses a cost criterion to discard branches from the exploration. In this respect, both of these methods can pride themselves in providing the best gain at the end of the search.

In contrast, granular search achieves speedups by restricting the search to a promising set of so-called generator arcs but has no such guarantee. Applications such as [3] show the trade off between speed and best gain is warranted. The current state of granular search is more mature than SS on at least two fronts: It is more flexible with respect to the objective function and there already exists rich VRP implementations, see [4].

In this work, we are interested in 1) applying learning mechanisms to forecast threshold bounds to accelerate the search, and 2) extending the realm of application of SS by adapting the theory to the multi-depot VRP. Such is the organization of this abstract.
2 Core principle and perfect information

The threshold criterion is based on a decomposition argument. It states that if a move has an improving gain, at least one of its composing part must be improving regardless of how the decomposition is performed. Figure 1 captures the essence of a 2-opt move affecting two deleted arcs \((i, s_i)\) and \((j, s_j)\) as well as two inserted arcs \((i, j)\) and \((s_i, s_j)\). A snake-shaped link forms a directed path of arbitrary length whereas a straight link indicates a single arc. A solid connection keeps its orientation once the move is completed while a dashed one is inverted in the process. A loosely dotted arc is marked for deletion while a densely dotted one is inserted. The indices 1 and 2 on the depots denote the route number \(r \in \mathbb{R}\) in the current solution \(x\). In particular, the index is not the depot number such that \(d_1 = d_2\) would indicate the same depots are used in both routes. The gain of this move can be computed as \(g := c_{i,s_i} + c_{j,s_j} - c_{ij} - c_{s_i,s_j}\) and it is improving if \(g > 0\).

\[
\begin{align*}
\text{(a) 2-opt (intra).} \\
\text{(b) 2-opt (inter).}
\end{align*}
\]

**Figure 1:** 2-opt neighborhood decomposition.

Breaking down such a move into two partial gains, we obtain that the strict improving inequality \(g > 0\) is fulfilled only if

\[
c_{i,s_i} - c_{ij} > 0 \quad \text{or} \quad c_{j,s_j} - c_{s_i,s_j} > 0
\] (1)

In other words, from each partial gain it is already possible to forecast whether or not a move can be improving. This means that when exploring \(\mathcal{N}(i)\), the outgoing neighbors of \(i\), only those arcs with \(c_{ij} < t^0 := c_{i,s_i}\) are needed, whereas neighbors from \(s_i\) can be limited to those with \(c_{s_i,s_j} < t^0\).

Figure 2 illustrates the behavior of sequential search in the 2-opt neighborhood. The solid arcs form the current solution which includes the two deleted arcs \((c_8, c_{14})\) and \((c_{11}, c_6)\) exchanged for the inserted arcs \((c_8, c_{11})\) and \((c_{14}, c_6)\) for a move with the best gain. Assume the search phase at this point has the arc \((c_8, c_{14})\) up for deletion testing. The next step is to identify a neighboring node to \(c_8\) or \(c_{14}\) that supports an insertion arc ultimately uniquely defining a complete 2-opt move. By preemptively sorting the neighbor sets by increasing cost \(c_{ij}\), sequential search explores the potential moves starting from the closest neighbor node and spiraling outwards until the given threshold is reached.
The three concentric circles depict different thresholds denoted $t^0$, $t^*$, and $t^*_s$ from largest to smallest respectively. Notice that by definition of the circle area $\pi r^2$, the geometrical search space decreases quadratically with every unit reduction of the threshold. The latter two thresholds are reverse engineering the obtention of the move. They are thus treated as perfect information and are the center of attention for the learning phase.

While $t^0$ is certainly large enough by construction of (1) to ensure we always obtain the best gain, let us show now that the same is true for $t^*$ and $t^*_s$. Given a lower bound on the target gain (achievable or not), say $\gamma \in \mathbb{R}$, then analogously to (1), we can state that $g > \gamma$ only if

$$c_{i,s_i} - c_{ij} > \gamma/2 \quad \text{or} \quad c_{j,s_j} - c_{s_i,s_j} > \gamma/2$$  \hspace{1cm} (2)

yielding a threshold value $t^* := c_{i,s_i} - \gamma/2$. By fixing $\gamma^* := g^* - 1$, one creates the tightest threshold value $t^* := c_{i,s_i} - \gamma^*/2$ capable of guaranteeing the identification of a move with the best gain $g^*$. This threshold is conservative by construction of the criterion (2) and its worst case nature. Indeed, the limit case gives insights about a gain which barely meets said criterion. Consider the costs $c_{ij} = t - \epsilon$ and $c_{s_i,s_j} = t$ for the inserted arcs for some positive constant $\epsilon$. Assuming the move is feasible, its gain is $c_{i,s_i} - (t-\epsilon) + c_{j,s_j} - t$ which is greater than $\gamma$ if and only if $c_{j,s_j} \geq c_{i,s_i}$. The worst case to account for therefore happens when the cost distribution surrounding the best gain is almost perfectly symmetric among the deleted and inserted arcs respectively. Observe that one could separate the partial components of the move in any way. In this case, by keeping an equal number of arcs in both components, we aim to keep their respective threshold value minimal simultaneously.

The threshold is validated through a worst case analysis. Since it is highly likely that practical instances do not always behave badly, the threshold can be skewed towards the most interesting part of the decomposition such that pruning is greatly intensified thus reducing search time. This is illustrated in Figure 2 where $t^*$ is the threshold obtained from the best gain and $t^*_s$ is the threshold obtained from the best gain as well as exploiting the asymmetry in the cost distribution.
3 Multi-depot adaptation

Theoretically, the threshold needs to be adapted to the multi-depot environment because some moves need to be repaired to maintain matching depot assignments. Since this repair incurs an additional cost not yet taken under account, the threshold is too small to ensure all promising moves are treated. Figure 3 depicts the different cases that need to be handled for the 2-opt* to ensure sequential search maintains its ability to identify the best gain.

(a) 2-opt* — typical scenario.

(b) 2-opt* — exception scenario 1, \( j = p_2 \) and \( s_j = d_2 \).

(c) 2-opt* — exception scenario 2, \( i = d_1 \) and \( s_i = s_1 \).

(d) 2-opt* — exception scenario 3, \( i = d_1 \) and \( j = d_2 \).

Figure 3: 2-opt* (multi-depot)

In the typical scenario, the arcs affected by the 2-opt* move are distinct from those needed to fix the depots. The total gain of such a move is computed as \( g = [c_{i,s_i} - c_{j,s_j} + c_{p_1,d_1} - c_{p_1,d_2}] + [c_{j,s_j} - c_{i,s_i} + c_{p_2,d_2} - c_{p_2,d_1}] \). As usual, we break down the gain into a conditional expression, say according to the bracketed part, implying that \( c_{j,s_j} < c_{i,s_i} + c_{p_1,d_1} - \min_{d \in T} c_{p_1,d} \), where \( c_{p_1,d_2} \) is replaced by a lower bound over any depot reconnection since we do not know which route completes the move yet. We find this correction quite elegant because it goes in line with intuitive expectations of the multi-depot environment; If the customer \( p_1 \) is already attached to the nearest depot than the right side resolves to the original \( t^0 \) value. Otherwise, the threshold takes into account the potential for depot swapping as the difference between the original depot attachment and the actual nearest one. When considering all scenarios, we obtain the following threshold:

\[
t^0_{MD} := t^0 + \max \{ c_{p_1,d_1} - \min_{d \in T} c_{p_1,d} ; c_{p_1,d_1} - c_{i,d_1} + \max_{r \in \mathcal{R}} (c_{p_r,d_r} - c_{p_1,d_1}) - \gamma / 2 ; c_{d_1,s_1} - \min_{d \in T} c_{d_1,s_1} ; \max_{r \in \mathcal{R}} [(c_{d_r,s_r} - c_{d_1,s_1}) + \max_{j \in \mathcal{P}(r) \setminus j \in \mathcal{N}} c_{j,s_j}] - \gamma / 2 \}.
\]
4 Computational results

Figure 4 takes a look at the 2-opt* operator and shows the potential of exploiting perfect information \( g^* \) and \( g^*_s \) upon launching SS rather than starting from scratch. We see that, as the instance size grows, providing the best gain is not helping so much whereas steering away from the worst case always yields at least 50% speed up.

![Figure 4: Average speedup factor of sequential search when providing \( g^* \) and \( g^*_s \) relative to instance size and average number of customers per route \( f \) for 2-opt*.](image)

5 Summary and conclusion

In summary, the results indicate a high potential for using learning techniques to predict more aggressive pruning thresholds. The multi-depot vehicle routing problem seems to have very little computational impact on sequential search and therefore extends its potential for applicability. The computational results are obtained through a standard local search descent. It is therefore planned to embed this work into a metaheuristic.

References


Column Generation Approach for Feeder Vessel Routing and Synchronization at a Congested Transshipment Port

Jian Gang Jin
School of Naval Architecture, Ocean & Civil Engineering, Shanghai Jiao Tong University, China

Qiang Meng
Department of Civil & Environmental Engineering, National University of Singapore, Singapore

Hai Wang
School of Information Systems, Singapore Management University, Singapore

Email: {jiangang.jin@sjtu.edu.cn; ceemq@nus.edu.sg; haiwang@smu.edu.sg}

1 Introduction

The global container shipping system is commonly structured as the hub-and-spoke network, in which large container vessels (long-haul services) visit transshipment ports (hubs) while feeder vessels connect the hub with neighboring ports. In the literature on container shipping, most of the works focus on hub-and-spoke network design and long-haul service routing and scheduling [1], while the problem of feeder service design is not well investigated [2]. In this paper, we study the feeder vessel routing and synchronization problem (FVRSP) at a congested transshipment port where only fixed time slots are open to feeder vessels. Particular attention is paid to synchronizing the transshipment between long-haul and feeder services as efficient as possible. The key is to adjust the schedules of feeder services with respect to the long-haul service schedules. In doing so, the vessel traffic can be well coordinated and balanced in the temporal dimension, so to avoid schedule conflicts and potential congestions, and improve the transshipment connection between long-haul and feeder services [3]. Another key motivation of this study is to propose a proactive operational strategy for transshipment hub port for the purpose of mitigating the congestion: proactively designating visiting times for feeder services during uncongested periods.
2 Set covering model for the FVRSP

In the FVRSP two decisions for designing feeder services are considered: (1) determining the port-call sequence (i.e., routing) and (2) assigning visiting time slot (i.e., synchronization). The former decision concerns the optimal selection of feeder vessels routes (i.e., port-of-call sequence), vessel type (in terms of capacity) and number of vessels to be deployed for maintaining weekly service, and the amount of containers to be picked up and delivered along each route, with the objective of minimizing the operational cost. The latter decision is to determine the hub port calling time for feeder services within the hub port’s specified times. Note that this is especially effective for those congested transshipment hub ports to reduce the waiting time of feeder vessels. Besides, in order to ensure efficient transshipment connection (i.e., synchronization) between long-haul and feeder services while respecting the vessel traffic demand at the hub port [4], efficient synchronization should be ensured which helps to keep the transit time for container delivery from origin to destination ports at a minimum level, and thus attract more shipment demand (as is illustrated by Figure 1).

![Diagram of feeder services connections](image)

(a) Two connections within current week  
(b) One connection in current week and the other in next week

Figure 1: Two cases of transshipment connection between long-haul and feeder services

We develop a set covering model for the FVRSP, in which each column (i.e. variable) is characterized by compact information related with a feeder vessel, including a sequence of port-of-call, vessel type, number of vessels, cycle time and hub visiting time slot assignment. Let $R$ denote the set of feeder service plans. Define binary decision variable $\delta_r$ to be 1 if service route $r$ is employed, and 0 otherwise. Then, the problem can be formulated
as follows:

\[
\begin{align*}
\text{min} & \quad \sum_{r \in R} (c_{\text{fixed}}^r + c_{\text{variable}}^r + c_{\text{connection}}^r) \delta_r \\
\text{s.t.} & \quad \sum_{r \in R} \lambda_i^r \delta_r = 1 \quad \forall i \in N \\
& \quad \sum_{r \in R} \epsilon_s^r \delta_r = 1 \quad \forall s \in S \\
& \quad \sum_{r \in R} \beta_k^r \delta_r = 1 \quad \forall k \in \Phi \\
& \quad \delta_r \in \{0, 1\} \quad \forall r \in R
\end{align*}
\]

(1) (2) (3) (4) (5)

The objective function (1) minimizes the sum of the fixed and variable costs of employed feeder services as well as the transshipment connection cost. Constraint (2) ensures that each feeder port is served by one feeder route exactly. Constraint (3) guarantees the total number of each type of vessels respect the fleet size. Constraint (4) restricts that each hub calling time slot can only be assigned to at most one feeder service.

### 3 Column generation based approach

Considering of the exponential number of columns in the set covering model, we employ a column generation algorithm to solve the linear relaxation of the model. The pricing sub-problem needs to be activated for identifying feeder service plans (i.e., columns) with negative reduced cost. Let \( \pi_1^i, \pi_2^s, \pi_3^k \) be the dual variables associated with Constraints (2)-(4), respectively. Define decision variable \( x_{ij} \) be 1 if leg \((i, j)\) is served; \( e \) as the required number of vessels; and \( y_k \) be 1 if time slot \( k \) is assigned to the route. Then, the reduced cost corresponding to the feeder service plan is:

\[
\bar{c} = \sum_{(i,j) \in A} (c_{ij} - \pi_1^i) x_{ij} + (c_s - \pi_2^s) e + \sum_{k \in \Phi} (c_k - \pi_3^k) y_k
\]

where \( c_{ij}, c_s \) and \( c_k \) are coefficients associated with link (i.e., voyage) cost, vessel fixed cost and connection cost, respectively. We find the pricing sub-problem is a variant of shortest path problem with negative weights, path dependent cost and time slot assignment related connection cost (corresponding to the above three items). We develop a divide-and-conquer method by enumerating the time slot assignment decision and solving the reduced problem via efficient heuristic method. In order to accelerate the computational efficiency, we further develop an enhanced column generation based heuristic method.

**Column generation scheme (COL0).** The standard column generation procedure is firstly conducted. The master and pricing sub-problems are solved in an interactive
manner until no feeder service plan with negative reduced cost can be found. At this stage, we obtain an initial column set $R_1$.

**Extended column generation scheme (COL1).** At the end of the first step, we can find those saturated (i.e., completely utilized) vessel fleet and time slots by identifying those constraints with negative dual variables. For those saturated fleet and time slots, we reduce the right-hand-side of the corresponding constraints by $m_s/\rho$ and $1/\rho$, respectively. With the updated constraints, the column generation procedure is further activated to expand the column set. Such an extended column generation procedure can be activated for times and finally obtain the column set $R_2$.

**Obtaining integer solutions.** With the column set $R_2$ generated from the above two steps, we impose the binary integer restriction for the decision variables $\delta_r$ and employ a post branch-and-bound procedure to find integer solutions.

4 **Computational tests and remarks**

A case study based on the Southeast Asia container shipping network is conducted indicating that the solution method is applicable for solving real-world size feeder network design problems. The developed method outperforms CPLEX both in solution quality and computational efficiency, while CPLEX fails to generate feasible solutions for instances with 15 ports. Results also demonstrate that container transshipment synchronization at the hub port can be significantly enhanced at the cost of introducing marginal changes to the feeder network configuration.

References


A Lagrangian approach to a cross-docking problem with multiple gates

Manlio Gaudioso, M. Flavia Monaco
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica
Università della Calabria, Rende, Italy

Marcello Sammarra
Istituto di Calcolo e Reti ad Alte Prestazioni - CNR
Rende, Italy
Email: sammarra@icar.cnr.it

1 Introduction

A cross-docking center (CD) is a distribution facility, where goods entering the facility by inbound trucks are unloaded, consolidated with respect to customers’ orders, and delivered by outbound trucks being stored for a very short time or even not stored at all. From a physical point of view, it consists of a number of gates, determining the number of trucks that can be simultaneously processed. Goods unloaded at the inbound gates are transferred to the outbound gates by a conveying system; there they are finally loaded into the outbound trucks. Note that, in both the loading and unloading phases, a truck is not allowed to leave the gate where it is processed until the loading/discharging operation is completed [3].

The management at the operative level of a CD aims, therefore, at optimizing the truck sequencing at the gates. For a complete review of the literature the interested reader can refer to [1], [2], [4]. Here we address the general case where multiple inbound and outbound gates are considered and each inbound (outbound) truck ships (requires) possibly multiple products. The Multi-Gates Multi-Products Cross-Docking problem (MGMPCD) is then to schedule the inbound and outbound trucks at the respective gates, and to decide the amount of each product to be shipped from each inbound truck to each outbound truck, so as to minimize the makespan.

We propose a Lagrangian heuristic and discuss some preliminary computational results obtained by our algorithm on a set of randomly generated test instances, by a comparison with the solutions computed by a benchmark ILP solver.
2 The Lagrangian approach

Let $I$, $O$, $P$, $G^I$, $(|G^I| < |I|)$ and $G^O$, $(|G^O| < |O|)$ be the set of the inbound trucks, outbound trucks, commodities, inbound and outbound gates respectively. We denote by $d_{ip}$, $i \in I, p \in P$ the amount of commodity $p$ delivered by the inbound truck $i$; $r_{jp}$, $j \in J, p \in P$ the amount of commodity $p$ required by the outbound truck $j$. We assume that: a) all the trucks are ready at the time 0; b) they have the same capacity and require the same processing time; d) the time needed for conveying the commodities from the inbound to the outbound gates is negligible; e) $\sum_{i \in I} d_{ip} = \sum_{j \in O} r_{jp} \forall p \in P$.

Thanks to assumptions b) and d), the planning horizon can be discretized into equally length time-slots able to accommodate the processing of a truck. Let $K$, $(|K| = |I|)$ and $L = \{1, \ldots, H\}$, $H \geq |I|+|O|$, be the set of time-slots for the inbound and outbound truck services, respectively. By this way the MGMPCD problem can be viewed as a structured Integer Linear Problem consisting of:

**PI** An assignment phase, involving inbound trucks, gates and time-slots, with decision variables $x_{gik}^i = 1$ if $i \in I$ is discharged at the gate $g \in G^I$ during the time-slot $k \in K$, 0 otherwise.

**PO** An assignment phase, involving outbound trucks, gates and time-slots, with decision variables $y_{gjh}^j = 1$ if $j \in G^O$ is loaded at the gate $g \in G^O$ during the time-slot $h \in L$, 0 otherwise.

**PT** A multicommodity balanced transportation phase (see assumption e), with decision variables $z_{ij}^p \geq 0$ representing quantity of product $p \in P$ shipped from $i \in I$ to $j \in O$.

We look for $x, y, z$ that minimize the makespan $C_{\text{max}}$. These three phases are linked by the following constraints

$$\sum_{g \in G^O} \sum_{h \in L} h y_{gjh}^j \geq \sum_{g \in G^I} \sum_{k \in K} k x_{gik}^i + 1 - n (1 - v_p^i) \quad i \in I, j \in O, p \in P$$

where $v_p^i = 1$ if a non-zero amount of commodity $p$ is shipped from $i$ to $j$, that is if $z_{ij}^p \geq 1$. In this case, the time slot assigned to $j$ must be strictly greater than the time slot assigned to $i$. Dualizing constraints (1) in a Lagrangian fashion with multipliers $\lambda_{ij}^p \geq 0, i \in I, j \in O, p \in P$, the MGMPCD actually separates into three sub-problems, one for each of the previously described phases, no longer dependent on each other, and whose objective functions depend on the $\lambda$’s. By solving **PI, PO, and PT** for each choice of $\lambda$ we get the Lagrangian solutions $x_R, y_R, z_R$, that is $v_R$. If $x_R, y_R, v_R$ satisfy constraints (1), they give a feasible solution $\bar{x} = x_R, \bar{y} = y_R, \bar{z} = z_R, \bar{v} = v_R$ to the MGMPCD, of value

$$\bar{C}_{\text{max}} = \max_{j \in O} \left\{ \sum_{g \in G^O} \sum_{h=1}^H h y_{gjh}^j \right\}$$
In case \( x_R, y_R, v_R \) violate (1), we can recover a feasible solution by:

1. setting \( \bar{x} = x_R, \bar{z} = z_R, \bar{v} = v_R \);

2. sorting the outbound trucks \( j \) assigned to the outbound gate \( g \) with respect to the minimum starting time-slot that makes feasible the corresponding constraint (1), that is

\[
h_{\min} = \max_{k \in K} \left\{ k \sum_{g \in G_0} \bar{x}_{ik} \mid i \in I, \ p \in P, \ \bar{v}_{ij}^p = 1 \right\} + 1.
\]

3. scheduling, for each outbound gate \( g \), the trucks assigned to \( g \) with respect to the sorting previously defined.

Embedding such a recovering procedure within an iterative Subgradient Algorithm, we get our Lagrangian Heuristic (LH), able to generate many feasible solutions to the MGMPCD and to gather the best one among them.

### 3 Computational experience and conclusions

To test our algorithm, we have considered two classes of instances. In the class \( S_1 \), each inbound truck transports a different product, while each outbound truck possibly requests multiple products. In the class \( S_2 \), each outbound truck requires a different product, transported by one or more inbound trucks. For each class we have considered five sets of instances of different dimensions, as shown in Table 1. Finally for each set, we have randomly generated 10 instances obtaining by this way a test set of 100 different instances. In particular, in the \( S_1 \) class for fixed \( i \in I \) the amount \( d_{ip} \) of product \( p \in P \) the truck \( i \) supplies is randomly shared among the outbound trucks. Conversely, in the \( S_2 \) class for fixed \( j \in J \) the amount \( r_{jp} \) of product \( p \) the truck \( j \) requires is randomly supplied by one or more inbound trucks. The LH Algorithm has been coded in C++ and Cplex 12.6 has been used as benchmark ILP solver. The experiments have been run on a machine equipped with a 3.1 GHz CPU and 16GB of RAM, letting Cplex to run for at most 600 seconds and the LH Algorithm to perform 500 iterations. The solution values returned by Cplex and LH are summarized, as averaged values, in Table 1. We report also the average relative error \( ARE = \frac{1}{10} \sum_{i=1}^{10} \frac{(UB^H_i - UB^C_i)}{UB^C_i} \), the number \( N^* \) of optimal solutions discovered by Cplex and the number of times \( N^{BS} \) the LH Algorithm has returned a solution better than or equal to the solution computed by Cplex.

Two facts can be easily retrieved from the entries in Table 1: a) looking at the columns related to Cplex, we note that the class \( S_1 \) seems globally harder to solve than the class \( S_2 \); b) looking at the columns of the LH Algorithm, we observe that, in one case the LH Algorithm has returned solutions that are slightly worse than the solutions returned by Cplex \( (ARE = 0.01) \); in three cases both Cplex and LH have returned the same solutions.
in three cases LH behaves slightly better than Cplex ($ARE = -0.01$); finally, in three cases the gap between LH and Cplex starts to become significant ($ARE \leq -0.03$).

Table 1: Comparison between Cplex and the LH Algorithm.

<table>
<thead>
<tr>
<th>Class</th>
<th>CPLEX</th>
<th>LH Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>I</td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>30</td>
<td>4</td>
</tr>
</tbody>
</table>

From these preliminary computational results, we can say that Lagrangian Relaxation seems to be a viable approach for solving the MGMPCD. Actually, in small amount of time, it allows to search the space of the feasible solutions by a bound-ascent guide, since the updating of Lagrangian multipliers is aimed at determining globally increasing lower bounds, and returns solutions that are mostly comparable or even better that those returned by Cplex.

References


Dynamic Crowd-shipping with Transfers

Alp M. Arslan, Niels Agatz
Rotterdam School of Management, Erasmus University
Email: arslan@rsm.nl

1 Introduction

The rapid growth in the demand for express delivery services in urban areas leads to major challenges for logistic service providers. Customers expect to receive goods, such as food or flowers, within a few hours after they place their orders at minimal additonal cost. At the same time, it is very challenging to cost-efficiently organize same-day deliveries as the short lead-time makes it difficult to consolidate deliveries. [2] show that crowd-shipping can be a viable solution to overcome some of the challenges of same-day delivery.

Crowd-shipping aims to utilize the existing traffic flows to support parcel deliveries. In this concept, an online crowd-shipping platform brings together delivery tasks (parcels) and ad-hoc drivers who are willing to deliver parcels on their way to their original destinations. These platforms often match parcels with suitable drivers to maximize the parcel deliveries and minimize the system-wide total delivery costs.

In earlier works on crowdsourced delivery, [1] and [2] study a setting in which a parcel can be carried by a single driver or a dedicated vehicle, and transferring a parcel between drivers is not possible. [3] look into a crowd-shipping problem with time windows and transfers in a static long-haul transportation environment. Passenger transfers between rides in ride-sharing has been studied by [4]. They propose a decomposition based heuristic to cope with the dynamic multi-hop problem. Their findings also show that the possibility of transfers increases the number of served riders significantly.

In this study, we examine the benefits of allowing transfers in a peer to peer (P2P) same-day delivery platform as described in [2]. We study a real-time setting in which delivery tasks and ad-hoc drivers arrives dynamically. Also, we proposed a solution approach that is capable of responding the frequent announcement arrivals in short time. Furthermore, we assess the benefits of possible transfers with extensive numerical experiments.

2 Problem Definition

We consider a same-day delivery platform that receives delivery tasks and ad-hoc drivers dynamically within a service period. Let $P$ be a set of delivery tasks/parcels and $D$ be
the set of ad-hoc drivers that arrive over the service period. We consider a set of stores from which all parcels are supplied, denoted by set $S$. We also assume that each driver originates her trip from one of the pickup stores. Each task $i$ (driver $k$) has a unique destination $d(i)(d(k))$. We assume that task $i$ can only be supplied from a unique pickup store. For convenience, let $L$ be a set that consists of all locations including all pickup stores, the destinations of parcels and the destinations of the drivers. The travel distance and the travel time between two points, $a, b \in L$, denoted by $d_{ab}$, $t_{ab}$, respectively; both follow the triangle inequality.

A parcel $i$ or driver $k$ arrives at $a_{i(k)}$ within the delivery service time $[0, T]$. After a fixed order preparation time, the parcel is ready to be shipped at $e_i := a_i + p$. The parcel has to be delivered before its latest arrival time $l_i := a_i + l$, where $l$ denote a delivery lead time. Similarly, a driver can depart at $e_k := a_k + u$ after a positive announcement lead time $u$ and has to be at her destination at $l_k := e_k + T_k + t_{o(k),d(k)}$. $T_k$ denotes the departure time flexibility of driver $k$.

The platform has two options to serve a delivery task:

- A **direct delivery** is associated with a pair $(i, k)$ of task $i \in P$ and driver $k \in D$ in which driver $k$ picks task $i$ up and delivers it to the destination,

- A **delivery with transfers** is associated with a collection $(i, \{k_1, k_2, \cdots \}, \{s_1, \cdots \})$ of task $i \in P$, drivers $\{k_1, k_2, \cdots \} \subset D$ and transfer stores $\{s_1, \cdots \} \subset S$. As an example of a delivery with a single transfer, $(i, \{k_1, k_2\}, \{s\})$; driver $k_1$ picks task $i$ up and brings it to transfer point $s$ and driver $k_2$ delivers it to the destination of task $i$, $d(i)$.

The platform aims to serve as many delivery requests as possible. Furthermore, the platform incurs a distance-based penalty for each unserved task, which reflects alternative delivery cost, and it gives a detour-based compensation to the drivers who make deliveries. Thus, the minimization of the delivery cost is the secondary objective.

To solve the problem described above, we initially make the following assumptions.

- Drivers can carry at most one parcel,

- Transfers are restricted to the pickup stores,

- A parcel is not allowed to transfer more than once, which also means at most two drivers can carry the same parcel,

- A driver can pick a parcel up only if the parcel’s origin is the same with the driver.

These modelling assumptions can be considered as the base setting. We will relax some of them to investigate the impact on the solution approach and the results.
3 Solution Approach

To deal with the dynamic arrivals of announcements, we use an event-based rolling horizon approach which re-optimizes a snapshot problem after an arrival of a new driver or task.

3.1 Matching Formulation

At a decision epoch $t$, a snapshot problem can be formed with the corresponding active sets of parcels $P_t$, drivers $D_t$. Let set $J_t$ be the set of jobs (feasible deliveries) at time $t$, either direct or with transfers. Let $x_j, j \in J_t$ be binary decision variable that job $j$ is in the solution or not. Let $J_k$ be the set of jobs $k$ is in, and $J_i$ be the sets of jobs that parcel $i$ is in. Then, the problem can be formulated as follows:

$$\max z = \sum_{j \in J_t} (M - c_j) x_j$$  \hspace{1cm} (1)

s.t \hspace{1cm} \sum_{j \in J_i} x_j \leq 1 \hspace{1cm} \forall i \in P_t, \hspace{1cm} (2)

\sum_{j \in J_k} x_j \leq 1 \hspace{1cm} \forall k \in D_t, \hspace{1cm} (3)

x_j \in \{0,1\}.

where $M$ is a big number. Equation 1 is the objective function that maximizes the number of matched parcels and minimizes the total cost. Constraints 2(3) ensures that each parcel(driver) is served(used) at most one.

3.2 Determining the feasible jobs

To form the above mathematical problem given in 1 - 3, the set of feasible jobs has to be generated. A job $j$ represents a feasible delivery, a collection of a parcel, a driver or drivers if the job includes a transfer. A job is considered as feasible job if the following conditions are satisfied.

- **Time-feasibility**: The associated driver(s) and parcel of the job have overlapping timelines.

- **Detour-feasibility**: The corresponding detour(s) for the parcel delivery should be less than or equal to each driver’s detour flexibility.

- **Synchronization-feasibility**: (Only applicable for the jobs including transfers) The driver who picks the parcel up should deliver it the next driver’s origin before her departure flexibility is over.

4 Preliminary Results and Discussion

To quantify the benefits, we run several numerical experiments. We generate instances on a 15x15 km square shaped plane. Four pickup stores are located on the corners of a 5x5
square which is positioned at the centre of the plane. The origins of tasks and drivers are randomly chosen among the pickup stores and the destinations are uniformly distributed on the plane. The arrival time of task/driver announcement is chosen uniformly within an eight hour time span.

Table 1 shows the results for the hindsight cases, which assume that all information for the delivery period is given, averaged on ten replications per instance type. \( n_p : n_d \) represents the number of delivery tasks and drivers, respectively. We set the delivery lead time \( l \) to 60 minutes and the departure time flexibility \( T_k \) for each driver \( k \in D \) to 20 minutes.

<table>
<thead>
<tr>
<th>( n_p : n_d )</th>
<th>No Transfer</th>
<th>With Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost (%)</td>
<td>Matched Task (%)</td>
</tr>
<tr>
<td>50:50</td>
<td>100</td>
<td>58.0</td>
</tr>
<tr>
<td>100:100</td>
<td>100</td>
<td>73.1</td>
</tr>
<tr>
<td>50:100</td>
<td>100</td>
<td>85.3</td>
</tr>
<tr>
<td>100:50</td>
<td>100</td>
<td>42.8</td>
</tr>
<tr>
<td>Avg.</td>
<td>100</td>
<td>64.8</td>
</tr>
</tbody>
</table>

The initial results indicate that with the possibility of parcel transfers, the number of tasks matched served by drivers increases up to 4.4% compared to the case without transfers. Furthermore, the total cost decreases by 13.2% on average.

We are currently working on the real time implementations. Also, we plan to extend the current model by relaxing some of the model assumptions. For example, we plan to examine the impact of increasing the number of tasks that a driver can carry.

References


1 Introduction and Problem Definition

The distribution planning problem with consolidation addresses the coordination of distribution activities between a set of suppliers and a set of customers, through the use of intermediate facilities in order to achieve savings in transportation cost. We study the problem from the perspective of a third-party logistics provider (3PL) that is coordinating shipments between suppliers and customers. Given customer demand of products from different suppliers, the goal is to consolidate the shipments in fewer high volume loads, from suppliers to the consolidation center(s) and from the consolidation center(s) to customer. We assume that suppliers have a finite set of transportation options, each with a given capacity and time of arrival at the consolidation center(s). Similarly, customers have a set of transportation options, each with a given capacity and dispatch time from the consolidation center(s). The 3PL wants to determine the optimal transportation options, or shipment schedule, and the allocation of shipments to transportation options from suppliers to consolidation center(s), and from consolidation center(s) to customers, that minimize the total transportation cost and holding cost at the consolidation center.

The literature contains many variations of this problem, which assume deterministic demand. The present paper extends the problem for stochastic demand and formulates it as a two-stage stochastic programming model. We model the case where the choice of transportation options is a strategic or contractual decision, and a 3PL needs to decide on which options to reserve for a given planning period, subject to stochastic customer demand. Therefore, the choices of transportation options are the stage one variables in the two-stage stochastic program. The second stage variables, which are decisions that are made after the uncertainty conditions become known, represent the allocation of orders to reserved transportation options, as well as shipping orders through a spot-market carrier, when necessary, at a greater transportation cost.
2 Problem Formulation

To formulate the problem, we first define sets $I$, $J$ and $K$ to represent the sets of suppliers, customers and products, respectively. $I(j)$ denotes the set of suppliers from which customer $j$ orders. $J(i)$ indicates the set of customers that order from supplier $i$, and $K(ij)$ represents the set of products that are ordered by customer $j$ and supplied by supplier $i$. We also define $Q$ and $L$ as the sets of transportation options for a given supplier $i$, and a given customer $j$, respectively, and $S$ as the set of scenarios. We formulate the problem as follows:

$$\min \sum_{i \in I} \sum_{q \in Q} f(x_{iq}) + \sum_{j \in J} \sum_{l \in L} g(y_{jl})$$

$$+ \sum_{s \in S} \sum_{i \in I} \sum_{j \in J(i)} \sum_{k \in K(ij)} p^s [d_{ijkl}^s h_k(\sum_{l \in L} t_{jil}(w_{ijkl}^s + \lambda_{ijkl}^s)) - \sum_{q \in Q} t_{iq}(u_{ijqk}^s + \mu_{ijqk}^s)]$$

$$+ \sum_{s \in S} \sum_{i \in I} \sum_{q \in Q} \sum_{j \in J(i)} \sum_{k \in K(ij)} p^s \pi d_{ijkl}^s (\mu_{ijqk}^s) + \sum_{s \in S} \sum_{j \in J} \sum_{l \in L} \sum_{i \in I} \sum_{k \in K(ij)} p^s \pi d_{ijkl}^s (\lambda_{ijkl}^s)$$

$$\text{s.t.} \quad \sum_{q \in Q} (u_{ijqk}^s + \mu_{ijqk}^s) = 1, \quad i \in I, j \in J(i), k \in K(ij), s \in S \quad (2)$$

$$\sum_{l \in L} (w_{ijkl}^s + \lambda_{ijkl}^s) = 1, \quad j \in J, i \in I(j), k \in K(ij), s \in S \quad (3)$$

$$\sum_{q \in Q} t_{iq}(u_{ijqk}^s + \mu_{ijqk}^s) \leq \sum_{l \in L} t_{jil}(w_{ijkl}^s + \lambda_{ijkl}^s), \quad j \in J, i \in I(j), k \in K(ij) \quad (4)$$

$$\sum_{j \in J(i)} \sum_{k \in K(ij)} d_{ijkl}^s u_{ijqk}^s \leq C_{iq} x_{iq}, \quad i \in I, q \in Q, s \in S \quad (5)$$

$$\sum_{i \in I(j)} \sum_{k \in K(ij)} d_{ijkl}^s w_{ijkl}^s \leq C_{jl} y_{jl}, \quad j \in J, l \in L, s \in S \quad (6)$$

$$x_{iq}, y_{jl} \in \{0, 1\}, \quad i \in I, q \in Q, j \in J, l \in L$$

$$u_{ijqk}^s, \mu_{ijqk}^s \in \{0, 1\}, \quad i \in I, q \in Q, j \in J(i), k \in K(ij), s \in S$$

$$w_{ijkl}^s, \lambda_{ijkl}^s \in \{0, 1\}, \quad j \in J, i \in I(j), l \in L, k \in K(ij), s \in S \quad (7)$$

where $x_{iq}$ indicates whether transportation option $q$ for supplier $i$ is chosen, and $C_{iq}$ is the associated capacity of that option. Similarly, $y_{jl}$ represents whether transportation option $l$ is chosen for customer $j$, and $C_{jl}$ is its associated capacity. $u_{ijqk}^s$ indicates the allocation of commodity $k$ for supplier-customer pair $(i, j)$ in scenario $s$ to a given available transportation option $q$, while $\mu_{ijqk}^s$ represents the allocation of a given commodity $k$ to a spot market carrier that leaves the consolidation center at the same time as option $q$. Similarly, $w_{ijkl}^s$ indicates the allocation of commodity $k$ for supplier-customer pair $(i, j)$ in scenario $s$ to a given available transportation option $l$, while $\lambda_{ijkl}^s$ represents the allocation of commodity $k$ to a spot market carrier that leaves the consolidation center at the same
time as option \( l \). Additionally, \( p^s \) denotes the probability of a given scenario \( s \), \( d^s_{ijk} \) is the demand of product \( k \) for supplier-customer pair \((i, j)\) in scenario \( s \), and \( h_k \) is the holding cost of commodity \( k \) at the consolidation center. Finally, \( t_{iq} \) and \( t_{jl} \) are the arrival time of transportation option \( q \) at the consolidation center for a certain supplier \( i \), and the departure time from the consolidation center of transportation option \( l \) for a given customer \( j \), respectively.

The objective function (1) minimizes the total transportation cost of the chosen transportation options for suppliers and customers, \( f(x_{iq}) \) and \( g(y_{jl}) \), respectively, the expected holding cost at the consolidation center, and the expected cost of shipping through a spot market carrier. Constraints (2) and (3) ensure that the model allocates each product to exactly one inbound shipment, and exactly one outbound shipment, respectively, whether the shipment is through a reserved first-stage transportation option, or a spot market carrier. Constraint (4) makes sure that the time of departure of a given product from the consolidation center is greater than the time that product arrives at the consolidation center. Constraints (5) and (6) ensure that the total demand allocated to a reserved transportation option, for given supplier and customer, respectively, does not exceed the capacity of that option.

### 3 Solution Methodology

Because of the high computational demand of the model, the integer L-shaped method is applied to decompose the problem. To increase the efficiency of the algorithm, we create two valid cuts with the goal of generating stronger cuts than the L-cut. We also apply three algorithm enhancement techniques to speed up the convergence of the algorithm, namely, partial decomposition [1], the alternating cut strategy [2], and single tree search for the algorithm master problem (MP) with a callback routine.

#### 3.1 Valid Cuts

The integer L-shaped cuts by Laporte and Louveaux [3], which can be expressed for our problem as shown below in Equation (8), are general cuts that can be applied to any stochastic program when the first stage variables are all binary. Those cuts, however, may be weak as they set the projected cost of a subproblem \( (\theta^s) \) to the lower bound of the subproblem \( (L^s) \) or to a lower value, if there is a change in MP solution \((\bar{x}_{iq}, \bar{y}_{jl})\). If there is no change in MP solution, \( \theta^s \) takes the value of the subproblem objective function value \( z^s \). We propose two valid cuts with the goal of better approximating the change in subproblem objective function value when the MP solution changes.

\[
\theta^s \geq (L^s - z^s) \left( \sum_{q \in \tilde{Q} \setminus Q} \bar{x}_{iq} + \sum_{l \in \tilde{L} \setminus L} \bar{y}_{jl} \right) + z^s, \quad \forall s \in S
\]  

(8)
Valid Cut 1 can be expressed as shown in Equation (9):

\[ \theta^s \geq \sum_{q \in Q \setminus \tilde{Q}} \delta^s_{iq} x_{iq} + \sum_{l \in L \setminus \tilde{L}} \delta^s_{jl} y_{jl} + z^s, \ \forall s \in S \]

(9)

where \( \tilde{Q} \) and \( \tilde{L} \) respectively are the sets of supplier and customer transportation options, which have a value of 1 in a given MP solution. The goal of Valid Cut 1 is to tightly overestimate the maximum impact of the addition of a transportation option to the subproblem objective function value.

We compute this maximum improvement \( \delta^s_{iq} \) and \( \delta^s_{jl} \) heuristically to avoid resolving the subproblem. However, for the heuristic method used, estimating the impact of adding the unreserved transportation options in a single cut affects the strength of the cut. Therefore, we propose Valid Cut 2, expressed in Equation (10), which estimates the improvement of the subproblem with the addition of one transportation option at a time. If a given \( x_{iq} \) with \( q \notin \tilde{Q} \) gets a value of 1 in the MP solution \( \tilde{x}_{iq} \) at a subsequent iteration, and no other changes take place, the cut sets \( \theta^s \) to the value of \( z^s + \delta^s_{iq} \). Otherwise, \( \theta^s \) takes the value of the subproblem lower bound or a lower value, similar to the L-shaped cut.

\[ \theta^s \geq \delta^s_{iq} x_{iq1} + (L^s - z^s) \left[ \sum_{q \in Q \setminus (\tilde{Q} \cup \{q^1\})} x_{iq} + \sum_{l \in L \setminus \tilde{L}} y_{jl} \right] + z^s, \ \forall q \in Q \setminus \tilde{Q}, \forall s \in S \]

\[ \theta^s \geq \delta^s_{jl} y_{jl1} + (L^s - z^s) \left[ \sum_{q \in Q} x_{iq} + \sum_{l \in L \setminus (\tilde{L} \cup \{l^1\})} y_{jl} \right] + z^s, \ \forall l \in L \setminus \tilde{L}, \forall s \in S \]

(10)

4 Results and Conclusion

Numerical results show that the performance of our proposed methodology and valid cuts is comparable to that of CPLEX, and is better for some instances. For the tested instances, the values of \( \delta \)'s in the valid cuts are computed at every iteration, which limits the number of iterations the algorithm completes in a set time limit, and consequently, affects the algorithm's overall computational efficiency. We suggest promising areas of future research, relating in particular to other possible methods of estimating \( \delta \)'s in the valid cuts, to further improve the computational efficiency of the algorithm.

References


1 Introduction

In recent years, the study of more realistic and involved variants than the classical Capacitated Vehicle Routing Problem (CVRP) is attracting an increasing academic attention. This growing body of literature is stimulated by two main reasons. On the one side, by the desire of bridging the gap between academic problems and real-world applications. On the other side, by the recent advances in optimization methods and computer capabilities that are making it possible to jointly solve strongly interdependent problems that have been, until recently, addressed independently. *Integrated vehicle routing problems* is the term increasingly used to denote a class of problems where the Vehicle Routing Problem (VRP) arises in combination with other optimization problems within the broader context of logistics and transportation. Some examples of problems in this class are location-routing problems; production-routing problems; inventory-routing problems; multi-echelon routing problems; and routing problems with loading constraints where the routing of vehicles and the loading of goods onto them are simultaneously optimized (see the survey by Iori and Martello [3]). A remarkable number of surveys, as well as special issues in international journals that appeared in the literature, including the one edited by Bektaş et al. [1], testify the increasing attention that this research area is attracting among academics.

As integrated vehicle routing problems combine optimization problems that are usually $\mathcal{NP}$-hard by themselves, the prevailing attitude among operations researchers has been, until recently, to tackle each problem independently, at the expense of global optimization. In fact, solving each problem, even by means of an exact method, and then combining
the partial solutions obtained, typically leads to a sub-optimal solution for the integrated problem. On the other side, combining two, or more, hard problems causes a significant increase of the computational burden required, but tends to provide considerably better solutions than solving optimally each problem, often even if the integrated problem is solved with a heuristic. It is important to motivate an integrated approach, quantifying the magnitude of the benefits that can be achieved addressing the integrated problem directly instead of tackling each problem independently. To the best of our knowledge, only few papers appeared in the literature along this line of research.

The VRP is one of the most important and investigated class of combinatorial optimization problems. In the CVRP, which is the simplest and most studied member of the class of VRPs, the demand of each customer is expressed by a single value, often representing the total weight of the items to be transported. Thus, a solution is feasible for the CVRP if the sum of the demands of the customers assigned to each vehicle does not exceed its capacity. Nevertheless, in many real-world freight transportation applications it cannot be neglected that the items are characterized not only by a weight but also by a shape. In such situations, a solution that is feasible for the CVRP may prove to be infeasible in practice, since it is impossible to determine a feasible loading pattern to allocate all the items within the loading area of the vehicles. These loading issues are closely related to multi-dimensional packing problems, especially extensions of the classical (one dimensional) Bin Packing Problem (BPP). Several operational restrictions often complicate the problem further.

2 The optimization problem

In this talk, we consider the CVRP with Two-dimensional Loading constraints, henceforth referred to as 2L-CVRP, that is a variant of the CVRP where rectangular-shaped items have to be transported and loading constraints have to be satisfied. The assumption that characterizes the 2L-CVRP is that the items cannot be stacked on top of each other. The 2L-CVRP models applications concerning the transportation of heavy or fragile items, such as furniture and household appliances, or pieces of catering equipment, such as food trolleys, or when customer orders are loaded onto pallets which cannot be stacked on top of each other (for more examples, see Wang et al. [5]). In the 2L-CVRP, the demand of each customer is composed of a set of rectangular-shaped items such that, for each item, its weight, shape and orientation are given. A fleet of identical vehicles based at a single depot is available to deliver these items. The vehicles have a given weight capacity and a rectangular loading area that can be accessed only from one side. The 2L-CVRP calls for the determination of a minimum-cost set of routes to be traveled by the given fleet of vehicles to serve the customers, subject to the following set of constraints: (a)
Weight capacity constraints: the total weight of the items loaded onto a vehicle cannot exceed its capacity; (b) Classical BPP constraints: there must exist a non-overlapping loading pattern of all the items into the loading area of the vehicles; (c) Item clustering constraints: all the items of a given customer must be assigned to the same vehicle; (d) Item orientation constraints: each item has a fixed orientation and cannot be rotated; (e) Orthogonality constraints: each item has to be loaded with its edges parallel to those of the vehicle; (f) LIFO constraints: items of the current customer must be directly available by pulling them out from the rear doors without moving any item of other customers.

The 2L-CVRP has been introduced by Iori et al. [4] along with the first exact algorithm for its solution.

3 Goals and contributions

Generally speaking, we refer to an integrated approach when the integrated problem is tackled directly as a single and unique problem. We refer to a not integrated approach when the sub-problems that make up the integrated problem are addressed separately, often one after the other as the solutions are related to each other. The general idea of a not integrated solution approach for the 2L-CVRP is to address separately, at least to a certain extent, the routing and the loading sub-problems instead of tackling the problem as a whole. A variety of Not Integrated Solution Approaches can be designed, where the approaches differ for the type and degree of separation of the sub-problems.

We describe a mathematical formulation for the 2L-CVRP, and propose a solution approach that solves this integrated problem with an exact algorithm. The latter solution method is based on a Branch-and-Cut algorithm that iteratively checks if the solution found at each node of the branching tree violates the loading constraints. We compare such an approach with three not integrated approaches that consider the routing and the loading aspects of the problem separately. In more details, we propose three not integrated approaches for the solution of the 2L-CVRP based on solving separately the routing and the loading sub-problems. The first approach is based on a neat separation between the solution of the two sub-problems, the second tries to mimic the possible behavior of a logistic operator, whereas the third is more sophisticated and based on the solution of a routing problem with profits. The main goal of this talk is to provide evidence of the importance of an integrated approach for the 2L-CVRP, studying the worst-case performance of the not integrated approaches, as well as reporting on extensive computational experiments conducted on benchmark and new instances.
4 Conclusions

We show, through a worst case analysis, that the cost of a solution obtained with a not integrated approach may be as large as twice the cost of an optimal integrated solution. We also show empirically, through extensive computational experiments, the importance of the integration. The computational results indicate that the integrated problem provides better solutions, both in terms of total cost, number of vehicles routed and load factors achieved. In particular, on the instances where an optimal solution is found for the integrated approach, the average cost increase of the three not integrated approaches is 6.98%, 7.55% and 6.12%, respectively, for the benchmark instances, whereas it is 8.17%, 8.61% and 6.62% for the new test instances. The results obtained suggest that it is worthwhile to jointly tackle strongly interdependent problems that have been, until recently, addressed separately and that this is true even in the cases the integrated problems cannot be solved exactly, provided the error generated by a heuristic remains smaller than the cost increase of a not integrated approach. The line of research proposed in this research can be extended to other integrated problems to motivate their scientific study and to evaluate the potential benefit of their implementation in practice.

This talk is based on the paper by Côté et al. [2].

References


Comparing Sequential and Integrated Approaches for the Production Routing Problem

Nabil Absi
Department of Manufacturing Sciences and Logistics,
Ecole des Mines de Saint-Etienne, Gardanne, France

Claudia Archetti
Department of Economics and Management,
University of Brescia, Italy

Stéphane Dauzère-Pérès, Dominique Feillet
Department of Manufacturing Sciences and Logistics,
Ecole des Mines de Saint-Etienne, Gardanne, France

M.Grazia Speranza
Department of Economics and Management,
University of Brescia, Italy
Email: grazia.speranza@unibs.it

1 The Production Routing Problem

Traditionally, optimization problems in logistics and supply chain management have separately considered location decisions, inventory management, production and distribution decisions. In the last decade, the amount of literature that studies complex optimization problems, that jointly optimize interdependent problems, has increased substantially. This trend is made possible thanks to the advance in optimization methods and computer capabilities, and is motivated by the potential benefits that can be achieved. These problems are often called integrated problems and are usually computationally much more challenging than the sequential solution of simpler problems. While it is obvious that integrated problems, if optimally solved, provide better solutions, most integrated problems can only be solved heuristically. Thus, the issue of whether and when it is beneficial to tackle an integrated problem becomes relevant and deserves to be investigated. To the best of our knowledge, Chandra and Fischer [7] were the first to study the value of coordinating production and distribution decisions. A few more recent papers have studied the benefits of optimizing integrated problems.
One of the most complex integrated problems arising in logistics jointly optimizes production planning, inventory management and distribution planning decisions. The Production Routing Problem we consider was first introduced in [6]. Given the complexity of the problem, most researchers have focused on the development of heuristic methods while the literature on exact methods is more restricted. A recent survey on formulations and solution algorithms for the Production Routing Problem can be found in [3].

In this paper we consider the Production Routing Problem studied in [5], and generalized in [1] with the introduction of additional constraints at the factory. A factory produces one commodity which is distributed to retailers by a vehicle over a discrete time horizon to satisfy the demands of retailers. Inventory capacity at the factory and at the retailers must be taken into account and stock-outs are not allowed. The production phase generates fixed production set-up costs and variable costs. At the factory and at the retailers, inventory holding costs are charged, proportional to the inventory levels. The cost of the distribution is proportional to the distance traveled by the vehicle. The production plan, i.e. when and how much to produce, and the distribution plan, i.e. when and how (to which retailers and the routes of the vehicle) to distribute, have to be jointly identified in such a way that the total production, inventory and transportation cost is minimized. The so called Maximum Level (ML) policy is used for the inventory management. In the ML policy, a maximum inventory capacity is set at the retailers. Any quantity may be delivered as long as the capacity limits are respected. With respect to the Production Routing Problem introduced in [5] we consider only one vehicle with unlimited capacity. In [5], an exact algorithm was proposed for the solution of this Production Routing Problem in the case of one vehicle and a matheuristic for the case of multiple vehicles. A large neighborhood search algorithm was proposed in [2], whereas the state-of-the-art heuristic for the Production Routing Problem is the two-phase iterative method proposed in [1], that iteratively focuses on production and distribution decisions.

2 Objectives

The goal of this paper is to investigate the benefits of solving the integrated Production Routing Problem with respect to sequentially solving simpler problems. Although it is obvious that integrating decisions allows for better solutions, quantifying the improvements is extremely important for two main reasons. The first is that an integrated approach implies a higher degree of organizational and computational complexity that should be justified, the second is that integrated problems are very unlikely to be solved by optimal methods and, thus, heuristics may vanish the advantages of an integrated approach.

We compare the integrated approach, where production and distribution decisions are simultaneously optimized, with two sequential approaches:
1. in the first approach, denoted ProduceFirst, production decisions are optimized first and then distribution decisions are optimized based on the fixed production plan;

2. in the second approach, denoted DistributeFirst, distribution decisions are optimized first and then production decisions are optimized based on the fixed distribution plan.

In the sequential approaches, the production problem is a lot sizing problem while the distribution problem is an Inventory Routing Problem, both well studied problems.

We identify the most appropriate solution algorithms for the Production Routing Problem and for the subproblems of the sequential approaches and we compare them on a set of benchmark instances obtained from instances proposed for the IRP, and generated by making use of two parameters that allow us to control the trade-off between production and distribution cost, and the average number of time periods between consecutive production setups. The computational results are aimed at showing the impact of the integrated approach, and also to compare the two different sequential approaches, in dependence of the structure of the instance solved.

3 Solution approaches and results

The IRP solved in the distribution problem turns out to be the problem introduced in [4] for the single-vehicle case. Since the Production Routing Problem cannot be solved to optimality even on small size instances, it is solved heuristically with an algorithm which is a variant of the one proposed in [1]. As we investigate the benefits of global optimization, it is essential to optimally solve the two problems in the sequential approaches. While obtaining the optimal solution of the lot sizing problem is not computationally challenging for the problem complexity and for the size of instances considered, computing optimal solutions for the IRP is much more challenging. This is why we consider only one vehicle per time unit. This allows us to optimally solve many instances of the IRP and the lot sizing problem with a standard solver.

The solutions obtained with the different approaches are analyzed and some properties derived. Computational experiments are performed on instances of different size which are generated using two critical parameters. The first parameter, called Production-Distribution Ratio (PDR), allows us to control the trade-off between production cost and distribution cost, whereas the second parameter, called Setup cost-Holding cost Ratio (SHR), is well-known in the lot sizing literature and is used to define a Time-Between-Orders (TBO), i.e. an average number of time periods between consecutive setups. The solution of the integrated approach is compared to the solutions obtained by the two sequential approaches DistributeFirst and ProduceFirst on instances where the subproblems of the sequential approaches can be solved to optimality. In this case the gaps between solutions of the sequential approaches and of the integrated one underestimate the savings.
that can be achieved. The numerical results illustrate the properties derived and show that the benefits of the integrated approach over the two sequential ones depend on the trade-off between production and distribution costs and on the trade-off between setup and inventory costs in production. The frequently large size of the gaps (often two digit gaps) shows that the decision about whether and when to adopt an integrated approach may have a huge economic impact.

References


Branch-and-price algorithms for bi-objective vehicle routing problems

Nicolas Jozefowiez
LCOMS, Université de Lorraine, Metz, France,

Sandra Ulrich Ngueveu
CNRS, LAAS, INP Toulouse, Toulouse, France,

Estèle Glize
CNRS, LAAS, INSA, Toulouse, France
Email: estele.glize@laas.fr

1 Context

This paper presents an exact method for bi-objective Vehicle Routing Problems (BOVRPs) where arcs are associated with two costs. These two objectives can be conflictive: in motor vehicle, the travel time differs from the distance. Many industrials are interested in finding a good compromise.

Multi-objective VRPs (MOVRPs) are more and more studied. A complete survey of MOVRP can be found in Jozefowiez et al. [1]. In addition to the minimization of travel distance, most MOVRPs aim to minimize the number of vehicles or to maximize the fairness of routes. Green VRPs modeled as MOVRPs also minimize the fuel consumption or the gas emission. So, these problem present several costs on each arc but are different from the problem presented in this paper because pollution costs are function of multiple variables, not only the arcs. They are solved via heuristics [2, 3].

In the larger scope of multi-objective integer programming, exact methods are divided into two classes: methods working on the feasible solutions space [4] and those working on the objective functions space [5]. These last methods solve a sequence of mono-objective problems and so, rely on the efficiency of single-objective integer programming solvers. The ε-constraint method is the most commonly used objective space search algorithm [6].

The main purpose of this work is to propose a competitive method for finding the complete and exact Pareto front of BOVRP where the objectives are to minimize the two total costs of the journey.
2 Methods

We propose an objective space search algorithm to solve the BOVRP based on an Baldacci et al’s method [7] that efficiently solves the mono-objective VRP to optimality. It generates all routes whose reduced cost is contained between a lower bound \(LB\) and an upper bound \(UB\). This algorithm will be referred to as \(GENROUTE(UB, LB)\) and produces a reduced set of routes \(\Omega\). The integer problem is finally solved on \(\Omega\) by an integer solver.

2.1 Reference method

This method is called the reference method as it is the more direct way to use the Baldacci et al. method in a BOVRP within an \(\epsilon\)-constraint method. The \(\epsilon\)-constraint approach minimizes the first objective \(c^1\) under the constraint that the second objective \(c^2\) has to be lower than a certain value \(\epsilon\). The reference method is summarized in Algorithm 1.

Algorithm 1 Algorithm of the reference method

\[
\begin{align*}
\epsilon & \leftarrow +\infty \\
\textbf{while } & \exists \text{ a solution } \textbf{do} \\
& \text{Solve the linear relaxation of the problem for } \epsilon \text{ to obtain } LB \\
& \text{Find a feasible solution } UB \text{ of the integer problem for } \epsilon \\
& \Omega \leftarrow GENROUTE(UB, LB) \\
& \text{Solve the integer problem on } \Omega \text{ to obtain } S_{OPT} \\
& \text{Set } \epsilon \leftarrow S_{OPT}^2 - 1 \\
\textbf{end while}
\end{align*}
\]

2.2 Two-steps method

This second method minimizes the weighted-sum of the two objectives \(\lambda c^1 + (1 - \lambda)c^2\), \(\lambda \in [0, 1]\) [8]. We also need to introduce the call of Baldacci et al. method for the weight \(\lambda\): \(GENROUTE(UB, LB, \lambda)\), where the cost of a solution is computed with respect to \(\lambda\).

The algorithm is divided into two steps. First, the supported points are computed in a dichotomous approach and then, the non-supported points are found in areas not explored yet (Algorithm 2).

The first step needs as input the optimal solutions \(S_1\) and \(S_2\) that minimizes the costs \(c^1\) and \(c^2\) respectively. The gradient \(\lambda_1\) of these two points is computed. Then, the optimal solution \(LB_1\) of the linear relaxation of the parametrized formulation for \(\lambda_1\) is computed. Finally, we apply the algorithm \(GENROUTE(LB_1, UB_1 = S_1, \lambda_1)\) to have a reduced set of routes \(\Omega\) on which we optimally solve the integer problem for \(\lambda_1\) to obtain \(S_3\). If the solution is different from \(S_1\) and \(S_2\), we apply a recursive algorithm that computes the
gradient $\lambda_i$ of successive solutions and optimally solves the integer problem on $\Omega$ in the new direction $\lambda_i$.

The second step aims to explore each triangle defined by two consecutive supported points and their nadir point. The algorithm between two consecutive non-dominated points is defined in Algorithm 2.

**Algorithm 2 Second step of two-steps method**

**Require:** Supported Point $S_i$ and $S_{i+1}$

1. Compute direction $\lambda_i'$, gradient between $S_i$ and $S_{i+1}$
2. Solve the linear relaxation of the parametrized formulation for $\lambda_i'$ to obtain $LB_i'$
3. Set $UB_i'$ to the nadir point $(S^1_{i+1} - 1, S^2_i - 1)$
4. $\Omega \leftarrow GENROUTE(LB_i', UB_i', \lambda_i')$
5. Solve the integer parametrized problem for $\lambda_i'$, denoted $PMP(\lambda_i')$, on $\Omega$ to obtain $S_i'$
6. If $PMP(\lambda_i')$ is feasible then
   - Repeat algorithm for $S_i$ and $S_i'$ and for $S_i'$ and $S_{i+1}$

**3 Results and Discussion**

Each method returns the exact minimum complete set of non-dominated points. Instances of BOVRPTW are built by merging pairs of Solomon’s instances. The implementation is in C++ and the linear problems and the integer problems are solved with Gurobi 7.1.

The figure 1 shows the comparison of the execution time for the methods in graphs with 25 customers. It shows that, for small graphs and random structures, the two-steps method is generally more efficient than the reference method.

The methods have also been tested on clustered instances and graphs with 50 customers (instance name beginning with 50) and the results are presented in the Table 1. For these
larger graphs and clustered instances, the *two-steps* method becomes much more efficient. Indeed, the *two-steps* method is a good compromise between the number of mono-objective solutions of the problem and the size of the gap between lower and upper bounds.

<table>
<thead>
<tr>
<th>Methods</th>
<th>C106_RC1</th>
<th>C106_C2</th>
<th>C106_R1</th>
<th>50R101_C2</th>
<th>50R101_RC2</th>
<th>50RC101_R1</th>
<th>50R105_C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref. method</td>
<td>&gt; limit</td>
<td>&gt; limit</td>
<td>&gt; limit</td>
<td>523</td>
<td>778</td>
<td>3208</td>
<td>41803</td>
</tr>
<tr>
<td>two-steps</td>
<td>3200</td>
<td>6150</td>
<td>22314</td>
<td>202</td>
<td>273</td>
<td>3061</td>
<td>15354</td>
</tr>
</tbody>
</table>

Table 1: Execution time of the methods in seconds. Time limit of 10 hours.

4 Conclusion

We have proposed an exact method for BOVRP that can generate all non-dominated points, supported and non-supported. This method is generic for classes of VRPs as it doesn’t exploit specific property. The results in Section 3 also prove that the *two-steps* method is competitive compared to the *reference method* in most cases, mainly for larger graphs and clustered instances. The method will be improved to be more competitive on all instances by exploiting the bi-objective property directly in the *GENROUTE* procedure and not only by cutting the objective space in different areas of research.

References


Solving a multi objective shortest path problem

Antoine Giret
Université François Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305,
Tours, France

Emmanuel Néron
Université François Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305,
Tours, France

Yannick Kergosien
Université François Rabelais de Tours, CNRS, LI EA 6300, OC ERL CNRS 6305,
Tours, France
Email: yannick.kergosien@univ-tours.fr

1 Introduction

The shortest path problem is a well-known problem in combinatorial optimization [8]. Since the first studies [9] and [11] – conducted more than 60 years ago – several variants of this problem are still being studied. The Multi-Objective Shortest Path (MOSP) problem is defined as follows: let \( G = (V, A) \) be a directed graph where \( V \) is the set of vertices and \( A \) the set of arcs. \( c_{ij}^k \) denotes the cost associated with the arc \((i, j) \in A\) for the criterion \( k \in K\), with \( K\) the set of criteria. We suppose that \( c_{ij}^k \geq 0\). The problem consists in finding a set of paths \( P \) from a source node \( s \) to a target node \( t \) minimizing several sum-type objectives functions. The result of the MOSP problem is a set of strictly non-dominated paths also called Pareto front. An efficient label-correcting algorithm – called Label-Correcting with Dynamic update of Pareto Front (LCDPF) – is proposed to solve the MOSP problem. This work is an extension of [10]. Computational experiments show that the LCDPF algorithm matches or outperforms the recent algorithms in the literature.

2 Recent advances in MOSP with sum-type objectives

In 2010, [13] proposes a label-correcting and label-setting algorithm (bLSET) using an acceleration technique improving the efficiency of these types of algorithm. He showed that it is not always necessary to propagate a label to the target node to confirm that it is dominated. According to [16], bLSET algorithm presented the best computational times among labelling approaches proposed during this period. Recently, [14] proposed a new
exact method, called Pulse algorithm, for the bi-objective shortest path problem and large-scale road networks. Pulse algorithm is based on recursive method using pruning strategies that speed up the graph exploration. The results show that the proposed algorithm outperforms the bLSET algorithm on very-large scale instances from the DIMACS dataset. In [15], the authors proposed a Dijkstra-like method generalizing the original method to only determine all extreme supported solutions of the bi-objective shortest path problem. The authors proved that the running times of the proposed method are $O(N(m+n \log n))$ with $n$ is the number of nodes, $m$ is the number of arcs and $N$ is the number of extreme supported points in the outcome space. Finally, [17] proposed a lower set calculation method for the bi-objective shortest path problem. The proposed lower bounds aim to improve the NAMOA* method [18] which is an exact generalization of an $A^*$ method to the multi-objective one-to-one problem.

3 The LCDPF algorithm

The LCDPF algorithm is composed of two phases, similarly to the general two-phase method introduced by [12] for solving bi-objective problem. The first phase consists in two steps. The goal of the first step is to determine some initial solutions belonging to the final Pareto front (the supported solutions). The second step computes partial paths, from pertinent nodes of the graph to the target node, in order to deduce some upper and lower bounds. This step also allows to remove useless nodes from the graph (nodes for which none non-dominated path in the final Pareto front solution passes through them). Both steps are based on a set of resolutions of reversed mono-objective shortest path problems using dijkstra’s algorithm of [11]. The second phase consists in finding all non-dominated solutions by exploiting informations obtained in the previous phase. To do this, the proposed algorithm is based on a classical label-correcting algorithm which has been introduced in [19] and for which we developed additional improvements to make the algorithm more efficient. The main specificity of the LCDPF algorithm is that the final Pareto front is dynamically built all along the process and not necessary when the target node is reached. This dynamic update allows to obtain a better estimation of the final Pareto front during the search and so the classic techniques to prune labels (partial solutions) are more efficient.

4 Computational Experiments

In order to compare our results with benchmark algorithms solving the MOSP problem, computational experiments have been performed on instances from the 9th DIMACS challenge. It consists of 3 graphs that represent New York City (264,346 nodes, 733,846 arcs), San Francisco Bay Area (321,270 nodes, 800,172 arcs) and the state of Florida (1,070,376 nodes, 3,241,380 arcs). The experiments show that the LCDPF algorithm outperforms the bLSET algorithm on very-large scale instances from the DIMACS dataset.
nodes, 2,712,798 arcs). For each comparison, we used the same pairs of source and target nodes. We developed our algorithm in C++ using the Boost Graph Library. Experiments are performed on a Linux machine with an Intel Xeon 2.67GHz, 8 cores and 8GB of RAM. First, we compared the LCDPF algorithm to some adaptations of the NAMOA* algorithm proposed by [17] (KDLS) and for which they used the NYC graph. Table 1 compares the better execution times among the algorithms proposed by [17] (columns KDLS) and the execution times of LCDPF algorithm (column LCDPF). The |S| column indicates the number of non-dominated solutions on the Pareto front for each instance.

Table 1: Comparison to [17]

| #  | KDLS  | LSDPF | |S| | #  | KDLS  | LSDPF | |S| | #  | KDLS  | LSDPF | |S| |
|----|-------|-------|----|----|----|-------|-------|----|----|----|-------|-------|----|----|
| 1  | 499.2 | **0.1** | 1,089 | 8  | -  | **2.1** | 7,391 | 15 | 0.7 | **0.4** | 1 |
| 2  | 21,363.7 | **0.6** | 1,469 | 9  | 3,379.2 | **1.4** | 919 | 16 | 5,201.4 | **4.4** | 2,034 |
| 3  | 4.2   | **0.4** | 16 | 722.7 | **2.0** | 774 | 17 | 7,668.2 | **0.9** | 1,724 |
| 4  | -     | **3.9** | 5,121 | 11 | 346.5 | **0.5** | 631 | 18 | 1,153.3 | **0.4** | 1,276 |
| 5  | 15,710.4 | **0.6** | 2,451 | 12 | 14,272.4 | **2.0** | 1,573 | 19 | -   | **1.5** | 4,224 |
| 6  | 5,697.4 | **1.2** | 1,502 | 13 | 21,782.2 | **0.6** | 3,046 | 20 | -   | **1.9** | 3,262 |
| 7  | 129.7 | **0.4** | 272 | 18,693.8 | **0.4** | 2,957 | |

Table 1 shows that execution times of algorithms proposed by [17] (with a time limit equals to 12h) are bigger than LCDPF algorithm. LCDPF algorithm may have an execution time of 12000 times faster than the best others algorithms. Computation times of LCDPF algorithm remain below the 5 second which is reasonable.

We also compared LCDPF algorithm to the bounded label-setting algorithm (blSET) proposed by [13] and the Pulse algorithm proposed by [14]. As presented in [14], the 30 instances of each graph has been clustered into the same three equal sized groups, denoted S (small), M (medium) and L (large), based on the number of non-dominated solutions found in the Pareto front. Table 2 compares average execution times between blSET, Pulse and LCDPF algorithms. Table 2 shows that execution times of blSET algorithm are in average bigger than other algorithms. Regarding the Pulse and LCDPF algorithms, the Pulse algorithm is particularly efficient on instances with a small number of solutions. For these instances, LCDPF is in average 1 to 4 times less effective to find the entire Pareto front, but execution times remain reasonable (< 2 seconds). LCDPF algorithm is much more efficient than the two others on medium to large instances. Indeed, it is in average 20 times faster than the Pulse algorithm and 120 faster than the blSET algorithm.

Finally, we compared our algorithm to the Ratio-Labeling BSP (RLBSP) algorithm proposed by [15] to find the supported extreme non-dominated solutions only. The authors also proposed a modified bi-objective label-setting algorithm for finding this type of solutions (called SLSET). Table 3 compares execution times between RLBSP, SLSET
and LCDPF* algorithms. LCDPF* algorithm is based on LCDPF algorithm that has been modified to find only the supported extreme non-dominated solutions. The results of Table 3 shows that LCDPF* algorithm has a better execution times in average than the others algorithms.

|       | blSET  | pulse | LCDPF | $|S|$ |
|-------|--------|-------|-------|-----|
| NY S  | 62,392.4 | 315.2 | 378   | 34.1|
| NY M  | 513,945.8 | 131,304 | 1,131 | 163 |
| NY L  | 8,812,59.6 | 1,367,664 | **2,849.5** | 422.7|
| BAY S | 6,777.7 | 160.1 | 381   | 8.8 |
| BAY M | 61,950 | 9,326.7 | 484   | 57.2|
| BAY L | 317,432.3 | 105,549.2 | **2,302** | 171.8|
| FLA S | 330,122.3 | 349.8 | 1,389 | 14.7|
| FLA M | 562,195.9 | 348,114.7 | **1,647** | 116.1|
| FLA L | 2,627,432.9 | 888,586 | **3,045** | 552.3|

Table 2: Comparison to [13] and [14]

<table>
<thead>
<tr>
<th></th>
<th>SLSET</th>
<th>RLBSP</th>
<th>LSDPF*</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>16,450</td>
<td>2,300</td>
<td><strong>791.9</strong></td>
</tr>
<tr>
<td>BAY</td>
<td>10,720</td>
<td>1,980</td>
<td><strong>536.9</strong></td>
</tr>
<tr>
<td>FLA</td>
<td>119,170</td>
<td>12,800</td>
<td><strong>1,878.4</strong></td>
</tr>
</tbody>
</table>

Table 3: Comparison to [15]

5 Conclusion

An efficient label-correcting algorithm – called Label-Correcting with Dynamic update of Pareto Front (LCDPF) – is proposed to solve the MOSP problem. Even if the graph is large, the LCDPF algorithm is able to find all non-dominated solutions in a short computing time, compared to best benchmark algorithms for real road networks. It composes of best improvement techniques that we can find in the literature about this problem and it integrates new ones referred later as: (1) quickly update of the Pareto front during the search that allows to efficiently prune partial paths of the search space and (2) remove useless nodes from the graph in a preprocessing phase.

References


Multi-objective optimization of a bi-modal two-echelon vehicle routing problem with synchronization arising in urban logistics

Alexandra Anderluh
WU Vienna University of Economics and Business, Austria
Email: alexandra.anderluh@wu.ac.at

Vera Hemmelmayr
WU Vienna University of Economics and Business, Austria

Pamela Nolz
AIT Austrian Institute of Technology, Austria

Teodor Gabriel Crainic
Université du Québec à Montréal, Canada

1 Introduction

The undeniable fact of increasing urbanization combined with the looming threats of climate change constitute currently two major challenges in the field of city logistics. Both developments complicate the supply of citizens with all required goods without deteriorating the quality of life in a city.

One way of dealing with this challenge is the use of emission-free or at least low-emission vehicles for urban freight transport. Cargo bikes fit perfectly in this profile of requirements, although two drawbacks of this mode of transport have to be taken into account. On the one hand cargo bikes have a limited capacity (100-250 kg for typical cargo bikes) and on the other hand the operational distance is restricted to some ten kilometers per day.

A two-echelon routing scheme with synchronization of vans and cargo bikes for transshipment purposes can tackle the above mentioned drawbacks of cargo bikes. Beside the economic objective of the problem also external effects as the environmental objective expressed by the minimization of emissions and the social objective including health risks induced by noise, should be considered for a holistic approach. Hence, factors of all three aspects of sustainability are to be included in the respective optimization model.

In the scientific literature one of the first papers dealing with multiple objectives in
routing focuses on a multi-objective truck and trailer problem [1]. Since then a number of papers have been published dealing with multi-objective pollution routing [2] or the consideration of environmental aspects in two-echelon vehicle routing, but not in a multi-objective way [3]. Multi-objective two-echelon routing problems have mainly been studied in the field of location routing [4].

Therefore, we develop a multi-objective bi-modal metaheuristic solution procedure to solve a two-echelon vehicle routing problem with synchronization between echelons and customer deliveries on both echelons. Preliminary computational results illustrate the influence of external effects and can therefore give planners decision support in using a sustainable and sophisticated freight distribution scheme in a city.

2 Problem Description and Methodology

The problem at hand deals with the combined and synchronized usage of cargo bikes and vans in a city distribution scheme. In this special two-echelon vehicle routing problem it is assumed that customer deliveries can be made by vehicles of both echelons. The reloading of goods from vans to cargo bikes requires synchronized meetings of vans and cargo bikes at satellites which are assumed to be transshipment points without storage facilities (see blue triangles in Figure 1). In contrast to the single-objective problem with preassigned customers to echelons dealt with in [5] (see green dotted circle in Figure 1 that marks the area inside the city center as “bike-supplied” and the area outside as “van-supplied”) we now assume a so-called “grey-zone” (see grey ring in Figure 1) - an area outside of the inner-city center - where customer deliveries can be done by vehicles of both echelons.

![Figure 1: City distribution scheme with vans and cargo bikes](image)

This assumption stems from discussions with experts as well as considerations about the usability of cargo bikes. Especially in European cities with a historic city center
we have to face numerous narrow one-way streets, pedestrian zones and the absence of parking space for vans in the inner-city center. In Vienna for example, this holds for the first district. Therefore, this area seems appropriate for cargo bike deliveries only. On the other side the area near the outskirts of a city is too far away to be supplied by cargo bikes assuming an action radius of 3-5 km around the bike depot for cargo bikes. Hence, all customers located outside this radius are preassigned to vans. All remaining customers located in the ”grey-zone” can be served either by vans or by cargo bikes. Therefore, the resulting problem is bi-modal in part.

Besides focusing on economic cost as only objective of the optimization problem, we take external effects into account in this work. We consider green house gas (GHG) emissions as climate-relevant externality and noise as health-relevant externality based on ideas described in [7]. Both external effects are assumed as distance-based and looked at in a combined way. GHG emissions have a direct component caused by burning fossil fuel and an indirect one caused by providing the required fuel or energy. Hence, also cargo bikes may cause a small amount of indirect GHG emissions, if they are electrically assisted. Noise on the other hand is assumed as only be caused by vans. Such modeling provides the means to analyze the various cost elements and decide on the strategy to adopt for both individual companies and the system regulator (e.g., municipal authorities).

The consideration of external effects beside economic cost as two equally important objectives in the vehicle routing problem requires a multi-objective method [6]. In this paper the \( \epsilon \)-constraint method is chosen to find pareto-efficient solutions.

The solution procedure starts with finding a feasible initial solution for one objective. First, cargo bike routes for all preassigned bike-customers are built by an approach inspired by a nearest neighbor construction heuristic. Based on the information about required synchronizations gained out of those cargo bike routes, van routes are built for all preassigned van-customers with the same approach and promising satellites are inserted in bike routes as well as van routes to fulfill all synchronization constraints. Then all remaining customers are inserted by a best insertion procedure. Additional satellites are inserted as needed. This initial solution is then improved by a neighborhood search with 4 different operators: 2-opt, move, swap and satellite-exchange. Then the procedure is repeated by adding the value of the second objective as additional constraint reduced by a varying factor \( \epsilon \) to find a number of pareto-efficient solutions.

3 Preliminary Results

First computational tests with artificial test instances have shown that the developed method is appropriate for this problem to find a pareto front of non-dominated solutions (see Figure 2). In detail those preliminary results show that the load transported by cargo
bikes can be increased by around 30% while economic cost increases by around 3% and external effects decrease by around 10% along the pareto front.

Figure 2: Pareto front of solutions

These results can already give first interesting insights into the problem, but further tests with different instances are required to confirm those findings. In addition, a multi-directional search [8] will be tested to further improve the pareto front.

Furthermore, external effects for a realistic test instance of the city of Vienna will be extended by taking city characteristics - the number of people living in a district as well as the vicinity of schools, kindergartens and hospitals - into account. Weighing external effects based on these characteristics shall give some further insights. The results of those further experiments, including on real data from the City of Vienna, will be presented at the conference.

Acknowledgements Part of this work is funded by the Austrian Research Promotion Agency as part of the JPI CONCOORD project (FFG Project No. 839739).

References

1 Introduction

Over the years, we can point to many cases where advancements in technology motivate new problems in operations research. The Close-Enough Traveling Salesman Problem (CETSP) is one of them. The classic Traveling Salesman Problem (TSP) requires a salesman to visit customers at their exact locations. For example, some utility companies send their workers to read the meters at every residential location. However, with radio frequency identification (RFID) technology, meters can be read from a distance. Today, instead of visiting every customer, workers need only get close enough to a house to read the meter. The CETSP also applies to aerial surveillance. An aircraft does not have to fly directly above the targets but only has to get close enough to survey them.

Typically, the problem is defined on a Euclidean plane. The salesman must start from and end at the depot. Every customer has a service region, which is usually assumed to be a circular disk centered at the customer location. A customer is visited when the salesman enters the customer’s service region. The objective is to visit all customers and return to the depot in the shortest distance traveled. The ratio of average customer radii
to the length of a side of the smallest square that encloses all customer disks is termed the overlap ratio. In the general case, where some customer radii are positive, the nodes of the tour are not all known in advance. Therefore, we must determine not only the sequence, but also the locations to visit.

The state-of-the-art heuristics that have been tested on large-size problems are found in Mennell [3]. The author developed several heuristics, with most based on Steiner zones. A Steiner zone is an overlap of disks. If a route passes through an overlap of several disks, all customers that define those disks are served. The Steiner zone heuristics differ in how to identify the Steiner zones and how to construct and improve feasible solutions.

In this paper, we develop a three-phase Steiner Zone Variable Neighborhood Search (SZVNS) heuristic for the CETSP that produces better than or comparable solutions to those produced in [3]. The running times are shorter. We also test SZVNS over small instances with known optimal solutions. SZVNS finds optimal solutions to most of the instances.

2 Algorithm

We develop a Steiner Zone Variable Neighborhood Search (SZVNS) heuristic to solve the CETSP. SZVNS has three phases. In the first phase, we remove customers whose disks contain the depot or a disk of another customer.

In the second phase, we develop a sweep line procedure that avoids examining all subsets of customers for overlaps. Then, we solve a set covering problem (SCP) to select the fewest Steiner zones that cover all customers. We then generate a feasible solution based on the selected Steiner zones by constructing a tour that passes through a Steiner point from each Steiner zone.

In the third phase, we integrate six search operators into a variable neighborhood search framework to improve the feasible solution. The TSP solver operator assumes fixed Steiner points and seeks a sequence to visit them that produces a shorter tour length. The Steiner point selection operator assumes a fixed visiting sequence of the Steiner zones and seeks locations, or Steiner points, to visit the Steiner zones to reduce tour length. The one-point insertion operator considers relocation of each Steiner zone to every possible position on the tour. The operator takes into account possible re-selection of Steiner points as well. In the two-point insertion operator, instead of relocating one Steiner zone, we relocate two Steiner zones visited consecutively on the tour. In the string insertion operator, we identify a possibly poorly routed string of Steiner zones guided by the convex hull of customer locations and relocate them to other positions on the tour. In the Steiner zone reconstruction operator, we find another set of Steiner zones based on the current tour to construct new solutions. This is a perturbation operator.
Table 1: Computational results for SZVNS on 720 instances from [2]

<table>
<thead>
<tr>
<th>Customer radius</th>
<th>Number of instances</th>
<th>Number optimal</th>
<th>Average gap (%)</th>
<th>Maximum gap (%)</th>
<th>Average time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>240</td>
<td>232</td>
<td>0.01</td>
<td>0.86</td>
<td>0.074</td>
</tr>
<tr>
<td>0.50</td>
<td>240</td>
<td>227</td>
<td>0.02</td>
<td>1.10</td>
<td>0.079</td>
</tr>
<tr>
<td>1.00</td>
<td>240</td>
<td>221</td>
<td>0.01</td>
<td>0.89</td>
<td>0.080</td>
</tr>
</tbody>
</table>

3 Computational experiments

We collected 802 benchmark instances and divided them into two groups. The first group has 740 instances with as many as 30 customers found in Coutinho et al. [2] and Carrabs et al. [1]. The second group has 62 instances with 100 to 1001 nodes given by Mennell et al. [3].

3.1 Results produced by SZVNS on 740 small instances

The 740 instances are divided into two sets. The first set of 720 instances have 6 to 20 customers. In each instance, the locations of the customers and depot were generated randomly on a 16 by 10 plane. The three customer radii are 0.25, 0.50, and 1.00. These instances have been solved optimally by Coutinho et al. [2]. The performance of SZVNS on this set is summarized in Table 1. There are 240 instances for each customer radius. In total, SZVNS produces 680 of 720 (94.4%) of the optimal solutions.

The second set was generated by Carrabs et al. [1]. This set has 60 instances with 25 and 30 customers. The results generated by SZVNS are comparable with those by Carrabs et al. but SZVNS runs much faster.

3.2 Results produced by SZVNS on 62 large instances

Among the set 62 large instances, 48 instances have constant radius (27 instances with varied overlap ratios and 21 instances with fixed overlap ratios) and 14 instances have arbitrary radii. In Table 2, we compare the performance of SZVNS with the best heuristic in the literature in each instance category. In summary, SZVNS outperforms the best heuristic in terms of average percentage gap and running time on instances with fixed radius and varied overlap ratio and instances with arbitrary radius. On instances with constant radius and fixed overlap ratios, SZVNS produces solutions with acceptable quality in much shorter time.
Table 2: Performance of SZVNS compared with the best existing heuristics in each instance category

<table>
<thead>
<tr>
<th>Instance category</th>
<th>Heuristic</th>
<th>Average gap (%)</th>
<th>Average time (s)</th>
<th>Number of shortest tours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant radius &amp; varied overlap ratio</td>
<td>SZVNS</td>
<td>0.37</td>
<td>93.224</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>HYBRID2</td>
<td>1.64</td>
<td>4940.799</td>
<td>8</td>
</tr>
<tr>
<td>Constant radius &amp; small overlap ratio</td>
<td>GTSP2</td>
<td>0.26</td>
<td>1906666.429</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>SZVNS</td>
<td>0.62</td>
<td>50.527</td>
<td>4</td>
</tr>
<tr>
<td>Constant radius &amp; moderate overlap ratio</td>
<td>SZ3₀−₃₆₀</td>
<td>0.71</td>
<td>7123.710</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>SZVNS</td>
<td>2.07</td>
<td>104.162</td>
<td>0</td>
</tr>
<tr>
<td>Constant radius &amp; large overlap ratio</td>
<td>SZ3₀−₃₆₀</td>
<td>0.00</td>
<td>8712.053</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>SZVNS</td>
<td>0.31</td>
<td>102.640</td>
<td>2</td>
</tr>
<tr>
<td>Arbitrary radius</td>
<td>SZVNS</td>
<td>0.12</td>
<td>12.618</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>HYBRID2</td>
<td>3.45</td>
<td>1371.606</td>
<td>1</td>
</tr>
</tbody>
</table>

4 Conclusions

We developed a three-phase heuristic (SZVNS) for the CETSP and tested SZVNS on 802 benchmark instances. On the smaller instances where the optimal solutions are known, SZVNS produced optimal solutions to 94.4% of the instances in a fraction of a second. On the other instances, SZVNS produced better or comparable results in shorter time compared to state-of-the-art algorithms.

References


Evaluating Exact vs. Rule-Based Algorithms for the Unending Real-Time Traveling Repairperson Problem Under True Simulated Operating Conditions

Matthew E. H. Petering
Department of Industrial and Manufacturing Engineering
University of Wisconsin-Milwaukee
Email: mattpete@uwm.edu

Copyright 2017 Matthew E. H. Petering

1 Introduction

This study continues the author’s work on intense unending real-time operational challenges (IURTOCs) that was first presented at Odysseus 2015 [1]. This research is motivated by the increasing prevalence in our world of industrial systems whose operations are unending, consistently busy, intense, and dynamic. Examples of such systems are international airports that never close, large seaport container terminals in Asia, and factories that operate 24/7. These systems operate around the clock and are consistently busy; the traditional paradigm of “plan at night” and “execute during the day” or vice versa does not apply. Instead, managers must plan and execute concurrently. In these systems, operations are intense; new jobs arrive frequently, processes are brief, and scores if not hundreds of unfinished jobs await to be served by scarce resources from time 0 to infinity. In many of these systems, operations are dynamic; new information relevant for decision making—concerning new or existing jobs and/or process durations—becomes available as operations unfold. The purpose of this research is to identify methods by which managers can repeatedly assign scarce system resources to jobs so as to minimize the (completion delay)*(weight) of the average job.

2 Literature review

Despite their pervasiveness, IURTOCs have been the focus of relatively few theoretical and experimental studies in the decision science literature. Several outstanding theoretical papers such as [2] have derived competitive ratios for, or have proven the asymptotic optimality of, online algorithms for a variety of online (i.e. real-time) settings. In most of these studies, the objective is to minimize makespan or total weighted job completion time. Note that makespan does not apply in an unending setting. Also, note that minimizing total weighted job completion time is a less demanding objective than minimizing average weighted job delay in an unending setting. Indeed, any algorithm that achieves a finite average job delay is asymptotically optimal if the objective is to minimize total job completion time [3]. Yet, a 20 min avg. job delay is preferable to a 200 min avg. job delay. Finally, note that a competitive ratio is a proven bound on an algorithm’s worst case performance in an online instance compared to the optimal solution of the corresponding offline instance. However, managers of real-world systems are probably more interested in an algorithm’s average performance.

IURTOCs have also received little attention in the experimental decision science literature. More than 99% of this literature has focused on finite, static, or calm (i.e. non-intense) settings. Among several hundred studies that consider unending, dynamic, and intense settings—most of which adopt a simulation methodology—only about 100 studies evaluate and compare different options for making decisions in real time [4]. These studies
typically embed one or more decision making algorithms (DMAs) within a discrete event simulation (DES) model in order to test their performance under simulated operating conditions [5]. For example, a previous study by the author proposes using a Hungarian-algorithm-based DMA for assigning multiple trucks to multiple cargo transportation tasks in real time on a continual basis at a multiple-berth seaport container terminal that never closes. The DMA is tested by embedding it in a DES [6]. Importantly, the DMAs in these studies are only partially embedded in a DES model. That is, only the decisions identified by the DMAs are fed back into a DES model; the computation times used to identify the decisions are not fed back into a DES model and are assumed to be zero.

3 Conceptual development

In this research, we introduce an alternative, improved experimental setup for testing DMAs for IURTOCs [1]. In particular, we suggest that proposed DMAs be fully embedded in a DES model so that both (A) the decisions made by the DMA and (B) the computation time used by the DMA are fed back into the DES model so as to affect the future evolution of the system state as tracked by the DES model [1]. In this alternative experimental setup, nontrivial DMA computation time can create a simulated delay for the machines/vehicles/jobs whose future schedules are being decided by the DMA. For example, the DES model may “blow up”—i.e. the number of unfinished jobs could increase without limit as time approaches infinity—if DMA runtime is too high to keep pace with the appearance of new jobs. Note that this phenomenon is very relevant to real-world industrial systems but is ignored by the traditional experimental framework in which DMAs are only partially embedded in DES models.

In this alternative experimental setup, we say that the DMA is being tested under true simulated operating conditions [1]. This concept is particularly important if a DMA’s average runtime is non-trivial compared to the average time that elapses between consecutive calls to the DMA. For example, if a DMA’s average runtime is 10 s and the DMA is called once per minute on average, it would be important to test it under true simulated operating conditions. However, if a DMA’s average runtime is 10 µs and the DMA is called roughly once per minute, then testing it under true simulated operating conditions is not likely to yield additional insight compared to traditional evaluation methods.

Note that, if the main objective of an IURTOC is time-related—e.g. to minimize average job delay—then the concept of true simulated operating conditions reconciles two measures of DMA effectiveness—solution quality and computation time—in a fair and unbiased manner. Consider two DMAs—1 and 2—that are proposed for an IURTOC with a time-related objective. DMA 1 (2) may be an exact (a heuristic) method that identifies high- (low-) quality decisions using much (little) computation time. Which DMA is better for addressing this IURTOC? The answer is clearly obtained by fully embedding each within the same DES model of that IURTOC. The DES model consolidates the solution quality and computation time of each DMA into a single numerical value (e.g. average job delay), leaving no doubt about which DMA is better for real-world use. To our knowledge, no other method in the literature is able to directly consolidate the two primary measures of DMA effectiveness—solution quality and computation time—into a single numerical value.

4 Unending real-time traveling repairperson problem (URTTRP)

In this project, we explore the above issues in the context of the traveling repairperson IURTOC. In this IURTOC, R repairpersons (RPs) must service repair jobs that appear at various times and locations in a 1000m x 1000m square region so that the weighted average time that elapses from a job’s earliest allowed start time to when a RP finishes servicing it is minimized. As time progresses, individual jobs or batches of jobs appear according to a user-
defined stochastic process. RPs do not need to return to a depot. Any RP can service any job; the repair time depends on the associated job and RP. The DES model is described in [1].

Four levels of job information imperfection are considered. At level 0, full information concerning each job’s (a) earliest allowed start time $e_j$, (b) location, (c) service time for each RP, and (d) weight is available at time 0. At level 1, job $j$’s full info for a-d becomes available N(20,10) min prior to $e_j$; no info is available prior to this time. At level 2, full info for a-b and partial info for c-d for job $j$ becomes available N(20,10) min prior to $e_j$; no info is available prior to this time. Full info for c-d becomes available at time $e_j$. At level 3, job $j$’s full info for a-d becomes available only at time $e_j$. There is no stochasticity beyond the above dynamism.

Table 1 shows the parameter values used. When $R = (5, 10, 15)$, about (480, 960, 1440) jobs appear during a 480 min timespan. At time 0, initial jobs are assigned to RPs via a call to CPLEX 12.5’s branch-and-cut method to solve an MILP model [1] that is inspired by [7].

### Table 1. DES model input parameter values (DU is the discrete uniform distribution).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of RPs</td>
<td>5, 10, or 15</td>
<td>Job X and Y coords. (independent)</td>
<td>DU(0,1000)</td>
</tr>
<tr>
<td>Duration simulated</td>
<td>480 min</td>
<td>RP speed</td>
<td>125 m/min</td>
</tr>
<tr>
<td>Assumed completion time for</td>
<td>490 min</td>
<td>Job repair duration in minutes</td>
<td>(0.8)*DU(1,9)</td>
</tr>
<tr>
<td>unfinished jobs at end of simulation</td>
<td></td>
<td>Job weight ($w_i$)</td>
<td>DU(1,9)</td>
</tr>
</tbody>
</table>

Experiments compare five types of DMAs—(1) “SRP,” (2) “MRP-TimeLimit-No,” (3) “MRP-Cts-Single-No,” (4) “MRP-Cts-Multi-No,” and (5) “MRP-Cts-Multi-Yes”—that automatically add jobs to the end of RPs’ to-do lists in repeated fashion as the simulation unfolds. DMA type 1 calls a lightning-fast rule-based method to decide the next job to be performed by a single RP (“SRP”) whenever a RP completes his/her to-do list. DMA type 2 calls the MILP to add many jobs to many RPs’ (“MRP”) to-do lists whenever a RP completes his/her to-do list. Four MILP time limits—10, 30, 60, and 120 sec—are considered, and RPs that finish their to-do lists while CPLEX is solving a MILP remain idle until the MILP routine terminates. DMA types 3-5 call the MILP routine continuously in rolling-horizon fashion. When the MILP routine finishes, these DMAs immediately call the MILP routine again with a pre-computed time limit intended to cause the routine to terminate at the first instant when an RP will finish his/her to-do list. These DMAs give CPLEX significant computation time while minimizing RP idleness due to delayed CPLEX results. The (“Single,” “Multi”) option determines whether CPLEX runs in single- or multi-core mode. In the former (latter) case, less (more) computation power is utilized, but the routine is (is not) guaranteed to finish at the pre-specified time limit. The (“Yes,” “No”) option refers to whether the job weights are considered in the MILP. Overall, DMA type 1 emphasizes acting, not thinking; DMA type 2 emphasizes the separation of thinking and acting to the extent possible; and DMAs 3-5 emphasize thinking while acting over the entire 480 min simulation.

## 5 Results and discussion

Table 2 shows the results. Each cell shows the average weighted job delay across ten reps. Only the best results in a given setting among (4, 4, 6, 6, 6) variations of DMA type (1, 2, 3, 4, 5) are shown. The variations of DMAs 3-5 impose a (5, 10, 15, 20, 30, 40) job limit on the MILP subroutine. Note the surprisingly good performance of the SRP DMA and “No” option.

## References


Table 2. Experimental results (values are the average weighted job delay in minutes).

<table>
<thead>
<tr>
<th># RPs</th>
<th>Job Inter.</th>
<th>Type of Decision Making Algorithm (DMA)</th>
<th>Level of Information Imperfection</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>E(1.1)</td>
<td>SRP</td>
<td>26.1</td>
<td>31.3</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>21.8</td>
<td>35.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>18.4</td>
<td>39.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>21.2</td>
<td>46.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>22.1</td>
<td>41.1</td>
</tr>
<tr>
<td>5</td>
<td>E(1)</td>
<td>SRP</td>
<td>34.7</td>
<td>39.4</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>44.1</td>
<td>59.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>29.2</td>
<td>57.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>35.3</td>
<td>76.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>26.7</td>
<td>58.9</td>
</tr>
<tr>
<td>5</td>
<td>E(0.9)</td>
<td>SRP</td>
<td>49.9</td>
<td>54.4</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>126.7</td>
<td>129.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>59.2</td>
<td>98.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>88.1</td>
<td>124.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>51.6</td>
<td>94.6</td>
</tr>
<tr>
<td>10</td>
<td>E(0.55)</td>
<td>SRP</td>
<td>12.4</td>
<td>17.8</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>8.7</td>
<td>16.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>7.8</td>
<td>21.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>7.7</td>
<td>23.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>7.7</td>
<td>22.3</td>
</tr>
<tr>
<td>10</td>
<td>E(0.5)</td>
<td>SRP</td>
<td>15.1</td>
<td>20.8</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>29.2</td>
<td>28.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>12.3</td>
<td>37.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>15.7</td>
<td>36.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>13.4</td>
<td>30.4</td>
</tr>
<tr>
<td>10</td>
<td>E(0.45)</td>
<td>SRP</td>
<td>20.8</td>
<td>25.8</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>86.7</td>
<td>104.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>27.9</td>
<td>55.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>74.0</td>
<td>74.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>38.0</td>
<td>46.6</td>
</tr>
<tr>
<td>15</td>
<td>E(0.3667)</td>
<td>SRP</td>
<td>8.6</td>
<td>13.3</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>4.9</td>
<td>11.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>4.9</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>5.1</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>4.9</td>
<td>15.1</td>
</tr>
<tr>
<td>15</td>
<td>E(0.3333)</td>
<td>SRP</td>
<td>10.3</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>16.6</td>
<td>24.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>7.1</td>
<td>18.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>7.6</td>
<td>20.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>7.4</td>
<td>20.5</td>
</tr>
<tr>
<td>15</td>
<td>E(0.3)</td>
<td>SRP</td>
<td>12.9</td>
<td>18.2</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
<td>MRP-TimeLimit-No</td>
<td>111.1</td>
<td>103.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Single-No</td>
<td>13.7</td>
<td>29.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-Yes</td>
<td>19.3</td>
<td>34.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MRP-Cts-Multi-No</td>
<td>17.3</td>
<td>26.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg.</td>
<td>28.6</td>
<td>42.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>42.6</td>
<td></td>
</tr>
</tbody>
</table>
Solving Time Dependent Traveling Salesman with Time Windows Problems

Mike Hewitt, Minh Vu Duc  Natashia Boland, Martin Savelsbergh
Loyola University Chicago  Georgia Institute of Technology

1 Introduction
We study a class of problems wherein a vehicle departs from an initial location, often called a depot, visits each location in a known set of locations exactly once and during a pre-defined time window, and then returns to the depot. The problem of determining a set of such movements, what is commonly referred to as a tour, is known as the Traveling Salesman Problem with Time Windows (TSPTW). We also consider travel times and travel costs that depend on when travel occurs, meaning that we seek to solve what has been called the Time Dependent Traveling Salesman with Time Windows Problem (TD-TSPTW). As is commonly done, we assume travel times adhere to the First-In-First-Out (FIFO) property, meaning that when traveling between two locations, a later departure leads to a later arrival. We present an algorithm that can handle different objective functions, including a Makespan objective wherein the objective is to complete the tour, including returning to the depot, as quickly as possible, and a tour travel time objective.

The problem we study wherein the Makespan objective is optimized was also considered by [1] and [4]. [4] presents the best results for this problem to date, which they achieved by solving a new formulation of the problem with a branch-and-cut algorithm. We present an integer programming formulation of the problem based on a time-expanded network, wherein the same location at different points in times is represented with different nodes. By doing so, we embed time dependency in the definition of the network. We adapt the Dynamic Discretization Discovery method presented in [2], and previously used to solve the TSPTW [3], to solve this problem wherein there are time dependencies. We see that our method outperforms that of [4] on the problem wherein a Makespan objective is minimized. We are unaware of any methods that optimize the TD-TSPTW under the Travel time objective.

2 Problem description and formulation
The TD-TSPTW is defined as follows. We let \((N, A)\) denote a directed graph, wherein the node set \(N = \{0, 1, 2, \ldots, n\}\) includes the depot (node 0) as well as the set of locations (node 1, \ldots, n) that must be visited. Associated with each location \(i = 1, \ldots, n\) is a time window \([e_i, l_i]\) during which the location must be visited. Note that the vehicle may arrive at city \(i \in N \setminus \{0\}\) before \(e_i\), in which case it must wait until the time window opens.
There is also a time window, \([e_0, l_0]\), associated with the depot which indicates that the tour can depart the depot no earlier than \(e_0\) and must return no later than \(l_0\). The set \(A \subseteq N \times N\) consists of arcs that represent travel between locations. Associated with each arc \((i, j) \in A\) and time, \(t\), at which travel can begin on the arc, is a travel time, \(\tau_{ij}(t)\), and travel cost, \(c_{ij}(t)\). Mathematically, the FIFO property implies that for each arc \((i, j) \in A\) and times \(t, t'\) wherein \(t' \geq t\), we must have \(t + \tau_{ij}(t) \leq t' + \tau_{ij}(t')\). Thus, formally, the vehicle travels on a tour which departs from node 0 at time \(t \geq e_0\), arrives at each city \(j \in N\) exactly once and within its time window \([e_j, l_j]\) by traveling on arcs in \(A\), and then returns to node 0 at time \(t' \leq l_0\).

We formulate determining this tour as an integer program defined on a time-expanded network, \(D = (N, A)\), with node set \(N\) and arc set \(A\). As such, for each node \(i \in N, t \geq e_i, N\) contains the node \((i, t)\). \(A\) contains two types of arcs, with the first representing travel between locations and the second representing waiting at a location. Specifically, the first type of arc is of the form \(((i, t), (j, t'))\) wherein \((i, j) \in A, t \geq e_i, t' = \max\{e_j, t + \tau_{ij}(t)\}\) (the vehicle can not visit early), and \(t' \leq l_j\) (the vehicle can not arrive late). Similarly, the second type of arc is of the form \(((i, t), (i, t + 1))\) and models waiting at location \(i\).

To formulate the integer program, for each arc \(((i, t), (j, t')) \in A\) we define the binary variable \(x_{((i, t), (j, t'))}\) to represent whether the vehicle travels along that arc. We seek to solve the TD-TSPTW(\(D\)):

\[
\text{minimize } \sum_{((i, t), (j, t')) \in A} c_{ij}(t)x_{((i, t), (j, t'))}
\]

subject to

\[
\sum_{((i, t), (j, t')) \in A: i \neq j} x_{((i, t), (j, t'))} = 1, \quad \forall j \in N, \quad (1)
\]

\[
\sum_{((i, t), (j, t')) \in A} x_{((i, t), (j, t'))} - \sum_{((j, t'), (i, t)) \in A} x_{((j, t'), (i, t))} = 0, \quad \forall (i, t) \in N, \quad (2)
\]

\[
x_{((i, t), (j, t'))} \in \{0, 1\}, \quad \forall ((i, t), (j, t')) \in A. \quad (3)
\]

The nature of the objective depends on the values of \(c_{ij}(t)\). To represent the Makespan objective, we set \(c_{i0}(t) = t + \tau_{i0}(t), \forall ((i, t), (0, t')) \in A, i \neq 0\) to model that costs are only incurred when returning to the depot. We set \(c_{ij}(t) = 0, \forall ((i, t), (j, t')), j \neq 0\) for the same reason. Constraints (1) ensure that the vehicle arrives at each node exactly one time during its time window. Constraints (2) then ensure that the vehicle departs every node at which it arrives. Finally, constraints (3) define the decision variables and their domains. We note that the impact of time windows and time-dependent travel times on the feasible region of the TD-TSPTW are embedded in the time-expanded network, \(D\). Regarding time windows, that arrival must occur within a node’s time window is ensured because \(A\) does not contain arcs that arrive at a node outside of its time
window. Time-dependent travel times are embedded in the construction of the arcs, \(((i, t), (j, t')) \in \mathcal{A}\). Time-dependent travel costs are modeled in the objective function coefficients, \(c_{ij}(t)\). We illustrate an example of \(\mathcal{D}\) in Figure 1. Note that both the travel times and travel costs differ when departing from \(i\) at time period 2 instead of time period 1.

The challenge associated with such a formulation of the TD-TSPTW is that the time-expanded network, \(\mathcal{D}\), and resulting integer program may be too large for a commercial solver to solve in a reasonable period of time. As such, we adapt the method proposed in [2], which instead generates time-expanded networks iteratively and dynamically.

### 3 Solution method and computational results

We present in Figure 2 a flow chart of a Dynamic Discretization Discovery (DDD) algorithm. At an iteration, the solution approach derives a partially time-expanded network, \(\mathcal{D}_T = (\mathcal{N}_T, \mathcal{A}_T)\), from a given subset of the timed nodes, \(\mathcal{N}_T \subseteq \mathcal{N}\). Given \(\mathcal{N}_T\), the arc set \(\mathcal{A}_T \subseteq \mathcal{N}_T \times \mathcal{N}_T\) consists of arcs of the form \(((i, t), (j, t'))\), wherein \((i, t) \in \mathcal{N}_T\), \((j, t') \in \mathcal{N}_T\), \(i \neq j\), and \((i, j) \in \mathcal{A}\). In other words, \(\mathcal{A}_T\) consists of arcs that model travel between locations, and not arcs that model waiting. We note that we do not require that arc \(((i, t), (j, t'))\) satisfies \(t' = \max\{e_j, t + \tau_{ij}(t)\}\) when \(i \neq j\), but only \(t' \leq \max\{e_j, t + \tau_{ij}(t)\}\). We note that because of the presence of short arcs, the choice of arc, \(((i, t), (j, t'))\), implies that the departure time at \(i\) is at least \(t\).

One critical step in implementing a DDD algorithm is determining the MIP to solve at an iteration that is in fact a relaxation of the original problem (and thus yields a lower bound). To formulate such a relaxation, for each \(a = ((i, t), (j, t')) \in \mathcal{A}_T\) wherein \(i \neq j\), we set \(c_{ij}(t) = \min\{c_{ij}(\bar{h})| t \leq \bar{h} \leq l_j - \tau_{ij}(\bar{h})\}\). Thus, like travel times, we set these costs, \(c_{ij}(t)\), so they under-estimate the cost of travel in \(\mathcal{D}\). We have proven that optimizing the TD-TSPTW over the partially time-expanded network, \(\mathcal{D}_T\), and with respect to the arc costs \(c_{ij}(t)\), yields a lower bound on the original TD-TSPTW. In Figure 3, we illustrate an example of \(\mathcal{D}_T\) that corresponds to the full time-expanded network presented in Figure 1. Note the lack of node \(j\) at time 4 means that the arc \((i, j)\) departing
at time 2 instead terminates at (j, 3). Also, because the cost incurred by departing from i at time 3 is 7, that cost is associated with the departure at time 2.

A primal solution is produced at an iteration (Step 2) by solving a TD-TSPTW formulated on \( D_T \) wherein arc travel times are over-estimated, and thus the tour is guaranteed to be feasible (although the integer program may not be). Step 3 is performed by comparing the best known primal solution with the lower bound produced in Step 1. The implementation of Step 4, wherein \( D_T \) is refined, and nodes (arcs) added to \( N_T (A_T) \) is the same as the procedure proposed in [3].

In Table 1, we summarize the performance of the DDD algorithm and that of [4] (called TTBF-CB) on each set of instances presented in [1]. We note that [4] do not report results for all instances from Set \( w100 \). While TTBF-CB is able to solve nearly all the instances, DDD is able to solve all of them. In addition, DDD is faster.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Traffic pattern</th>
<th>TTBF-CB</th>
<th></th>
<th>DDD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Inst</td>
<td>Slv</td>
<td>Tme</td>
</tr>
<tr>
<td>Set 1</td>
<td>A</td>
<td>475</td>
<td>470</td>
<td>31.99</td>
</tr>
<tr>
<td>Set 1</td>
<td>B</td>
<td>474</td>
<td>470</td>
<td>20.44</td>
</tr>
<tr>
<td>Set ( w100 )</td>
<td>A</td>
<td>108</td>
<td>107</td>
<td>100.13</td>
</tr>
<tr>
<td>Set ( w100 )</td>
<td>B</td>
<td>108</td>
<td>107</td>
<td>93.10</td>
</tr>
</tbody>
</table>

Table 1: Performance on [1] instances

The two sets of instances differ in the number of time intervals during the day during which travel times are constant (i.e. there is no time-dependency), with Set 1 having few intervals and Set \( w100 \) having many more. While the performance of TTBF-CB is sensitive to this instance parameter, DDD is not. We believe this robustness illustrates the benefits of using a time-expanded network to model time-dependent travel times and/or costs.

References


The Team Orienteering Problem with Overlaps

Christos Orlis
Department of Information, Logistics and Innovation
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
Email: c.orlis@vu.nl

Roberto Roberti
Department of Information, Logistics and Innovation
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Nicola Bianchessi
Department of Information, Logistics and Innovation
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Wout Dullaert
Department of Information, Logistics and Innovation
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

1 Introduction

In the Team Orienteering Problem (TOP), a set of routes is constructed with the aim of maximizing the total collected profit from a set of service points within a maximum route length constraint. We introduce the Team Orienteering Problem with Overlaps (TOPO) that generalizes the TOP by incorporating a monotone submodular objective function and models the case where the profit of serving a set of service points is lower than the sum of the profits of the individual service points. The TOPO is motivated by a real-life Automated Teller Machine (ATM) replenishment problem maximizing the number of bank customers having access to cash withdrawal. The submodularity in our case comes from the fact that bank customers may have access to cash withdrawal by using one of several nearby ATMs over a given period of time. Thus, the profit obtained by replenishing a subset of ATMs is subset-dependent and possibly lower compared to the profit of the TOP counterpart.
2 Problem Description

The TOPO is defined as follows. A set of customers \( C \) is given. Customers are served via a set of service points \( S \). In particular, each customer \( c \in C \) can be served by a subset of service points \( S_c \subseteq S \). The subset of customers that can be served by service point \( s \in S \) is indicated by \( C_s \subseteq C \). A set of homogeneous vehicles \( K \) located at a depot can be used to serve the customers via the service points. The depot is indicated by 0. Each vehicle can perform a route that starts from the depot, visits some service points, and ends at the depot. The maximum route length is indicated with \( T \). The travel time between the depot and each service point, and between each pair of service points is indicated with \( t_{ij} \). Travel times are asymmetric and strictly positive. The TOPO aims at finding a subset of routes, each respecting the maximum route length constraint, visiting each service point at most once in order to maximize the number of customers served.

3 Compact Formulation

The TOPO can be represented on a directed graph \( \mathcal{G} = (\mathcal{V}, \mathcal{A}) \), where the vertex set is defined as \( \mathcal{V} = \{0\} \cup S \) and the arc set as \( \mathcal{A} = \{(i, j) \mid i, j \in \mathcal{V} : i \neq j \} \).

Let us define the following sets of variables:

- \( x_{ij} \in \{0, 1\} \): binary variable equal to 1 if arc \( (i, j) \in \mathcal{A} \) is traversed by one of the vehicles (0 otherwise);
- \( y_c \in \{0, 1\} \): binary variable equal to 1 if customer \( c \in C \) is served (0 otherwise);
- \( z_s \in \mathbb{R}_+ \): continuous variable indicating the arrival time at service point \( s \in S \).

\[
\begin{align*}
\text{max} \quad & \sum_{c \in C} y_c & \quad (1) \\
\text{s.t.} \quad & \sum_{(i, j) \in \mathcal{A}} x_{0j} \leq |K| & \quad (2) \\
& \sum_{(i, j) \in \mathcal{A}} x_{ij} \leq 1 & \quad i \in S \quad (3) \\
& \sum_{(i, j) \in \mathcal{A}} x_{ij} - \sum_{(j, i) \in \mathcal{A}} x_{ji} = 0 & \quad i \in S \quad (4) \\
& z_i + (M + t_{ij})x_{ij} \leq z_j + M & \quad (i, j) \in \mathcal{A} \quad (5) \\
& \sum_{(i, j) \in \mathcal{A} : i \in S_c} x_{ij} \geq y_c & \quad c \in C \quad (6) \\
& x_{ij} \in \{0, 1\} & \quad (i, j) \in \mathcal{A} \quad (7) \\
& t_{0s} \leq z_s \leq T - t_{s0} & \quad s \in S \quad (8) \\
& y_c \in \{0, 1\} & \quad c \in C \quad (9)
\end{align*}
\]
The objective function (1) asks for maximizing the objective function value of the TOPO. Constraints (2) ensure that no more than \(|\mathcal{C}|\) routes are designed. Constraints (3) guarantee that each service point is visited at most once. Constraints (4) are flow conservation constraints for the service points. Constraints (5) link \(x\) and \(z\) variables to set the arrival times based on the traversed arcs and also prevent subtours in the designed routes. Constraints (6) ensure that each customer is served only if one of the corresponding service points is visited. Constraints (7)-(9) define the range of the decision variables. Constraints (8) also guarantee that the maximum route length is not exceeded.

4 Proposed Heuristic Method

In order to be able to tackle instances of reasonable magnitude, a heuristic solution procedure has been developed. We propose an augmented large neighborhood search method incorporating local search, insertion, and replacement improvement algorithms. In particular, after creating an initial feasible solution by using a construction heuristic, we start generating more feasible solutions of which the \(n\) best are stored in memory. Furthermore, solutions with objective function values lower than the \(n\) best are also stored in memory when their structures are highly possible to be significantly different than the existing ones. In each iteration of the heuristic, a feasible solution is selected randomly from the memory and some of its service points are removed in a random fashion in order to create space for new candidate solutions. We apply two families of insertion operators: one based on a simple insertion, and one incorporating a random component exchanging small sets of visited service points with unvisited ones. Both insertion operators are accompanied with two local search operators minimizing 1) the total routing distance and 2) the minimum routing distance and creating more space for inserting more service points in the routes.

5 Computational Results

In order to assess the behavior of our heuristic method a computational study is conducted. Our heuristic is tested on two families of synthetic instances for the TOP and the TOPO by utilizing the well-known set of TOP instances of Chao et al. [1]. Promising results are produced. More specifically, for the TOPO we show that we are able to produce results of high quality in relative short computation times. On top of this, we show that our heuristic is competitive against existing heuristic methods for solving instances of the TOP.

References

Multi-visit Clustered Team Orienteering Problem

Saïd Hanafi
LAMIH, CNRS UMR 8201,
University of Valenciennes and Hainaut Cambrésis, France

Renata Mansini, Roberto Zanotti
Department of Information Engineering,
University of Brescia, Brescia, Italy
Email: renata.mansini@unibs.it

1 Introduction

The introduction of clusters generalizes vehicle routing problems by grouping customers into groups, and by imposing that all customers in a cluster must be served consecutively by the same vehicle (see e.g. [3]). In this paper, we consider a variant of the Team Orienteering Problem (TOP) where each cluster may be served by possibly more than one vehicle (multi-visit clusters). This new problem provides the first contribution on a clustered variant of TOP.

In the classical TOP, each customer has a positive profit, and the goal of the problem is to determine one route for each vehicle, each one limited by a traveling time threshold, so to maximize the global collected profit. We consider a slightly modified version of TOP in which each customer is associated with a cluster of nodes that have to be visited in a precise order by possibly different vehicles. The problem is motivated by the real case study of a large company involved with kitchen equipments and their assembly on site. The company has to serve a huge number of customers and to this aim it has a team of technicians specialized in different activities (electricians, plumbers, masons, bricklayers, etc.). Each customer requires a service characterized by a set of tasks (nodes of the cluster) that have to be carried out in a precise order by possibly different technicians. For instance, to install a sink with a waste disposer, first we need a mason, then a plumber and finally an electrician, exactly in this sequence. Each task has a predefined duration. To avoid loss of time, the technicians have to move independently among the various customers respecting priority of tasks inside each cluster. The problem looks for the subset of clusters maximizing the collected profit without exceeding the working time limit established for each technician. We call this problem the Multi-visit Clustered TOP (MCTOP).

This paper contributes to the literature in several ways. We propose a new variant
of TOP never studied before, and provide a compact mixed integer linear programming (MIP) formulation for it. We introduce a new variant of the Kernel Search (KS) framework (see [1]) for its solution. Preliminary results show that our method provides very good solutions in a reasonable amount of time with respect to Gurobi 7.5.1 when solving the compact formulation with the addition of connectivity constraints in one hour time limit.

2 Problem definition and formulation

The MCTOP can be defined over a direct graph \( G = (V, A) \) with node set \( V = N \cup \{0, n + 1\} \), where 0 and \( n + 1 \) are initial and terminal depots and \( N = \{1, \ldots, n\} \), and arc set \( A = \{(i, j) : i \neq n + 1; j \neq 0; i \neq j; i, j \in V\} \). Let \( K \) be the set of technicians (vehicles) available to execute tasks. Nodes are partitioned into \( m \) disjoint and non-empty subsets \( C_h, h = 1, \ldots, m \), called clusters, such that \( N = \bigcup_{h=1}^{m} C_h \). Each node \( i \in N \) is associated with exactly one task that has a predefined execution time \( d_i \), and has to be performed by exactly one technician enabled to execute it. We refer to \( K_i \subseteq K \) as the subset of technicians that can execute the task associated with node \( i \). Nodes belonging to a cluster represent the tasks required by the same customer, have the same location, and are ordered according to a predefined execution sequence. A positive profit \( p_h, h = 1, \ldots, m \), is associated with each cluster \( C_h \). In order to collect such a profit, each node \( i \in C_h \) has to be visited by exactly one vehicle. We define as \( t_{ij} \) the nonnegative time needed to travel from node \( i \) to node \( j \). Each vehicle \( v \in K \) leaves the depot 0 at time 0. The MCTOP looks for a route (starting at 0 and ending at \( n + 1 \)) for each vehicle, such that the total collected profit is maximized and the total time required by each vehicle (as sum of traveling and execution time) does not exceed a threshold \( T_{max} \).

We model the problem by means of four sets of variables. Binary variables \( x_{v}^u, (i, j) \in A, v \in K \) and \( u_i^v, i \in N, v \in K_i \) are equal to 1 if arc \((i, j)\) is traversed and node \( i \) is visited by vehicle \( v \), respectively; continuous variable \( z_{ij}, (i, j) \in A \setminus \{(0, n + 1)\} \) indicates the arrival time at node \( j \) when arriving from node \( i \), whereas binary variable \( y_h \) takes value 1 when cluster \( C_h \) is visited (all nodes of the cluster are served). For each set \( S \subseteq N \), let \( \delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\} \) and \( \delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\} \) be the set of arcs leaving and entering the set \( S \), respectively, with \( \delta^+(i) = \delta^+\{\{i\}\} \) and \( \delta^-(i) = \delta^-\{\{i\}\} \). The compact MIP model is as follows:

\[
\max \sum_{h=1}^{m} p_h y_h \tag{1}
\]

\[
\sum_{(0,j) \in \delta^+(0)} x_{0i}^v = \sum_{(i,n+1) \in \delta^-(n+1)} x_{i,n+1}^v = 1 \quad v \in K \tag{2}
\]

\[
\sum_{(i,j) \in \delta^+(i)} x_{ij}^v = \sum_{(j,i) \in \delta^-(i)} x_{ji}^v = u_i^v \quad i \in N, v \in K_i \tag{3}
\]
\[ \sum_{v \in K_i} u_i^v = y_h \quad i \in C_h, h = 1, \ldots, m, \]  
\[ \sum_{(i,j) \in \delta^+(i)} z_{ij} - \sum_{(j,i) \in \delta^-(i)} z_{ji} = \sum_{v \in K} \sum_{(i,j) \in \delta^+(i)} (t_{ij} + d_i) x_{ij}^v \quad i \in N \]  
\[ z_{0i} = t_{0i} \sum_{v \in K} x_{0i}^v \quad i \in N \]  
\[ (t_{0i} + t_{ij} + d_i) \sum_{v \in K} x_{ij}^v \leq z_{ij} \leq (T_{max} - d_j - t_{j,n+1}) \sum_{v \in K} x_{ij}^v \quad (i,j) \in A \setminus \{(0,n+1)\} \]  
\[ \sum_{(k,i) \in \delta^-(i)} z_{ki} + d_i \sum_{v \in K} u_i^v \leq \sum_{(k,j) \in \delta^-(j)} z_{kj} \quad i \in \bar{C}_h, h = 1, \ldots, m, j = i + 1 \]  
\[ z_{ij} \geq 0 \quad (i,j) \in A \setminus \{(0,n+1)\} \]  
\[ x_{ij}^v \in \{0,1\} \quad (i,j) \in A, v \in K \]  
\[ u_i^v \in \{0,1\} \quad i \in N, v \in K_i \]  
\[ y_h \in \{0,1\} \quad h = 1, \ldots, m \]  

The objective function (1) maximizes the total collected profit. Constraints (2) ensure that exactly |K| vehicles exit node 0 and arrive at node n + 1. Constraints (3) impose that if vehicle v visits node i (u_i^v = 1), then the same compatible vehicle with i (v \in K_i) has to enter and leave the node. Constraints (4) ensure that each node i of a cluster C_h will be served by exactly one compatible vehicle, and that all nodes of the cluster are served if and only if the cluster is visited (y_h = 1). Constraints (5) ensure that, if a vehicle v visits node j immediately after node i, the time elapsed between the arrival times in the two nodes is equal to the time \(d_i\) needed to execute the task required by node i plus the travel time \(t_{ij}\) between i and j. Constraints (6) serve the same purpose of constraints (5) providing a bound on the starting node, whereas constraints (7) set lower and upper bounds on the duration of each route. Constraints (8) establish precedence relations among nodes belonging to the same cluster, where \(\bar{C}_h\) is the set of nodes in cluster \(C_h\) but the last one. Finally, constraints (9) – (12) impose nonnegative and binary conditions on variables.

In order to strengthen the formulation, classical connectivity constraints are also added:

\[ \sum_{(i,j) \in \delta^+(S)} x_{ij}^v \geq u_i^v \quad S \subseteq N, i \in S, v \in K. \]  

### 3 Solution algorithm and experimental analysis

Kernel Search is a heuristic framework developed for the solution of MIP problems. The method is based on the identification of a set of promising variables (the kernel set) and the sequential solution of small MIP subproblems constructed using the kernel set and the remaining variables divided into buckets. The method may consider the sequence of
buckets more than once (bucket iterations), possibly varying their size. We innovate on the method by changing the rules used to construct the initial kernel set and to update it at each iteration, respectively. Relevant variables used to create the kernel set and the buckets are the ones associated with the clusters. To identify initial kernel set, we boost up information provided by the LP relaxation with the introduction of a ranking function considering the location of each cluster, its profit and cardinality, and the total service time required to serve it. To allow an effective use of computing time, buckets size and partial overlapping is changed at each bucket iteration.

Tested instances have been generated from benchmark sets introduced for TOP in [2], by keeping the same graphs and profits. All remaining data are generated randomly. Preliminary results are extremely promising. In Table 1, we report the results obtained in 15 among the most complex instances. Columns (Obj) indicate the objective function value of the best integer solution found by Kernel Search in 10 minutes and by Gurobi in one hour, respectively. Symbol ‘–’ indicates that Gurobi was unable to find a feasible solution. Columns (tBest) show the time required to find the best feasible solution. In all tested instances, solutions found by our method are highly better than Gurobi’s ones. To provide a tighter bound, we are developing a Branch-and-Cut approach dynamically using different valid inequalities.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Kernel Search</th>
<th>Gurobi</th>
<th>Kernel Search</th>
<th>Gurobi</th>
</tr>
</thead>
<tbody>
<tr>
<td>p4.2.n</td>
<td>499 309</td>
<td>368 2976</td>
<td>312 52</td>
<td>312 3495</td>
</tr>
<tr>
<td>p4.3.o</td>
<td>304 292</td>
<td>– 297</td>
<td>312 183</td>
<td>312 3495</td>
</tr>
<tr>
<td>p4.3.q</td>
<td>356 482</td>
<td>– 3495</td>
<td>237 128</td>
<td>235 1605</td>
</tr>
<tr>
<td>p4.4.s</td>
<td>329 566</td>
<td>– 312</td>
<td>226 111</td>
<td>225 2256</td>
</tr>
<tr>
<td>p4.4.t</td>
<td>354 175</td>
<td>39 2593</td>
<td>461 216</td>
<td>376 3435</td>
</tr>
<tr>
<td>p6.2.n</td>
<td>732 349</td>
<td>750 828</td>
<td>399 204</td>
<td>146 3208</td>
</tr>
<tr>
<td>p6.3.l</td>
<td>438 358</td>
<td>420 1539</td>
<td>399 204</td>
<td>146 3208</td>
</tr>
<tr>
<td>p6.3.n</td>
<td>396 384</td>
<td>366 1477</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison between Kernel Search and Gurobi on 15 benchmark instances.

References


Dynamic programming for the Electric Vehicle Orienteering Problem with multiple technologies

Dario Bezzi
Department of Computer Science, University of Milan, Italy

Alberto Ceselli
Department of Computer Science, University of Milan, Italy

Giovanni Righini
Department of Computer Science, University of Milan, Italy
Email: giovanni.righini@unimi.it

1 Problem description

The Electric Vehicle Routing Problem (EVRP) has been introduced by Erdogan and Miller-Hooks under the name of Green Vehicle Routing Problem in [1]. Several variations have been studied, including problem with time windows, partial recharges, multiple technologies and both exact and heuristic algorithms have been developed. Examples of heuristic algorithms for the EVRP are given in Felipe et al. [2], Schneider et al. [3] and Koc and Karaoglan [4]. More references on VRP variants involving the use of electric vehicles can be found in a recent and extensive survey by Pelletier et al. [5].

The computation of exact solutions is more challenging than for the classical VRP, because of the additional subproblem of deciding the optimal recharges at some points along the routes. An additional source of complexity is the presence of different recharge technologies, each one characterized by a unit cost and a recharge speed. Schiffer and Walther [6] recently considered a similar problem in the context of location-routing. Sweda et al. [7] studied the optimal recharge policy when the route is given. As with many other variations of the VRP, the most common choice to design effective exact optimization algorithms is to rely upon branch-and-cut-and-price, starting from a reformulation of the routing problem as a set covering or set partitioning problem, where each column represents the duty of a vehicle. For instance, Desaulniers et al. [8] developed a branch-and-price-and-cut algorithm for the exact solution of the EVRP with time windows. In this study we investigate the Electric Vehicle Orienteering Problem, arising as a pricing sub-problem when the EVRP is solved by branch-and-price and in particular we consider a dynamic programming algorithm for the case with multiple technologies.
2 Formulation

Let $G = (\mathcal{N} \cup \mathcal{R}, \mathcal{E})$ be a given weighted undirected graph whose vertex set is the union of a set $\mathcal{N}$ of customers and a set $\mathcal{R}$ of recharge stations. A distinguished station in $\mathcal{R}$ is the depot, numbered 0. A fleet of $V$ identical vehicles, located at the depot, must visit the customers. All customers in $\mathcal{N}$ must be visited by a single vehicle; split delivery is not allowed. Each customer $i \in \mathcal{N}$ is characterized by a demand and each vehicle has a capacity as in the classical Capacitated VRP. Vehicles are equipped with batteries of given capacity $B$. Recharge stations can be visited at any time; multiple visits to them is allowed and partial recharge is also allowed. We consider a set of different technologies for battery recharge. For each technology we assume a given recharge speed. When visiting a station, only one of the available technologies can be used.

All vertices $i \in \mathcal{R} \cup \mathcal{N}$ are also characterized by a service time, representing the time taken by pick-up/delivery operations for $i \in \mathcal{N}$ or a fixed time to be spent to set-up the recharge for $i \in \mathcal{R}$. The distance $d_e$ and the travel time are known for each edge $e \in \mathcal{E}$. The energy consumption is assumed to be proportional to the distance through a given coefficient $\pi$. The duration of each route (including service time, travel time and recharge time) is required not to exceed a given limit.

A feasible route must comply with capacity and duration constraints. Furthermore the level of the battery charge must be kept between 0 and $B$ at any time. A set of feasible routes is a feasible solution if all customers are visited once and no more than $V$ vehicles are used. The objective to be optimized is given by the overall recharge cost, consisting of a fixed cost and a variable cost. Since batteries allow for a limited number of recharge cycles during their operational life, we associate a fixed cost $f$ with each recharge operation. The variable cost associated with a recharge operation at any station $i \in \mathcal{R}$ is proportional to the amount of energy recharged, but it also depends on the recharge technology.

We indicate with $\Omega$ the set of all feasible routes. We associate a binary variable $x_r$ with each feasible route $r \in \Omega$: Binary coefficients $y_{ir}$ take value 1 if and only if customer $i \in \mathcal{N}$ is visited along route $r \in \Omega$. We indicate by $c_r$ the cost of each route $r \in \Omega$. With these definitions and notation we obtain the following ILP model (master problem):

\[
\text{minimize } \sum_{r \in \Omega} c_r x_r \tag{1}
\]
\[
\text{s.t. } \sum_{r \in \Omega} y_{ir} x_r \geq 1 \quad \forall i \in \mathcal{N} \tag{2}
\]
\[
\sum_{r \in \Omega} x_r \leq V \tag{3}
\]
\[
x_r \in \{0,1\} \quad \forall r \in \Omega. \tag{4}
\]

At each node of a branch-and-bound tree the linear relaxation of the master problem is
solved by column generation. We indicate by $\lambda_i$ the non-negative dual variables vector corresponding to the covering constraints (2) and by $\mu$ the scalar non-negative dual variable corresponding to constraints (3) restated in $\geq$ form. With this notation, the expression of the reduced cost of a generic column $r$ is

$$\hat{c}_r = c_r - \sum_{i \in \mathcal{N}} \lambda_i y_{ir} + \mu.$$ 

3 The pricing sub-problem

The pricing problem, whose ILP formulation is not reported here for brevity, is a variation of the Orienteering Problem and it requires to find a minimum cost closed walk from the depot to the depot, not visiting any customer vertex more than once and not consuming more than a given amount of available resources (capacity, time and energy). Edges between stations can be traversed more than once. This problem is also a variation of the Resource Constrained Elementary Shortest Path Problem, in which the elementary path constraints are imposed only on a subset of vertices, the resources are partly discrete and partly continuous and one of the resources (energy) is renewable.

3.1 The algorithm

We have devised an exact pricing algorithm based on dynamic programming, where labels are associated with paths emanating from the depot and have the following form:

$$\mathcal{L} = (u, S, \phi, t, \hat{c}, \underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta}),$$

where $u$ is the endpoint of the path different from the depot, $S$ is the set of customer vertices visited along the path, $t$ is the minimum time required to traverse the path, $\hat{c}$ is the minimum reduced cost of the path, $\underline{\Delta}$ and $\overline{\Delta}$ (scalar values) are the minimum and the maximum amount of residual energy that can exist in the battery when the vehicle reaches $u$ from the depot, $\underline{\delta}$ and $\overline{\delta}$ (vectors with one component for each technology) are the lower and upper bounds on the total amount recharged with each technology along the path. For brevity, we indicate by $\mathcal{P}$ the polytope defined by the lower and upper bounds. The information conveyed by $t$, $\hat{c}$, $\underline{\Delta}$ and $\overline{\Delta}$ is indicated for convenience but it can be obtained from the knowledge of $\mathcal{P}$.

Relying upon these definitions we developed and tested a dynamic programming algorithm to price out columns. Besides fathoming dominated states, the algorithm also relies on acceleration techniques such as bounding and state space relaxation.

In our talk we will present computational results obtained on benchmark instances from the literature on the pricing problem for the EVRP.
References


An efficient multi-thread metaheuristic for the CARP

Philippe Lacomme
LIMOS (UMR 6158), Université Clermont Auvergne, 63177 Aubière, France

Christian Prins
ICD-LOSI (UMR 6281), Université de Technologie de Troyes, 10004 Troyes, France
Email: placomme@isima.fr, christian.prins@utt.fr

1 Introduction to CARP and parallel approaches

The NP-hard Capacitated Arc Routing Problem (CARP) consists in servicing a subset of edges with known costs and demands in an undirected network, using identical vehicles based at a single depot, to minimize the total cost of the routes. Applications include for instance urban waste collection and winter gritting. CARP heuristics are surveyed in [1]. Efficient metaheuristics include guided local search [2], memetic algorithms [3,4], ant colonies [5], and GRASP with path relinking [6]. They are all outperformed by two recent memetic/tabu search hybrid approaches HMA1 and HMA2 from Chen et al. [5]. As all these heuristics runs on a single processor, it is interesting to appraise the potential benefits of parallel implementations.

Four main trends can be identified in parallelization. The first one involves clusters of computers and communication protocols like MPI (Message Passing Interface). It is not suited to metaheuristics exchanging a lot of data, unless using expensive high speed networks. The second trend relies on distributed systems from the big data community. One example based on the MapReduce parallel programming model [8] is Hadoop for distributed computations in a batch context. Another example is STORM, a distributed real-time computation system suitable for processing large volumes of high-velocity data in real time environment. The third trend tries to exploit Graphics Processing Units (GPU). Such boards designed for games have a huge number of small processors which can be controlled using the parallel computing platform CUDA (Compute Unified Device Architecture) by NVIDIA. The problem is that their internal structure based on memory blocks makes difficult the parallelization of metaheuristics. Moreover, exchanges with the main processor and the RAM of the PC used slows down execution. In the same vein, NVIDIA has developed a new generation of fast parallel computing architectures, e.g., the Tesla workstations, but they are rather dedicated to professional applications. The last trend, considered in this abstract, is based on multicore PCs which are now widespread but still underused in OR. For instance, the Intel Core i7 6900K has 8 cores, able to run 16 tasks (threads) in parallel using the Intel hyper-threading technology. Compared to GPU, parallelization is less complicated because the threads have a fast and asynchronous access to the RAM, via primitives handled by the operating system and now callable from most programming languages like C++.
2 Proposed parallel MS-ELS metaheuristic

Our metaheuristic (figure) is of type 3 in the classification from Crainic and Toulouse [9], as it consists of several concurrent searches of solution space. Our goal is to get significant speedups via a cooperation requiring a reasonable parallel implementation effort. A good balance between diversification, intensification and cooperation is reached via some key-features:

- Each thread runs a multi-start evolutionary local search (MS-ELS) performing ne iterations from np randomized initial solutions. Iterated local search (ILS) is known to generate new solutions via perturbation and local search. ELS is similar but creates several child-solutions per iteration. By adding restarts, MS-ELS combines in a natural way more ingredients for intensification (local search) and diversification steps (perturbation and restarts). Our parallel MS-ELS (PMS-ELS) for the CARP operates on giant tours decoded by a splitting procedure before calling the local search, like in the memetic algorithms described in [3,4].
- One main thread (dispatcher) launches the MS-ELS threads with a given set of parameters.
- Cooperation is achieved via a simple pool where each thread stores and updates its current best solution. So, the pool contains only p complete solutions if p threads are running.

The efficiency and the degree of synchronization and cooperation critically depend on:

- The dispatcher strategy. We partition the threads in two groups of same size. All threads in the same group receive the same parameters (number of ELS iterations...), except the seed for the random generator which is distinct for each thread, to avoid duplicate trajectories. Moreover, each thread restarts from one initial solution obtained by applying random moves to the current best solution in the pool.
• The synchronization frequency. Due to mechanisms involved in multi-thread programming (mutual exclusion "mutex") locks, semaphores, condition variables, and reader/writer locks), a too frequent access to the pool can strongly degrade running times. In our case, before doing a restart, each thread updates its best solution in the pool with a probability P and only after a minimum number of restarts since the last update.

• The strategy to spread the threads in solution space. As the different MS-ELS are randomized algorithms building independent trajectories from diversified initial solutions, the probability of concentration in the same region of solution space is already low. However, we reduce it even further by forbidding identical solutions (clones) globally, i.e., solutions obtained by all threads must be different. The thread spreading strategy employs a simple but fast hashing technique. Using a hash function h(s) and a large array T of integers we can check T[h(s)] in $O(1)$ to see whether solution s is already known. Like the pool, the hash table T is shared by all threads but its access does not slow the application: the hardware automatically manages mutual exclusions when integers are read or written.

3 Experiments and concluding remarks

Due to lack of space, we provide only a comparison with the current best methods cited at the beginning. More results will be introduced at the conference. The classical CARP benchmarks used are Golden-DeArmon-Baker instances (Gdb), Valencia instances (Val), and Luc Muyllderman's sets C and E (Luc-C and Luc-E). Our PMS-ELS was programmed in C++ and evaluated with 1 to 10 threads on a 3.5 GHz Xeon with 32 GB RAM and Windows 7.

The provisional results in the table below concern the best solution for 10 to 30 runs, depending on the method. The gap of this solution to the best known lower bound is averaged on all instances. Time gives the average time per run in seconds, averaged on all instances and scaled for our PC. Slashes indicate unused benchmarks for some algorithms. We provide for our method the results for one thread (PMS-ELS-IT) and for the best version (Best-PMS-ELS). The best version may require a different number of threads for each benchmark: this number is indicated in brackets after the running time.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Gdb (23)</th>
<th>Val (34)</th>
<th>Egl</th>
<th>Luc-C</th>
<th>Luc-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>7 to 27</td>
<td>25 to 50</td>
<td>77 to 140</td>
<td>32 to 97</td>
<td>26 to 97</td>
</tr>
<tr>
<td>Edges</td>
<td>11 to 55</td>
<td>34 to 97</td>
<td>98 to 190</td>
<td>42 to 140</td>
<td>35 to 142</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runs</th>
<th>Gap</th>
<th>Time</th>
<th>Gap</th>
<th>Time</th>
<th>Gap</th>
<th>Time</th>
<th>Gap</th>
<th>Time</th>
<th>Gap</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAENS [4]</td>
<td>30</td>
<td>0.00</td>
<td>3.6</td>
<td>0.08</td>
<td>38.8</td>
<td>0.40</td>
<td>400.2</td>
<td>0.26</td>
<td>132.9</td>
<td>0.16</td>
<td>129.2</td>
</tr>
<tr>
<td>Ant-CARP-12 [5]</td>
<td>10</td>
<td>0.04</td>
<td>0.7</td>
<td>0.05</td>
<td>5.4</td>
<td>0.50</td>
<td>105.6</td>
<td>0.37</td>
<td>23.8</td>
<td>0.31</td>
<td>23.6</td>
</tr>
<tr>
<td>GRASP [6]</td>
<td>15</td>
<td>0.00</td>
<td>1.1</td>
<td>0.08</td>
<td>19.6</td>
<td>0.40</td>
<td>183.3</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>HMA1 [7]</td>
<td>30</td>
<td>0.00</td>
<td>1.3</td>
<td>0.02</td>
<td>17.2</td>
<td>0.26</td>
<td>226.1</td>
<td>0.25</td>
<td>40.2</td>
<td>0.06</td>
<td>46.4</td>
</tr>
<tr>
<td>HMA2 [7]</td>
<td>30</td>
<td>/</td>
<td>/</td>
<td>0.01</td>
<td>42.0</td>
<td>0.21</td>
<td>734.8</td>
<td>0.23</td>
<td>59.1</td>
<td>0.02</td>
<td>127.0</td>
</tr>
<tr>
<td>PMS-ELS-IT</td>
<td>20</td>
<td>0.00</td>
<td>2.4</td>
<td>0.05</td>
<td>6.8</td>
<td>0.48</td>
<td>71.1</td>
<td>0.23</td>
<td>22.7</td>
<td>0.20</td>
<td>29.7</td>
</tr>
<tr>
<td>Best PMS-ELS</td>
<td>20</td>
<td>0.00</td>
<td>0.5 (8)</td>
<td>0.02</td>
<td>5.3 (10)</td>
<td>0.29</td>
<td>83.5 (8)</td>
<td>0.23</td>
<td>11.3 (4)</td>
<td>0.04</td>
<td>14.4 (6)</td>
</tr>
</tbody>
</table>
Although these preliminary results must be interpreted with caution, our MS-ELS framework seems to be very efficient. Even with one thread, it is never the worst method in terms of cost, while being much faster than the other methods except on the smallest instances Gdb. Ant-CARP-12 [5] is also very fast but displays the worst gap on Egl, Luc-C and Luc-E sets, moreover this is the only algorithm which do not find all optima for Gdb instances. We can note that using more threads reduces the gaps and/or running times.

To conclude, the design of multi-thread algorithms is now possible using standard compilers which provide ad-hoc primitives. Despite this easier access to parallelization, most OR researchers still use single-thread implementations. An exception is represented by MILP solvers like CPLEX, able to use all cores of a PC. This abstract shows that a fairly simple multi-thread metaheuristic can yield promising results for the CARP, even using a few threads. Although no best solution has been improved, a solution quality competing with state of the art methods can be achieved in strongly reduced wall-clock times. This confirms similar improvements reported by Groër for a parallel record-to-record travel method (RTR) for the CVRP [10]. We plan to investigate if other metaheuristic frameworks could give better results, and to develop more efficient cooperation schemes.

References

Clustering Techniques for Very-Large Scale Arc Routing Problems in Curbside Waste Collection

Hani Zbib
Cluster for Operations Research and Logistics,
Department of Economics and Business Economics,
Aarhus University, Aarhus, Denmark
Email: haniz@econ.au.dk

Sanne Wøhlk
Cluster for Operations Research and Logistics,
Department of Economics and Business Economics,
Aarhus University, Aarhus, Denmark

1 Introduction

The No-Split Multi-Compartment Capacitated Arc Routing Problem (NSMC-CARP) is a variant of the CARP that arises in the curbside collection of waste/recyclables from households [1]. Each household has a demand at least one waste fraction, and collection is done on a street level for all households on a street, represented as an edge. The NSMC-CARP is defined on an undirected graph and its objective is to find a set of least cost closed routes that start and end at the depot node, such that all waste fractions of a required edge are serviced by the same vehicle without violating the vehicles’ compartment capacities, and such that each required edge is serviced by one vehicle only. The fleet of collection vehicles is a homogeneous fleet of multi-compartment vehicles. The CARP can be seen as a special case of the NSMC-CARP where there exists one waste fraction.

Kiilerich and Wøhlk [1] presented real life instances from waste collection companies in Denmark for the CARP and the NSMC-CARP, whose sizes (up to 11656 nodes and 12691 edges) are much larger than the previous benchmark instances for the CARP (up to 255 nodes, 375 edges, and 375 required edges for the Brandão and Eglese instances [2]).

The current state-of-the-art metaheuristics in the literature for the CARP, such as the ILS of Prins et al.[3] and the UHGS of Vidal et al.[4, 5], both use the optimal tour splitting algorithm [3] iteratively at the core of their metaheuristics. One run of this algorithm has been shown by Wøhlk and Laporte [6] to take a highly prohibitive run time on the very-large graphs, up to 23 hours for the largest. Therefore, the need for new approaches and
algorithms able to solve very-large problems has become a necessity. Zbib and Wøhlk [7] present a Multi-Move Chain Descent (MMCD) algorithm, an algorithm specifically designed to solve large instances of the CARP and NSMC-CARP, that was able to find the best known solutions thus far for the small to large-scale instances of [1] (up to 4000 required edges), but proved to be too time prohibitive for the very large-scale graphs.

Hence, in order to solve these very large instances, graph aggregation and reduction techniques, such as districting or clustering, need to be applied on the very-large graphs to reduce their sizes to reasonable sizes that allow the different algorithms to give good solutions to the problem in reasonable computational times.

The literature on clustering in a vehicle routing setting is abundant, but very little has been proposed in an arc routing setting. Vidal [5], who presents a structural decomposition of arc routing problems in which decisions about the traversal orientation of required edges are made optimally for each route, briefly explains how clusters can be modeled as part of the solution representation, but does not offer any approaches on how to cluster edges.

We present novel cluster types and clustering techniques in an arc routing setting that are common to real-life street networks. These cluster forms are tailored to be included in the solution representation based on optimal orientations of [5]. The aim of the clustering is to reduce the sizes of the graph considerably to sizes that are computationally unrestrictive. Once the clusters are formed and the graph is reduced, we subsequently solve the problem on the reduced graph using the MMCD algorithm of [7], allowing the algorithm to run on more than one solution within the same allocated time.

2 Clustering Techniques in an Arc Routing Setting

We define a cluster to be formed by a set of required edges, and a set of nodes in the graph which we denote as access nodes that can be used to enter and exit the cluster. With every cluster is associated a traversal cost between each pair of entrance and exit nodes in the set of access nodes. The cost corresponds to the least-cost tour starting at the entrance node, servicing all the required edges in the cluster, and ending at the exit node, including the deadhead incurred while doing so. The cost varies with the type of the cluster. A cluster can only be formed if the total cumulative demand of the cluster for each waste fraction does not exceed a certain portion of the compartment capacity in the vehicle for all fractions. The portion varies based on the cluster type.

In order to reduce the size of the graph in the solution representation with optimal orientations, the number of access nodes as opposed to the number of required edges in the cluster is relevant. Clusters with a small number of access nodes and a high number of required edges reduce the graph significantly more, and our cluster types are designed accordingly. We have developed different types of clusters in an arc routing setting that
arise in different real-life street networks, which we detail here (see figure 1 for examples of the clusters).

The first type of cluster is fishbone clusters. In many urban and suburban areas, residential blocks were designed based on a fishbone structure, with a lot of dead-ends in the fishbone, and where the block is accessed from the main road by one single node. In such a case, a fishbone cluster has one access node acting as an entrance and exit node, and the traversal cost of the cluster consists of double the cost of all edges in the cluster.

The second type of cluster is chain clusters, which consists of clustering a chain of edges connected to each other by nodes with degree two, with the ends of the chain being connected to nodes with a degree larger than two. This could be due to two cases. The first case arises in rural areas, where only some segments of a long street require to be serviced, so the street is represented in the graph as an alternation of required and non-required edges connected by degree two nodes. The second case arises as a consequence of fishbone clusters, where a node that had a degree higher than two in the original graph, now has a degree of two in the reduced graph, and the set of edges in that chain (including the fishbone cluster) can be clustered into a chain cluster. Its traversal cost consists of the sum of the costs of all its edges, and the cost of all fishbone clusters added to it.

The third type of cluster is triangle clusters, which consist of clustering a set of three required edges sharing two by two a node each, forming a triangle between them, with each of the three nodes of the triangle having a degree of 3 or more. This type of cluster occurs naturally in certain rural/suburban areas, where one road splits into two, both leading to different exits on the same third road. On the other hand, due to chain clustering, many such triangles might form with one (or more) of the three edges being a chain cluster. Its traversal cost consists of the sum of the cost of all edges and chain clusters forming it.

More generally, if a cluster is not a special type of cluster like fishbone, chain, or triangle clusters, but is rather formed by a set of closely located edges, it is denoted as a general cluster. The optimal traversal cost between all pairs of access nodes can be computed by solving a variant of the Rural Postman Problem (RPP) for each pair of access nodes, where the start and end nodes of the RPP tour are pre-defined.

We also define other types of intermediate clusters that arise from the clustering of different clusters together, such as cycle clusters, that we do not detail here.
3 Solution approach

Our solution approach consists of two phases: an initial Clustering Phase, followed by a Routing Phase to solve the resulting problem on the reduced graph. We have developed clustering algorithms for the different types of clusters, and used these algorithms sequentially in an initial Clustering Phase, reducing the graph significantly. According to our preliminary testing on the CARP and NSMC-CARP graphs, the usage of fishbone, chain, and triangle clusters alone would lead to 25-75% reduction in the number of elements in the graphs. Following the Clustering Phase, the Routing Phase consists of solving the resulting problem on the reduced graph using the Multi-Move Chain Descent Algorithm [7], which has been modified to handle clusters. The clustering positively affects the run time of the algorithm, allowing it to run for a higher number of iterations than it could with the entirety of the graphs.

References


The Drone Arc Routing Problem

James F. Campbell
College of Business Administration,
University of Missouri-St. Louis, St. Louis, USA

Isaac Plana
Departamento de Matemáticas para la Economía y la Empresa,
University of Valencia, Valencia, Spain

José M. Sanchis
Departamento de Matemática Aplicada,
Technical University of Valencia, Valencia, Spain

Ángel Corberán
Departamento de Estadística e Investigación Operativa
University of Valencia, Valencia, Spain
Email: angel.corberan@uv.es

1 Introduction

The research on routing for drones (or Unmanned Aerial Vehicles) has increased recently and new applications are found everyday. The use of drones in routing applications leads to a better service, since they allow high speed deliveries, reduced costs, and safety improvements by removing human operators. Most of these applications are related to the delivery of small commodities to a set of discrete customers, which can be modeled as node routing problems. However, there are other applications that can be better modeled as arc routing problems (ARPs), as, for example, those related to sensing and surveillance of roads, pipelines, street networks, etc. There are two main characteristics that set these problems apart from other arc routing problems. The first one is that the drones combine traveling on a network to service a subset of edges, with traveling directly between two points without following the edges of the network. The second one is that drones may start and end the service of an edge at any point of it.

Although there is an extensive Literature on node routing problems with drones, only a few papers dealing with ARPs with drones have been published. Among them, we can cite those by Oh et al. [1, 2], Dille and Singh [3], and Chow [4].

In the problem that we are studying here, we consider a set of homogeneous drones
that must jointly service a set of edges of a network. Drones start and end at a specified vertex, the depot, and have a route length limit (or distance or time) that applies to the total route, whether servicing an edge on the network or traveling directly “off the network”. We assume that the usage of the capacity depends on the activity, as servicing an edge may involve more intensive use of the battery (to record data or to send data back) than traveling off the network when not servicing. Although in real-life cases the capacity usage rate may depend on the speed of travel, we will assume the speed to be constant and therefore the capacity usage will depend only on the distance traveled. In a first step, we will assume here that the drones cannot service only part of an edge.

2 The Drone Arc Routing Problem

Given a set $E_R$ of required edges that need service, let $V$ be the set of vertices incident with edges in $E_R$ plus one additional vertex representing the depot (if it is not incident with a required edge). Let us consider the complete graph $G = (V, E)$. Note that $E_R \subseteq E$. There is a non-negative cost $c_e$ associated with the traversal of each edge $e \in E$. Additionally, for each required edge $e \in E_R$ there is another non-negative cost $\bar{c}_e \geq c_e$ associated with the traversal of the edge while servicing it. We call DARP solution to any set of $K$ tours (closed walks starting and ending at the depot) such that

(1) each required edge is serviced once by one tour, and

(2) the cost of each tour is less than or equal to a limit value $L$.

The Drone Arc Routing Problem, DARP, consists of finding a minimum cost DARP solution.

The following results illustrate the advantage of using drones over terrestrial vehicles:

**Proposition 2.1** If $L = \infty$ and $G_R = (V, E_R)$ is connected, then the cost of the optimal route servicing all the required edges by a truck (assuming that the cost of deadheading an edge is the same as the cost of traversing and servicing it) is at most twice the cost of the optimal route performed by a drone, and this value is attainable.

**Proposition 2.2** If $L = \infty$ and $G_R = (V, E_R)$ is not connected, then the ratio between the cost of the optimal route by a truck and the one by a drone can be as large as desired.

3 Formulation

In order to formulate the problem, we define the following variables.
For each edge \( e = (i, j) \in E \) we define \( 2K \) variables \( x_{ij}^k, x_{ji}^k \) which take the value 1 if edge \( e \) is traversed (without service) from \( i \) to \( j \) or from \( j \) to \( i \), respectively, by drone \( k \), and 0 otherwise. In addition, for each edge \( e = (i, j) \in E_R \) we define \( 2K \) variables \( y_{ij}^k, y_{ji}^k \) which take the value 1 if edge \( e \) is traversed and serviced from \( i \) to \( j \) or from \( j \) to \( i \) by drone \( k \), and 0 otherwise.

Given two node subsets \( S, S' \subseteq V \), \((S : S')\) denotes the set of edges with one end-point in \( S \) and the other in \( S' \). Given a node subset, \( S \subseteq V \), let us denote \( \delta(S) = (S : V \setminus S) \) and let \( E(S) = \{(i, j) \in E : i, j \in S\} \) be the set of edges with both endpoints in \( S \). The previous sets referred to the required edges are denoted by \( \delta_R(S) \), \( E_R(S) \) and \((S : S')_R \). Finally, for any subset \( F \subseteq E \), \( x_k(F) \) denotes \( \sum_{(i,j) \in F} (x_{ij}^k + x_{ji}^k) \) (similarly \( y_k(F) \), for \( F \subseteq E_R \)).

We propose the following formulation for the DARP:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{k=1}^{K} \sum_{e=(i,j) \in E} c_e(x_{ij}^k + x_{ji}^k) \\
\text{s.t.} & \quad \sum_{(i,j) \in \delta(i)} (x_{ij}^k - x_{ji}^k) + \sum_{(i,j) \in \delta_R(i)} (y_{ij}^k - y_{ji}^k) = 0, \quad \forall i \in V, \forall k \tag{1}
\end{align*}
\]

\[
x^k(\delta(S)) + y^k(\delta_R(S)) \geq 2(y_{ij}^k + y_{ji}^k), \quad \forall S \subset V \setminus \{1\}, \forall (i, j) \in E_R(S), \forall k \tag{2}
\]

\[
\sum_{k=1}^K (y_{ij}^k + y_{ji}^k) = 1, \quad \forall (i, j) \in E_R \tag{3}
\]

\[
\sum_{e=(i,j) \in E} c_e(x_{ij}^k + x_{ji}^k) + \sum_{e=(i,j) \in E_R} \bar{c}_e(y_{ij}^k + y_{ji}^k) \leq L, \quad \forall k \tag{4}
\]

\[
x_{ij}^k, x_{ji}^k \in \{0, 1\}, \quad \forall (i, j) \in E, \forall k \tag{5}
\]

\[
y_{ij}^k, y_{ji}^k \in \{0, 1\}, \quad \forall (i, j) \in E_R \forall k \tag{6}
\]

Symmetry equations (1) force each drone route to be symmetric, while connectivity inequalities (2) ensure that each drone route connects the edges it services and the depot. Equations (3) assure that each required edge is serviced by exactly one drone. Inequalities (4) imply that each cost drone tour is not longer than \( L \).

### 3.1 Other valid inequalities

The previous formulation does not guarantee the parity of the fractional solutions. Some of these solutions can be avoided by adding the following valid inequalities. The \( R \)-odd cut inequalities are:

\[
\sum_{k=1}^K x^k(\delta(S)) \geq 1, \quad \forall S \subset V \text{ such that } |\delta_R(S)| \text{ is odd} \tag{7}
\]

A generalization of the \( R \)-odd cut inequalities associated with a single tour is the following. Let \( \delta(S) \) be an edge cutset on \( G \) and let \( F \subset \delta_R(S) \) be a subset of required edges with \( |F| \) odd. For each drone \( k \), the following parity inequality is valid for the DARP:
\[ x^k(\delta(S)) + y^k(\delta_R(S) \setminus F) \geq y^k(F) - |F| + 1 \] (8)

Note that, if \( F = \delta_R(S) \) and we add the \( K \) inequalities (8) we obtain the above \( R \)-odd cut inequalities (7).

Inequalities (8) can be generalized from a single drone to several ones. Let \( \delta(S) \) be an edge cutset on \( G \) and let \( F \subset \delta_R(S) \) be a subset of required edges with \( |F| \) odd. For each subset of \( P < K \) drones \( \{k_1, k_2, \ldots, k_P\} \), the following \( P \)-aggregate parity inequality is valid for the DARP:

\[ \sum_{i=1}^{P} (x^{k_i}(\delta(S)) + y^{k_i}(\delta_R(S) \setminus F)) \geq \sum_{i=1}^{P} y^{k_i}(F) - |F| + 1 \] (9)

Note that if \( |F| \geq 3 \), inequalities (9) are not dominated by (8).

4 Branch-and-cut algorithm

In order to solve the DARP to optimality, we are implementing a branch-and-cut algorithm based on the previous inequalities that incorporates separation algorithms for inequalities (2), (7), (8) and (9). Computational results on instances randomly generated on grids simulating street networks will be presented at the conference.

References


Multimodal capacity planning with uncertainty on contract fulfillment and suppliers reliability

Mariangela Rosano
DAUIN - Politecnico di Torino, Turin, Italy
Email: mariangela.rosano@polito.it

Guido Perboli
DAUIN - Politecnico di Torino, Turin, Italy
CIRRELT, Montréal, Canada

Teodor Gabriel Crainic
Université du Québec à Montréal, Montréal, Canada
CIRRELT, Montréal, Canada

Walter Rei
Université du Québec à Montréal, Montréal, Canada
CIRRELT, Montréal, Canada

1 Introduction

Contracting with carriers or third-party logistics providers (3PLs) is a way for the shipping companies to face the logistic capacity planning. However, because of their medium-term nature, such contracts are concluded in an uncertain environment. Contrarily to the rich literature on capacity planning problems, few and recent studies address uncertainty in both supply and demand. In fact, according to [1], the current literature deals partially with the requirements of capacity planning, while is focused on operational issues. This paper attempts to fill this gap, proposing the following contributions. First, to investigate the capacity planning problem by taking into account the different sources of uncertainty that affect the contract fulfillment. Second, extend the analysis mentioned above, by considering the supplier selection problem and thus, introducing the uncertainty related to the supplier reliability. To do so, we propose a modeling framework that takes the form of two-stage stochastic programming with recourse bin packing problem. Extensive numerical experiments are conducted focusing on a City Logistics environment.
2 Problem setting

We investigate the logistics capacity planning problem with uncertainty on contract fulfillment arising in the context of the City Logistics. In particular, as the first contribution, we address the tactical decision of determining the quantity of capacity that a shipping company has to contract in advance at the best rate from a single and known 3PL, to procure transportation or warehousing capacity for the next planning period. This decision entails negotiations with the 3PL to define the capacity booked in advance and the clauses concerning additional logistics services (e.g., storage, bin operations, etc.) to ship or store the estimated demand for goods to deliver. Due to the medium-term nature of contracts, when tactical decisions are made, the shipping company does not have detailed knowledge of many decision parameters, which become known during the operations. For covering a wide range of situations, our model includes uncertainty on different decision parameters in both supply and demand, i.e., future demand, number, characteristics, and availability of the resources providing the required capacity each time the contract is applied, the future availability and cost of additional capacity when operations start. At each realization of the plan, information concerning these parameters becomes available and unfavorable situations (e.g., demand fluctuation, different characteristics of goods, leftovers and mechanical failures of vehicles) could make the booked capacity not available or not sufficient at shipping day, compromising contract fulfillment and making needed adjustments in the capacity plan. We model this problem by proposing a two-stage with recourse stochastic bin packing formulation named *Stochastic Variable Cost and Size Bin Packing Problem with Loss Capacity*. It generalizes prior work by [1] through the possibility that actual volumes of the contracted resources be different from those planned and thus, stochastic. The first stage concerns the tactical decision as the selection a priori of the capacity to ship and store the estimated demand for goods called items $i$. This capacity is expressed in terms of bins $y$, which are characterized by a specific type $t \in T$, a volume $V^t$ and a fixed cost $f^t$ defined by the contract according to the bin type. The objective function (1) minimizes the sum of the total fixed cost of the capacity plan and the expected cost associated with the extra capacity added during the operations. We then introduce constraints (2) and (3) to ensure respectively an order in the selection of bins of type $t \in T$ and the integrality requirements on $y$. The second stage refers to the operational decisions concerning the recourse actions to react to an unfavorable situation (i.e., random event $\omega \in \Omega$), and the assignment of items to available bins. These actions concern the acquisition of additional capacity (extra bins) in the so-called spot market, when the demand is revealed. The extra bins must be purchased at a higher cost $g^T(\omega)$ than the fare negotiated initially. Moreover, an extra cost $c^t$ is required to rearrange the loads. The objective function (4) minimizes the sum of the costs paid for the recourse actions. Constraint (5) and, (6) and (7) ensures respectively that each item is packed in a single bin and that the total volume of items
packed in each bin does not exceed the actual volume of the bin. Finally, constraint (8) imposes the integrality requirements on all second-stage variables.

The two-stage model may be formulated as:

$$\begin{align*}
\min_y & \sum_{t \in T} \sum_{j \in J^t} f^t y^t_j + E_\xi [Q(y, \xi(\omega))] \\
\text{s.t.} & \quad y^t_j \geq y^t_{j+1}, \quad \forall t \in T, j = 1, \ldots, |J^t| - 1, \\
& \quad y^t_j \in \{0, 1\}, \quad \forall t \in T, j \in J^t.
\end{align*}$$

(1)

(2)

(3)

The function $Q(y, \xi(\omega))$ then becomes:

$$Q(y, \xi(\omega)) = \min_{y,z} \sum_{\tau \in T} \sum_{k \in K(\tau)} g^\tau(\omega)z^\tau_k(\omega) + \sum_{t \in T} \sum_{j \in J^t} c^t(V^t - V^t_j(\omega))y^t_j$$

(4)

s.t.

$$\begin{align*}
\sum_{j \in J} x_{ij}(\omega) + \sum_{k \in K(\omega)} x_{ik}(\omega) &= 1, \quad \forall i \in I(\omega), \\
\sum_{i \in I(\omega)} v_i(\omega)x_{ij}(\omega) &\leq V^t_j(\omega)y^t_j, \quad \forall t \in T, j \in J^t, \\
\sum_{i \in I(\omega)} v_i(\omega)x_{ik}(\omega) &\leq V^\tau z^\tau_k(\omega), \quad \forall \tau \in T, k \in K(\omega), \\
x_{ij}(\omega); x_{ik}(\omega); z^\tau_k(\omega) &\in \{0, 1\}, \quad \forall i \in I(\omega), j \in J, k \in K(\omega), \tau \in T.
\end{align*}$$

(5)

(6)

(7)

(8)

Regarding the solution strategy, we adopted a heuristic based on the Progressive Hedging method [1, 2]. It applies a Scenario Decomposition technique that separates the stochastic problem by scenario subproblems, reducing the computational efforts.

Our second contribution aims to add a further investigation on the tactical capacity-planning decisions in the presence of supplier selection problem. Indeed, the need of supply chain risk mitigation is a key issue in City Logistics, leading the shipping companies to reconsider their reliance on single sourcing strategies and embrace multiple vendors (3PLs) sourcing. However, on the one hand having more than one 3PL reduces the capacity risks and the 3PL’s bargaining power, on the other hand, this strategy could increase other contractual, legal or managerial issues that affect the service quality. We consider the tactical capacity planning problem in which the shipping company evaluates, selects and contracts with more than one supplier. Each of these 3PLs is characterized by a cost to activate a business activity, a probability to fail the delivery and a reliability factor. This measure represents the shipping company’s belief about the 3PL reliability to fulfill the contract, according to internal and external information (e.g., historical data), biasing the supplier selection. Thus, the supplier selection problem introduces in the above-presented model additional challenges, decision variables and sources of uncertainty. In fact, the new model decides in the first stage which suppliers to use (by activating or not their contracts) and how many bins per each type to book in a contract to satisfy the demand. At the second stage, it reacts to the random events (i.e., unfavorable situations or the
supplier unreliability) by deciding if and how to adjust the plan. In fact, in addition to the spot market, new recourse actions can be considered, such as further negotiations of extra capacity with selected 3PLs. Finally, note that the supplier reliability affects the bins availability in the second stage and introduces a fee (named compensation cost) that the supplier has to refund to the shipper for the unavailability of part of the bins’ volume and the costs due to the reloading operations.

3 Experimental plan and computational results

An extensive set of experiments, using the case of an express courier in a City Logistics environment, has been conducted to determine the structure of the capacity plan when we consider uncertainty on contract fulfillment and supplier reliability. In particular, starting from the basic instances set generated for different bin packing problem variants, we create a new set of instances to explore the structure of the capacity planning for a wide range of configurations. Being an ongoing project and given the length restrictions, we present the preliminary results while we will discuss the model including supplier reliability and the complete results at the conference. One of the most relevant findings is that the structure of the capacity plan is affected by some parameters of the problem as the probability of a reduction of available capacity and the type and entity of the losses. If only a few types of bins are available on the spot market, if the availability of extra bins at the shipping day is limited, and if the losses of available volume are uniformly distributed among the bins, the firm should book almost all the capacity (between 60% and 83%) it will need for the planning horizon in advance. In particular, if there is a high probability of losing a large amount of capacity, the firm should book more capacity than needed. On the contrary, if there is a high probability of losing a large amount of capacity but the availability of extra bins is not limited, no capacity should be booked in advance. In this situation, the firm does not make a capacity plan, preferring a wait and see approach and thus, to purchase the necessary capacity at a premium price at the shipping date. Finally, if capacity losses are localized in a few bins that become unusable, the firm should book in advance the same capacity of the case with no losses, using more than two types of bins.

References


Routing of small parcels in a crowd-sourced network

Raviv Tal, Tenzer Z. Eyal
Department of Industrial Engineering, Tel Aviv University, Israel
Email: talraviv@eng.tau.ac.il
February 10, 2018

1 Introduction and problem definition

The rapid growth of e-commerce has caused a significant increase in the volume that passes through the parcel delivery industry. We envision a logistic process for delivering small parcels by regular commuters who subscribe as occasional curriers (OCs). Each OC is ready to transfer parcels between locations along her predetermined route in exchange for a monetary compensation that is proportional to her efforts and time loss. E.g., €1.00 for each stop and €0.25 for handling each parcel. A network of automatic service points (SPs) is deployed. The SPs are used as locations where parcels can be dropped-off by their senders and picked up by their recipients. Also, the SPs can serve as intermediate transfer points for the OCs.

Similar schemes were presented by [1] who proposed using occasional drivers to deliver parcels when there is a good match between his itinerary and the origin and destination of the driver. [2] introduced a scheme based on journeys made by taxi drivers, where a parcel is picked before and dropped-off after passengers trips, in nearby locations.

The operation of such a logistic network raises many challenges: On-line decisions about which OC should be offered to stop when passing by an SP and which parcels she should be offered to carry along her journey should be made. This abstract focuses on finding an optimal policy for these decisions. The policy optimization problem is formulated as a stochastic dynamic problem (SDP) and an efficient algorithm to solve this SDP is presented.

We consider a system with $n$ SPs. A set of OC journeys is given. Each journey is a sequence of SPs. There is a known Poisson process that governs the arrival of OCs to the beginning of each such journey. We note that this rate refers to the fraction of the active OCs that are ready to transfer parcels at a particular time. The travel time between a pair of SPs is denoted by $t_{ij}$. $C_i$ denotes the set of SP sequences suffixes that pass by $SP_i$ (but not end there). $C^k_i$ denotes the $k^{th}$ such sequence and it begins with $SP_i$. Note that $C^k_i$ is a result of the unification of all the suffixes of journeys that pass via $SP_i$ and continue along the same sequence of SPs. The rate of OCs that follows the suffix $C^k_i$ is denoted by $\lambda^k_i$. Finally $\lambda_i = \sum_k \lambda^k_i$ is the arrival rate of OCs at $SP_i$. Each parcel in the system is characterized by its origin and destination SPs.

Since the goal is to deliver the parcels as quickly as possible, the system is charged for any unit of time in which a parcel is not yet at its destination. We refer to this cost as “holding cost” and denote it by $h$. The goal is to assign parcels to OCs such that the total sum of the compensation paid to the OCs and the holding cost is minimized.

As a first step, we make the following assumptions: 1) An OC can carry an arbitrary number of parcels and stop at an arbitrary number of SPs along her journey. 2) the capacity of the SPs is not binding. 3) We ignore the stopping cost of the OCs and minimize only the handling and holding costs. Under these assumptions, the decisions described above can be made separately for each parcel based on its current location and destination only. These assumptions are later relaxed using heuristic considerations.
2 Solution method

The policy optimization problem described above is formulated as a stochastic dynamic program and solved efficiently using a Dijkstra-like algorithm. Since the problem is separable by parcels, we solve the optimal decision for a parcel with a given destination that is currently reside at any of the other SPs. The state space consists of two sets of states. The first represents situations in which the parcel is in \( SP_i \) and no OC passes by it. The second represents situations in which the parcel is in an SP that is visited by an OC with a known suffix. We denote the states of the first type by \( S_0^i \) and of the second by \( S_k^i \) where the index \( i \) refers to the current location of the parcel \( (SP_i) \) and the index \( k \) to the the sequence suffix. The decision space is a singleton for the \( S_0^i \) states (just wait for an OC to arrive) and \( C_k^i \) for the \( S_k^i \) states, which is to select for which SP the parcel should be carried. Recall that \( SP_i \in C_k^i \), which means that it is always possible to leave the parcel in the current SP. The Bellman equations are:

\[
v(S_t^k) = \min_{j \in C_k^i} \{ht_{ij} + r_{i,j} + v(S_j^0)\}, \tag{1}\]

and

\[
v(S_t^0) = \min_{K \subset \{1, \ldots, |C_t|\}} \left\{ \frac{h}{\sum_{K \in K} \lambda^k_i} + \sum_{k \in K} \frac{\lambda^k_i}{\sum_{l \in K} \lambda^l_i} v(S_t^k) \right\}. \tag{2}\]

Assuming, without loss of generality, that the destination of the parcel is \( SP_n \), the boundary condition can be stated as \( v(S_n^0) = 0 \). This represents the fact that there is no cost that is associated with a parcel that is already located at its destination.

The challenges in calculating the above Bellman equation are twofold. First, it is not clear in what order the equations can be evaluated, such that all the relevant values of other states are available when needed. Second, the evaluation of (2) requires optimizing over the exponentially large collection of all the subsets of \( \{1, \ldots, |C_t|\} \). We overcome the first challenge by keeping a set of SPs with the smallest values. At each iteration, an SPs, that were not added yet, with the smallest value is added to this set. Thus, at each iteration there is at least one \( S_0^i \) state that can be evaluated. Equation (2) is solved in polynomial time by exploiting the observation that the subset \( K \) that minimize the value consists of some \( k \) sequence suffixes with the smallest \( v(S_k^i) \) values among the members of \( C_t \). Thus one can find the optimal \( K \) by sorting the sequences in \( C_t \) in increasing order of \( v(S_k^i) \) and check all the prefixes. Algorithm 1 implements the above idea.

In lines 1-3 of Algorithm 1 the values of the SP states are initialized and the set \( Q \), of the yet to be calculated SP states is constructed. The main loop of the algorithm repeats a set of operations described below until \( Q \) is empty, that is the values of all the SP states are known. In line 5, the values of all the OC states are calculated based on the current value of the SP states. Next, in lines 6-9, the tentative values of all the unknown SP states are calculated based on the current values of the OC states. In line 10-11, the SP with the smallest tentative value among the SP states with unknown values is selected and removed from the set of unknown SP-states. In line 12, this tentative value is registered as the actual value of the selected SP state.

The complexity of the algorithm is dominated by the loop in lines 4-13 that iterates through the \( n \) SP states and the nested loop in lines 6-9 that iterates through all the OC states (equivalent of OC suffixes) and sort them. Let us denote the total number of such suffixes by \( m = \sum_i |C_i| \). Thus the overall complexity is \( O(nm \log m) \). Recall that the algorithm calculates the value of all the states for a given destination. Therefore, calculating the value table for the all the possible destinations can be accomplished in \( O(n^2m \log m) \).
Let $v(S_0^n) = 0$;

Let $Q = \{1, \ldots, N\} \setminus \{n\}$;

for $i \in Q$ do Let $v(S_i^n) = \infty$;

while $Q \neq \emptyset$ do

for $i \in Q, k = 1, \ldots, |C_i|$ do $v'(S_{ik}^k) = \min_{j \in C_i^k} \{ht_{ij} + r_{ij} + v(S_0^j)\}$;

for $i \in Q$ do Sort the sequences in $C_i$ increasing order of $v'(S_{ik}^k)$;

Let $v'(S_0^i) = \min_{k' \in \{1, \ldots, |C_i|\}} \left\{ \frac{h}{\sum_{k=1}^{k'} \lambda_i^{[k]}} + \frac{\lambda_i^{[k]}}{\sum_{l=1}^{k'} \lambda_i^{[k]}} v'(S_{ik}^k) \right\}$;

Let $i_{\text{min}} = \arg\min_{i \in Q} v'(S_0^i)$;

Let $Q = Q \setminus \{i_{\text{min}}\}$;

Let $v(S_0^{i_{\text{min}}}) = v'(S_0^{i_{\text{min}}})$;

end

Algorithm 1: Dijkstra like algorithm for the parcel routing problem

3 Results and conclusions

The parcel routing policy derived from the above dynamic program was tested in simulation based on realistic data about movements of people and parcels with 50 SPs located in Tel Aviv area. The results demonstrate that such a scheme can be used to provide a reliable next day delivery service at a modest cost (stems from the payments to the OCs).

In an ongoing study, we are checking the sensitivity of the scheme to our simplifying assumptions about the capacity constraints and the willingness of the OCs to serve all the requests. In particular, we run an extensive simulation study to test a reward mechanism that includes compensation both for handling each parcel (as in the DP model) and for each stop made by the OC. The capacity constraint and the more complex OC reward scheme require optimizing offers made to the OCs to allow better consolidation and a smaller number of stops. Our preliminary simulation study indicates that it is possible to meet a daily demand for 16000 parcels with daily 8000 OC journeys, which represent a tiny fraction of the number of journey made in the city. Assuming that the vehicle capacity of each active OC is 50 parcels and the capacity of the SPs is 400 parcels our simulation demonstrate the capability of our logistic model to provide a reliable same day delivery service in a fraction of the cost currently charged in the market. This small cost is achieved while rewarding the OCs for their effort and loss of time with incentives that are more than twice the average hourly wage. Therefore we suggest that such a crowd-sourced parcel delivery scheme has a potential to economize and reduce the ecological footprint of the small parcel delivery industry.

References


1 Introduction

The Capacitated Supplier Selection problem with Total Quantity Discount policy and Activation Costs under Demand Uncertainty (CTQD-AC$_{ud}$) models the medium and long-term procurement setting of a company that needs to purchase, from different suppliers, different products for its business while minimizing the overall purchasing costs. Let $M$ be the set of suppliers and $K$ be the set of products. Each product $k \in K$ can be purchased in a subset $M_k \subseteq M$ of suppliers at a price $f_{ik}$ and respecting an availability $q_{ik}$. A demand, unknown a priori, has to be satisfied for each product. More precisely, for each demand of a product $k \in K$, an estimated deterministic component $d_k$ and a stochastic oscillation $\hat{d}_k(\xi)$, where $\xi$ is a stochastic variable. As an incentive for the buyer to purchase more, the suppliers propose a total quantity discount (TQD) policy, i.e., each supplier $i \in M$ defines a set $R_i = \{1, \ldots, r_i\}$ of $r_i$ consecutive and non-overlapping quantity intervals $[l_{ir}, u_{ir}]$ associated with a discount rate $\delta_{ir} \in [0, 1)$ such that $\delta_{i,r+1} \geq \delta_{ir} \quad r = 1, \ldots, r_i - 1$. Then, for each supplier $i \in M$, the discount rate $\delta_{ir}$ is applied to the total purchase cost if the total quantity purchased lies in the interval $r \in R_i$, i.e., is greater than or equal to $l_{ir}$ and less than or equal to $u_{ir}$. However, the buyer can benefit from the discounts of a supplier $i \in M$ only by paying a fixed fee $a_i$ required to activate the underlying business activity.

The CTQD-AC$_{ud}$, recently introduced in [1], has been shown to bring great advantages to companies when addressed through Stochastic Programming (SP) techniques that explicit consider uncertain parameters instead of estimating their expected values. However, SP models are very hard to solve (mainly as the number of scenarios increases) and, up today, no efficient methods exist to deal with the CTQD-AC$_{ud}$. To overcome this drawback, we propose a Progressive Hedging-based heuristic approach exploiting its SP formulation.
1.1 Stochastic Programming formulation

The CTQD-AC\textsubscript{ud} can be modeled as two-stage SP formulation [1]. The first stage is about which suppliers to select at the beginning of the procurement period, how much we expect to purchase from each supplier and, consequently, in which discount interval we expect the total quantity of products purchased lies. The second-stage recourse actions, related to the operative purchasing, consist in modifying the purchased quantities within the intervals locked at the first stage, or in purchasing a quantity of product \( k \) outside from the selected suppliers (i.e., buy in the so-called spot-market) at a penalty price \( g_k \).

In this work, we propose a formulation (equivalent to the above two-stage decomposition) that explicitly considers a set \( S \) of scenarios to approximate the probability distribution of the stochastic variables \( \hat{d}_k \). More precisely, each scenario \( s \in S \) is associated with a realization of the demand oscillation \( \hat{d}_k^s \) that occurs with probability \( p^s \). Let us also define, for each scenario \( s \in S \), the following variables:

- \( x_i^s \in \{0, 1\} \) is a binary variable taking value 1 if a purchasing contract is activated with supplier \( i \in M \), and 0 otherwise;
- \( z_{ikr}^s \geq 0 \) represents the amount of product \( k \in K \) that we expect to purchase from supplier \( i \in M \) in interval \( r \in R_i \);
- \( y_{ir}^s \in \{0, 1\} \) is a binary variable taking value 1 if the total products quantity we expect to purchase from supplier \( i \in M \) lies in the discount interval \( [l_{ir}, u_{ir}] \) with \( r \in R_i \), and 0 otherwise;
- \( Z_{ikr}^s \) is a free variable representing the variation in purchased quantity, with respect to the expectation \( z_{ikr}^s \), of product \( k \in K \) from supplier \( i \in M \) in interval \( r \in R_i \);
- \( W_k^s \geq 0 \) represents the quantity of product \( k \in K \) that has to be purchased in the spot-market.

Then, the CTQD-AC\textsubscript{ud} problem is as follows:

\[
\begin{align*}
\min & \quad \sum_{s \in S} p^s \left[ \sum_{i \in M} a_i x_i^s + \sum_{k \in K} \sum_{i \in M_k} \sum_{r \in R_i} (1 - \delta_{ir}) f_{ik} (z_{ikr}^s + Z_{ikr}^s) + \sum_{k \in K} g_k W_k^s \right] \\
\text{subject to} & \quad \sum_{i \in M_k} \sum_{r \in R_i} z_{ikr}^s \geq d_k \quad k \in K, s \in S \\
& \quad \sum_{r \in R_i} z_{ikr}^s \leq q_{ik} \quad k \in K, i \in M_k, s \in S \\
& \quad l_{ir} y_{ir}^s \leq \sum_{k \in K} z_{ikr}^s \leq u_{ir} y_{ir}^s \quad i \in M, r \in R_i, s \in S \\
& \quad \sum_{r \in R_i} y_{ir}^s \leq x_i^s \quad i \in M, s \in S \\
& \quad \sum_{i \in M_k} \sum_{r \in R_i} (z_{ikr}^s + Z_{ikr}^s) + W_k^s \geq d_k + \hat{d}_k^s \quad k \in K, s \in S \\
& \quad \sum_{r \in R_i} (z_{ikr}^s + Z_{ikr}^s) \leq q_{ik} \quad k \in K, i \in M_k, s \in S \\
& \quad l_{ir} y_{ir}^s \leq \sum_{k \in K} (z_{ikr}^s + Z_{ikr}^s) \leq u_{ir} y_{ir}^s \quad i \in M, r \in R_i, s \in S \\
& \quad z_{ikr}^s + Z_{ikr}^s \geq 0 \quad k \in K, i \in M_k, r \in R_i, s \in S
\end{align*}
\]
Objective function (1) minimizes of the sum of activation and purchasing costs, weighted over all the scenarios by their probability. Constraints (2) ensure that the demand is satisfied for each product, whereas constraints (3) state that it is not possible to purchase from a supplier a quantity of a product larger than the availability. Constraints (4) define interval bounds for each supplier. Constraints (5) guarantee that at most one interval for each selected supplier is active, and that no intervals are active if the supplier is not selected. Constraints (6), (7), and (8) have the same meaning of the constraints (2), (3), and (4), respectively, but also consider the recourse decisions on the quantities purchased ($Z$-variables). Moreover, constraints (6) allow to satisfy part of the product demand by using the spot market ($W$-variables). Constraints (9) deny to purchase a negative quantity of product and, consequently, to change the discount interval chosen at the first-stage. Equations (10)–(12) are the non-anticipatity constraints forcing every scenario-based solutions to share the same first-stage decisions (i.e., to be implementable).

2 A heuristic Progressive Hedging for the CTQD-AC	extsubscript{ud}

Progressive Hedging (PH) is a decomposition-based algorithm proposed in for SP models [3]. Briefly, the PH first decomposes the problem over the scenarios by relaxing in a Lagrangean fashion the complicating constraints, i.e., equations (10)–(12) in our case. Then, at each iteration, the optimal solutions of all the single scenario problems and a temporarily global solution (TGS) for the complete problem are calculated by using some aggregation operators (the expectation function is used in our case). The algorithm stops when a complete consensus on the first-stage decisions over all the scenarios is met (i.e., when the TGS becomes implementable) or when some other termination criteria are satisfied (maximum CPU time, maximum number of iterations, and so on), otherwise it adjusts the Lagrangean multipliers of each single scenario problems and iterates again. Unfortunately, the PH has been proved to converge to the optimal solution only for continuous linear programs, hence, at the end of the algorithm, we optimally solve the original model (2)–(12) reduced in complexity by fixing those variables for which the consensus is met according to the best TGS found. This actually makes our algorithm an heuristic approach [2].

To enhance the basic PH implementation in terms of convergence and solution quality, we have developed several additional strategies: a) we pursue, during the PH, the consensus of a restricted set of first-stage variables (i.e., only the binary ones) and we complete the solution by solving the resulting linear program at the end; b) we run a primal heuristics based on the current non-implementable TGS to generate feasible solutions during
the procedure, and not only at the end; c) at each PH iteration, we compute the set of single scenario subproblems in parallel by means of multi-threading.

3 Results and conclusions

The PH has been implemented in C++ and compared to Cplex 12.7 solver on benchmark instances with up to 20 suppliers, 30 products, and 100 scenarios [1]. For each scenario \( s \in S \), the stochastic demand \( d^s_k \) of each product \( k \in K \) is drawn according to a Uniform or a Gumbel probability distribution in \([0.5d_k, 2d_k]\) and its oscillation is calculated as \( \hat{d}^s_k = d^s_k - d_k \). Results are summarized in Table 1, where \( t(s) \) is the CPU time needed for each method, \( \text{gap}\% \) is the percentage MIP gap of the best solution found by Cplex in 4 hours, and \( \Delta\% \) is the percentage deviation of the PH solution from the Cplex’s one.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Uniform distribution</th>
<th>Gumbel distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cplex 12.7</td>
<td>PH</td>
</tr>
<tr>
<td></td>
<td>gap%</td>
<td>t(s)</td>
</tr>
<tr>
<td>5 10</td>
<td>0.00</td>
<td>8</td>
</tr>
<tr>
<td>5 20</td>
<td>0.00</td>
<td>31</td>
</tr>
<tr>
<td>5 30</td>
<td>0.00</td>
<td>53</td>
</tr>
<tr>
<td>10 10</td>
<td>0.00</td>
<td>582</td>
</tr>
<tr>
<td>10 20</td>
<td>0.00</td>
<td>856</td>
</tr>
<tr>
<td>10 30</td>
<td>0.00</td>
<td>1765</td>
</tr>
<tr>
<td>20 10</td>
<td>1.24</td>
<td>7238</td>
</tr>
<tr>
<td>20 20</td>
<td>3.60</td>
<td>12762</td>
</tr>
<tr>
<td>20 30</td>
<td>4.55</td>
<td>14400</td>
</tr>
</tbody>
</table>

Table 1: Preliminary results

Never exceeding one hour and a half of CPU time, the PH is able to always find optimal or near-optimal solutions, sometimes improving the best known upper bounds for some non-closed instances (see negative \( \Delta\% \) values). Hence, our PH outperforms Cplex both in terms of efficiency and quality of the solutions. The results confirm that our PH is a very promising method to address efficiently even bigger CTQD-AC\(_{ud}\) instances.

References


The electric vehicle routing problem with energy consumption uncertainty

Fan E
McGill University, Montreal, Canada

Samuel Pelletier
HEC Montréal, Montreal, Canada
Email: samuel.pelletier@hec.ca

1. Introduction

In an urban environment, freight electric vehicles (EVs) are often exclusively charged at a central depot, and must thus be able to perform their routes solely with their departing battery charge. Moreover, a recent study by [1] suggests that several parameters of an energy consumption model commonly used in EV routing problems are difficult to measure or can vary depending on a variety of exogenous factors, and that it is thus wiser to think of their value to be within an uncertainty range. We therefore introduce the EV routing problem with energy consumption uncertainty (e-VRP-ECU), in which EVs must be routed so that they never get stranded along their routes regardless of the realization of energy consumption uncertainties. The uncertainty surrounding EVs’ energy consumption has been approached using robust optimization in an optimal path problem setting by [4], but we extend it to the VRP setting (in which carried loads influence energy consumption). We also investigate the impact of specific parameters based on realistic uncertainty ranges.

2. Problem description and mathematical model

The e-VRP-ECU is defined on a complete graph \( G = (N, A) \). Let set \( N_0 \) refer to the customer nodes, and set \( N \) refer to the union of \( N_0 \) and the depot node 0. The set of arcs is denoted by \( A \). Each arc \((i, j)\) has a travel speed \( v_{ij} \) (m/s), a distance \( d_{ij} \) (m), a travel time \( t_{ij} \) (s) and a road angle \( \theta_{ij} \) (rad). Each node \( i \) has a demand \( q_i \) (kg) and a service time \( \pi_i \) (s), both worth zero for the depot. Routes can have a duration of at most \( T \) (s). Set \( K \) is the fleet of EVs, each having a load capacity \( L \) (kg), a battery capacity \( Q \) (kWh), a curb mass \( w \) (kg), a frontal area \( A \) (m\(^2\)), an air drag coefficient \( C_d \), an auxiliary power demand \( P \) (W), and a drive train efficiency \( \phi \). A fixed cost of \( \alpha \) is incurred for dispatching a vehicle. Parameters \( c_E \) and \( c_L \) represent energy (\$/kWh) and labor (\$/s) costs, respectively. Binary variables \( x_{ijk} \) take value 1 if vehicle \( k \) travels arc \((i, j)\), and take value 0 otherwise. Variables \( f_{ijk} \) refer to the load (kg) carried on arc \((i, j)\) by vehicle \( k \). Variables \( E_k \) refer to the battery level (kWh) of vehicle \( k \) when it leaves the depot. Assuming no acceleration along the arcs, the amount of energy \( e_{ijk}(f_{ijk}) \) (kWh) required for vehicle \( k \) to traverse \((i, j)\) with a load of \( f_{ijk} \) is computed with the model of [2]:

146
\[ e_{ij}(f_{ijk}) = \frac{1}{3.6 \cdot 10^6} \left( \phi ((w + f_{ijk}) gd_{ij} \sin \theta_{ij} + 0.5 AC_d \rho v_{ij}^2 d_{ij} + (w + f_{ijk}) gC_r d_{ij} \cos \theta_{ij}) + P_{ij} \right), \]

where \( g \) is the gravitational constant, \( \rho \) is the air density, and \( C_r \) is the rolling friction coefficient. We derive arc parameters \( a_{ij} \) and \( b_{ij} \) so that

\[
\begin{align*}
\phi (w + f_{ijk}) gd_{ij} \sin \theta_{ij} &+ 0.5 AC_d \rho v_{ij}^2 d_{ij} &+ (w + f_{ijk}) gC_r d_{ij} \cos \theta_{ij} &+ P_{ij} \\
&= a_{ij} \cdot x_{ijk} &+ b_{ij} \cdot f_{ijk}.
\end{align*}
\]

Finally, we assume that there is uncertainty on some of the terms in \( a_{ij} \) and \( b_{ij} \) (see [1] for more details on the sources of this uncertainty). We derive a robust optimization model by using the following polyhedral uncertainty set (with parameter \( \Gamma \) controlling the degree of uncertainty considered):

\[
\begin{align*}
Z = \{ \zeta \in \mathbb{R}^{|A|} | & 0 \leq \zeta_{ij} \leq 1, \sum_{(i,j) \in A} \zeta_{ij} \leq \Gamma \}. \\
\end{align*}
\]

Note that \( \zeta_{ij} \) are not variables. A given \( \zeta \in Z \) provides a \( \zeta_{ij} \) value for each arc in \( A \), with \( \zeta_{ij} \) affecting the energy consumption along arc \((i,j)\) and \( \zeta_{ij} = 1 \) corresponding to the worst-case energy consumption along that arc. Our goal is to ensure that the EVs have enough energy to perform their routes for all \( \zeta \in Z \). The e-VRP-ECU is thus:

\[
\begin{align*}
\min \sum_{k \in K} c_E \cdot E_k &+ \sum_{k \in K} \sum_{j \in N_0} \alpha \cdot x_{0jk} &+ \sum_{k \in K} \sum_{(i,j) \in A} c_L \cdot (\pi_i + t_{ij}) \cdot x_{ijk} \\
\end{align*}
\]

subject to

\[
\begin{align*}
x_{0jk} \leq 1 & \quad k \in K \quad (2) \\
\sum_{j \in N_0} x_{ijk} = 1 & \quad i \in N_0 \quad (3) \\
\sum_{j \in N \setminus \{i\}} x_{ijk} = & \quad \sum_{j \in N \setminus \{i\}} x_{ijk} \quad i \in N, k \in K \quad (4) \\
q_i \cdot x_{ijk} \leq f_{ijk} \leq (L - q_i) \cdot x_{ijk} & \quad (i,j) \in A, k \in K \quad (5) \\
\sum_{j \in N \setminus \{i\}} f_{ijk} - \sum_{k \in K} \sum_{j \in N \setminus \{i\}} f_{ijk} = q_i & \quad i \in N_0 \quad (6) \\
\sum_{(i,j) \in A} \left((a_{ij} + \hat{a}_{ij} \cdot \zeta_{ij}) \cdot x_{ijk} + (b_{ij} + \hat{b}_{ij} \cdot \zeta_{ij}) \cdot f_{ijk}\right) \leq E_k & \quad k \in K, \zeta \in Z \quad (7) \\
0 \leq E_k \leq Q & \quad k \in K \quad (8) \\
\sum_{(i,j) \in A} (\pi_i + t_{ij}) \cdot x_{ijk} \leq T & \quad k \in K \quad (9) \\
x_{ijk} \in \{0,1\} & \quad (i,j) \in A, k \in K \quad (10). \\
\end{align*}
\]

The objective is to minimize total energy, fixed, and labor costs. Constraints (2)–(4) ensure that at most \( m \) EVs are dispatched, that each customer is visited once, and that each EV exits nodes it enters. Constraints (5)–(6) bound the load on each arc and ensure
balance of flow. Constraints (7) mean that the total energy consumed by an EV never exceeds its initial battery level \( \hat{a}_{ij} \) and \( \hat{b}_{ij} \) are maximum deviations of \( a_{ij} \) and \( b_{ij} \). Since \( Z \) is a polyhedron, there is an infinite amount of constraints (7). Constraints (8) and (9) force the departing battery levels and route durations to be at most \( Q \) and \( T \), respectively.

3. Solution methods and computational experiments

Small instances of the e-VRP-ECU can be solved by 1) reformulating the model of Section 2 so as to obtain a tractable deterministic mixed integer linear program that can be solved with a commercial solver, or 2) using a cutting-plane method [3]. For larger instances, we propose a metaheuristic based on adaptive large neighborhood search [5].

3.1 Reformulation

For a given solution of the e-VRP-ECU (i.e., for fixed variables), we can check if constraints (7) are respected for vehicle \( k \) by verifying if the optimal solution of the following linear program (in which \( \zeta_{ij} \) are now variables) is worth at most \( E_k - \sum_{(i,j) \in A} (a_{ij} \cdot x_{ijk} + b_{ij} \cdot f_{ijk}) \):

\[
\max \sum_{(i,j) \in A} (\hat{a}_{ij} \cdot x_{ijk} + \hat{b}_{ij} \cdot f_{ijk}) \cdot \zeta_{ij}
\]

subject to

\[
\sum_{(i,j) \in A} \zeta_{ij} \leq \Gamma \quad (12)
\]

\[
0 \leq \zeta_{ij} \leq 1 \quad (i, j) \in A.
\]

The dual of the above linear program can be written with variables \( \lambda_k \) and \( \sigma_{ijk} \) as:

\[
\min \Gamma \cdot \lambda_k + \sum_{(i,j) \in A} \sigma_{ijk}
\]

subject to

\[
\lambda_k + \sigma_{ijk} \geq \hat{a}_{ij} \cdot x_{ijk} + \hat{b}_{ij} \cdot f_{ijk} \quad (i, j) \in A
\]

\[
\lambda_k \geq 0
\]

\[
\sigma_{ijk} \geq 0 \quad (i, j) \in A.
\]

We can thus find a tractable deterministic mixed integer linear programming formulation of the e-VRP-ECU by replacing constraints (7) in the model of Section 2 with:

\[
\sum_{(i,j) \in A} (a_{ij} \cdot x_{ijk} + b_{ij} \cdot f_{ijk} + \sigma_{ijk}) + \Gamma \cdot \lambda_k \leq E_k \quad k \in K,
\]

and by adding constraints (15)–(17) for each vehicle \( k \in K \), since duality theory ensures that \( \Gamma \cdot \lambda_k + \sum_{(i,j) \in A} \sigma_{ijk} \geq \max_{\zeta \in Z} \left\{ \sum_{(i,j) \in A} (\hat{a}_{ij} \cdot x_{ijk} + \hat{b}_{ij} \cdot f_{ijk}) \cdot \zeta_{ij} \right\} \) for vehicle \( k \) as long as constraints (15)–(17) are verified. Note that \( \lambda_k \) and \( \sigma_{ijk} \) are variables in the reformulation.
3.2 Cutting-plane method

Although there is an infinite amount of constraints (7) in the model of Section 2, only a few of them will be binding for the optimal solution. We therefore initiate a branch-and-cut procedure on the nominal version of the problem (i.e., with no uncertainty). Whenever an integer solution is found, we verify whether the solution is robust with regard to the entire uncertainty set $Z$. If it is not, then we identify and add new lazy constraints.

3.3 Adaptive large neighborhood search

To construct an initial solution, we use the Clarke and Wright savings algorithm. We then use several destroy and repair operators that compete against each other to modify the current solution. Simulated annealing is used as the local search framework. Operators’ scores are updated every certain number of iterations according to several criteria. Robust routes are maintained in a pool during the procedure, and a set partitioning problem over the set of robust routes is solved after a number of iterations to determine the final solution.

3.4 Computational experiments

Preliminary experiments show that both exact methods can solve instances of the problem to optimality with at most 10 customers within 10,800 seconds. Additional tests will be conducted to evaluate the metaheuristic, to study the impact of the uncertainty surrounding individual parameters in the energy consumption model, and to compare robust solutions to solutions obtained with the nominal version of the problem.

References


Robust Optimization of Heterogeneous Vehicle Routing Problems under Demand Uncertainty

Anirudh Subramanyam  Chrysanthos E. Gounaris
Carnegie Mellon University, Pittsburgh, USA

Panagiotis P. Repoussis
Athens University of Economics and Business, Athens, Greece, and
Stevens Institute of Technology, Hoboken, USA
Email: panagiotis.repoussis@stevens.edu

1 Introduction and Motivation

The Heterogeneous Vehicle Routing Problem (HVRP) is a well-known generalization of the Capacitated Vehicle Routing Problem (CVRP) in which the fleet of vehicles is not homogeneous. In particular, the fleet may consist of vehicles with different capacities, fixed/one-time costs (e.g., reflecting rental or capital amortization costs) and variable/recurring costs (e.g., reflecting fuel costs). Moreover, in addition to the standard routing decisions in the CVRP, one must also make fleet composition decisions in the HVRP. The HVRP is a very powerful model since it subsumes a number of other VRP variants, including but not limited to, the Fleet Size and Mix VRP, the Fixed Fleet HVRP, the Site Dependent VRP and the Multi-Depot CVRP. We refer the reader to the survey [1] for an up-to-date review of existing studies of the HVRP, including applications, models and solution methods.

In almost all of the existing studies of the HVRP, the customer demands are assumed to be precisely known at the time of decision-making. However, in several practical applications, this information is not available, and the actual demand is observed only during the execution of the transportation plan. A popular approach is to solve the HVRP using some nominal values of demand and implement the resulting solution. The consequence of adopting this approach is that the resulting plan may become infeasible during actual execution, leading to potentially severe contractual and reputational penalties (e.g., lost sales, loss of customer goodwill), or too expensive in terms of routing costs (e.g., commissioning of additional vehicles). These penalties are particularly significant in the case of HVRPs since they are associated with longer-term fleet dimensioning decisions. Therefore, anticipating and explicitly incorporating uncertainty during decision-making is crucial.

To our best knowledge, [2] is the only paper to consider uncertainty in the HVRP.
particular, the authors model customer demands as uniform random variables and propose a continuous approximation heuristic to construct potential routes for each vehicle type such that the probability of failure along the designed routes is small. A survey of modeling and solution approaches for stochastic variants of general VRPs can be found in [3].

In contrast to the above, our approach is based on robust optimization. Instead of assuming specific probability distributions for the customer demands, we assume only that their true values may realize from a pre-specified uncertainty set; and aim to determine a cost-optimal fleet composition and transportation plan that remains feasible for all possible demand realizations from within the postulated set. For the CVRP under demand uncertainty, such approaches based on robust optimization have been studied in [4, 5, 6].

Our contributions are two-fold. First, we develop robust versions of various classical local search moves. Specifically, we show how local search can be augmented so as to efficiently generate candidate routes which are robust, i.e., remain feasible for any postulated demand realization. These robustified local search moves are then used within Iterated Local Search (ILS) and Adaptive Memory Programming (AMP) metaheuristics to generate high-quality robust feasible solutions. Second, we develop a new integer programming (IP) formulation and a branch-and-cut algorithm to obtain lower bounds on the optimal robust HVRP solution. This allows us to quantify the quality of the metaheuristic solutions.

2 Methodology

The HVRP is defined on a complete directed graph $G = (V, A)$ with nodes $V = \{0, 1, \ldots, n\}$ and arcs $A$. Node 0 represents the depot while $V_C = V \setminus \{0\}$ represents the set of customers. Each customer $i \in V_C$ has an uncertain demand $q_i$. The customer demands are assumed to be random variables with a known, non-empty and compact support $Q \subseteq \mathbb{R}^n_+$, also referred to as the uncertainty set. Let $K = \{1, \ldots, m\}$ denote the index set of vehicle types. Each vehicle of type $k \in K$ has capacity $Q_k$, and it incurs a fixed cost $f_k$ if it is used and a travel cost $c_{ijk}$ if it traverses the arc $(i, j) \in A$.

2.1 Heuristic Approaches: Local Search, AMP and ILS

All metaheuristic algorithms for the HVRP consist of a number of local node- and edge-exchange moves that are used to obtain locally optimal solutions within some pre-specified neighborhood structures. For example, in AMP, local search is extensively employed within its initialization and exploitation phases. Similarly, ILS consists of repeated applications of local search to a perturbation of the current best solution. Typically, we only accept moves which guarantee that the solution (that would result after adopting the move) is feasible. For the robust HVRP, this means that we must be able to verify that the total demand on a candidate route never exceeds its capacity for any possible demand realization. That
is, if \( R_k = (0, R_{k,1}, \ldots, R_{k,n_k}, 0) \) denotes a candidate route traversed by a vehicle of type \( k \in K \), where \( R_{k,l} \in V_C \) denotes the \( l^{th} \) customer on the route, then, before we accept the move that results in this route, we must verify that:

\[
\sum_{l=1}^{n_k} q_{R_{k,l}} \leq Q_k \quad \forall q \in Q \iff \max_q \sum_{l=1}^{n_k} q_{R_{k,l}} \leq Q_k.
\]  

(1)

Since local search typically postulates a large number of such candidate routes, it is crucial that the above inequality can be verified efficiently. This amounts to solving a convex optimization problem for general \( Q \) and makes the overall search computationally excruciating. Fortunately, we can show that the worst-case demand can be computed analytically (i.e., in closed form) for the following popular classes of uncertainty sets: (i) “disjoint budgeted” polyhedral sets \([5]\), (ii) “factor models” \([5]\), (iii) ellipsoidal sets \((2)\), and (iv) cardinality-constrained sets \((3)\). This enables fast verification of robust feasibility within local search and a computationally tractable metaheuristic algorithm.

\[
Q(\mu, \Sigma) = \{ q \in \mathbb{R}^n : \exists \xi \in \Xi \text{ such that } q = \mu + \Sigma^{1/2} \xi \} \quad \text{with } \Xi = \{ \xi \in \mathbb{R}^n : \|\xi\|_2 \leq 1 \} \quad \text{\(2\)}
\]

\[
Q(\mu, \bar{q}, \Gamma) = \{ q \in [\mu, \mu + \bar{q}] : \|q - \mu\|_0 \leq \Gamma \} \quad \text{\(3\)}
\]

2.2 Branch-and-Cut Approach

Our IP formulation uses binary variables \( y_{ik} \) to record if customer \( i \in V_C \) is visited by a vehicle of type \( k \in K \), and \( x_{ijk} \) to record if arc \((i, j) \in A\) is traversed by a vehicle of type \( k \in K \). In the following, \( V_k := \{ i \in V_C : \max\{q_i : q \in Q\} \leq Q_k \} \) denotes the index set of customers that can be served by a vehicle of type \( k \in K \) under any demand realization.

\[
\begin{align*}
\text{minimize} & \quad \sum_{k \in K} \sum_{i \in V_k} \sum_{j \in V_C} c_{ijk}x_{ijk} + \sum_{k \in K} \sum_{i \in V_C} f_kx_{0ik} \\
\text{subject to} & \quad \sum_{k \in K} y_{ik} = \sum_{k \in K} y_{ik} = 1 \quad \forall i \in V_C \quad \text{\(5\)} \\
& \quad \sum_{j \in V_C} x_{ijk} = \sum_{j \in V_C} x_{jik} = y_{ik} \quad \forall i \in V_C, \forall k \in K \quad \text{\(6\)} \\
& \quad \sum_{i \in V \setminus S} \sum_{j \in S} x_{ijk} + 2 \sum_{i \in S} (1 - y_{ik}) \geq 2 \left[ \frac{1}{Q_k} \max_{q \in Q} \sum_{i \in S} q_i \right] \quad \forall S \subseteq V_k, \forall k \in K \quad \text{\(7\)} 
\end{align*}
\]

Constraints (7) are a generalization of the rounded capacity inequalities. They break subtours and enforce conditional lower bounds on the number of vehicles of type \( k \in K \) that must serve a customer set \( S \subseteq V_k \). For any \( Q \), it can be shown that these constraints are both necessary and sufficient to induce a robust feasible solution for the HVRP. Observe that since the maximization on the right-hand side does not depend on any decision variables, we can replace this quantity with the optimal objective value of the corresponding optimization problem.
The above model is solved using a branch-and-cut algorithm by adding violated inequalities (7) at every fractional solution. The separation is done by employing a heuristic that iteratively constructs candidate subsets $S$ through a number of greedy perturbations. Since the heuristic typically constructs a large number of such subsets, it is crucial that the corresponding right-hand side of (7) can be evaluated efficiently. Fortunately, this latter quantity is exactly the same as the one which appears in the left-hand side of inequality (1) associated with $R_k = S$. Therefore, for the four broad classes of uncertainty sets that we discussed previously, the right-hand side of (7) can be computed analytically, avoiding the solution of a convex optimization problem at every iteration. This enables fast separation and a numerically tractable branch-and-cut algorithm.

3 Computational Results

We applied our branch-and-cut, ILS and AMP algorithms to the standard set of Fleet Size and Mix benchmark instances ($n = 20 - 100$, $m = 3 - 6$) for disjoint budget and factor model uncertainty sets. Our preliminary results indicate that the “robustified” metaheuristics perform as efficiently as their deterministic counterparts in terms of running time (~seconds). Moreover, the branch-and-cut algorithm is able to certify that the obtained solutions are optimal for $n < 50$, within 5% of optimality when fixed costs are present and within 8% of optimality otherwise. Finally, the “price of robustness” is less than 5% indicating that robustness can be achieved with a small increase in routing costs.

References


Robust Multi-Period Vehicle Routing under Customer Order Uncertainty

Anirudh Subramanyam
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA

Frank Mufalli  Jose L. Aguirre  Jose M. Pinto
Praxair Inc., USA

Chrysanthos E. Gounaris
Department of Chemical Engineering,
Carnegie Mellon University, Pittsburgh, USA
Email: gounaris@cmu.edu

1 Introduction and Motivation

Traditional variants of the Vehicle Routing Problem (VRP) are of an operational nature and the typical setting involves routing within a single work day. However, many transportation problems involve the tactical planning and routing of vehicles over a longer horizon spanning multiple days, e.g., a week. This is the case when customer orders are received dynamically during the week, and each request specifies a set of days during which service may take place. On any given day, the distributor must decide which orders to serve along with the actual vehicle routes. This is typically done by assigning a visit day to each pending request such that the total transportation cost over the planning horizon is minimized. In practice, the plan is implemented in a rolling horizon fashion: routes of the first day are executed while new orders are received. The set of unfulfilled and newly received orders are considered for scheduling the following day and the process is repeated on an updated planning horizon. Examples of such decision-making setups can be found in various systems, including food distribution [1] and auto-carrier transportation [2].

The planning problem to be solved on any given day has been referred to as the Dynamic Multi-Period VRP [1] and Tactical Planning VRP [3]. In these papers, decisions are determined through the solution of deterministic optimization problems by considering only those service requests that have already been placed and ignoring the potential for new requests to be received during the planning horizon. Unfortunately, such decisions can create situations which can either be infeasible, leading to additional costs to recover
feasible solutions (e.g., commissioning an additional vehicle due to limited fleet capacity), or too expensive in terms of transportation costs (e.g., need for driver overtime due to limited working hours). In contrast, it is possible to generate better, risk-averse decisions by explicitly accounting for customer order uncertainty on any given day.

The paper [4] considers probabilistic descriptions of customer order uncertainty in the context of multi-period vehicle routing. The authors assume that on any given day, the probability of a potential customer requesting service at any point in the future is known precisely, and use this information to formulate a Prize Collecting VRP to determine which orders to serve now and which ones to postpone. However, while distributors often have large amounts of historical data which can be used to obtain aggregate order forecasts, they typically do not have access to detailed probability distributions over future requests.

1.1 Our Contributions

In this paper, we study the modeling and solution of the tactical planning multi-period VRP through the lens of robust optimization. In particular, we propose a set-based model of uncertainty, each member of which represents a potential combination of future customer orders. We utilize this uncertainty set to formulate a two-stage robust optimization model. The model consists of first-stage visit decisions that assign known and potential orders to days and second-stage routing decisions which guarantee that the designed tactical plan remains “bin packing feasible” for any possible realization of future orders from the postulated uncertainty set. We also develop a numerically tractable branch-and-cut solution algorithm for this model. Computational experiments demonstrate that our approach is practically tractable and generates high quality robust plans at marginal cost increases above nominal plans. Details of the methodology can be found in our publication [5].

2 Methodology

Let $G = (N \cup \{0\}, E)$ denote an undirected graph with customer nodes $N$ and depot node 0. The depot node consists of $m$ vehicles of capacity $Q$ each, while customer node $n \in N$ is associated with a demand quantity $q_n > 0$. There is a travel cost $c_{ij}$ for each $(i, j) \in E$.

Let $P = \{1, \ldots, h\}$ denote the current planning horizon and let $V_0$ denote the set of pending customer orders. Each order $i \in V_0$ is associated with a customer $n_i \in N$ and a service day window $P_i = \{p_i, \ldots, p_i + \ell_i\} \subseteq P$ during which service must be rendered.

2.1 Model of Uncertainty

We assume there is a set of potential orders, $V_1$, each of which may or may not realize during the planning horizon. Each order $i \in V_1$ is associated with a customer $n_i \in N$, “order day” $p_i \in P$ and a service day window $P_i = \{p_i + e_i, \ldots, p_i + \ell_i\}$ ($1 \leq e_i \leq \ell_i$)
during which service must be rendered if the order realizes. As before, we may assume $P_i \subseteq P$. Note how this model allows us to capture the uncertainty in the order day $p_i$ by postulating multiple orders from the same customer on different days of the horizon.

Let $\xi_i \in \{0, 1\}$ denote a random variable indicating if order $i \in V_1$ realizes during the planning horizon. The set of potential order combinations that can occur during the horizon is captured by an uncertainty set of the form: $\Xi = \{\xi \in \{0, 1\}^{|V_1|} : A\xi \leq b\}$. Note that the uncertainty set consists of a finite (but possibly very large) number of points, each of which represents a potential future scenario. This model of uncertainty is very flexible, as it allows us to capture various practically meaningful scenarios that adhere to underlying correlations linking customer orders, and it can be readily regressed from historical data without requiring probability distributions. Moreover, its size can be tuned in order to reflect the distributor’s level of risk aversion. For example, we can model

- Budget of orders received during the horizon: $\sum_{i \in V_1} \xi_i \leq \Gamma$
- Budget of orders received on any given day $p \in P$: $\sum_{i \in V_1 : p_i = p} \xi_i \leq \Gamma_p$
- Budget of orders received from the same customer $n \in N$: $\sum_{i \in V_1 : n_i = n} \xi_i \leq 1$

### 2.2 Two-Stage Robust Optimization Model and Solution Algorithm

We formulate a two-stage robust optimization model in which the first-stage decisions correspond to assignments of (pending and potential) customer orders to days while the second-stage decisions guarantee that the first-stage plan remains feasible for any combination of future orders from the postulated uncertainty set $\Xi$. The model can be conceptually described as follows:

\[
\begin{align*}
\text{minimize} & \quad \left( S_1, \ldots, S_h \right) \in \mathcal{F} \sum_{p \in P} \text{CVRP}(S_p \cap V_0) \\
\text{subject to} & \quad \text{BPP} \left( S_p \cap (V_0 \cup \{i \in V_1 : \xi_i = 1\}) \right) \leq m \quad \forall \xi \in \Xi
\end{align*}
\]  

(1)

In this model, $\mathcal{F}$ denotes the set of all feasible assignments of customer orders to periods: each $(S_1, \ldots, S_h) \in \mathcal{F}$ represents a partition of the set $V := V_0 \cup V_1$ where $S_p$ is the (possibly empty) subset of orders to be visited in period $p \in P$. Note that each order $i \in S_p$ necessarily satisfies $P_i \cap S_p \neq \emptyset$ translating to service within its day window. In the above, for a given $W \subseteq V$, $\text{CVRP}(W)$ denotes the minimum cost of the Capacitated VRP to serve the orders in $W$, while $\text{BPP}(W)$ denotes the minimum number of bins (vehicles) needed to pack the items of $W$ using bin size $Q$.

The objective function minimizes the cost of routing the pending orders $V_0$ over the planning horizon. The constraints ensure that for every possible scenario $\xi \in \Xi$, the set of customers which will be visited on day $p$, $S_p \cap (V_0 \cup \{i \in V_1 : \xi_i = 1\})$, can be “bin packed” into $m$ vehicles, i.e., can be served without violating the available fleet capacity.
We formulated the conceptual model (1) as an integer program using binary variables $y_{ip}$ to record if order $i \in V$ is visited on day $p \in P$, i.e., $i \in S_p \Leftrightarrow y_{ip} = 1$. The standard two-index variables $x_{ijp}$ are used to record if edge $(i, j)$ is traversed in period $p \in P$. The capacity constraints are formulated via the following robust cover inequalities:

$$m + \sum_{i\in W} (1 - y_{ip}) \geq \text{BPP} \left(W \cap \left(V_0 \cup \{i \in V_1 : \xi_i = 1\}\right)\right) \quad \forall W \subseteq V, \forall p \in P, \forall \xi \in \Xi. \quad (2)$$

It can be shown that the robust cover inequalities (2) are both necessary and sufficient to induce a set of customer-to-period assignments that is feasible for all $\xi \in \Xi$.

The integer program is solved using a branch-and-cut algorithm in which the robust cover inequalities (2) are dynamically enforced by using a separation algorithm. The exact separation of these inequalities involves computing the maximizer $\xi^* \in \Xi$ of its right-hand side which amounts to the solution of a bilevel program for general $\Xi$. We can show that for uncertainty sets described as the intersection of “disjoint” budget constraints (see §2.1), the maximizer $\xi^*$ can be computed in closed-form, avoiding solution of a bilevel program.

3 Computational Experiments and Conclusion

We applied our algorithm to a large set of 5-period benchmark instances [3] for various uncertainty sets (ranging in size from $|V_0| \approx 20 - 200$, $|V_1| \approx 3 |V_0|$, $m \approx 2 - 6$). Our results indicate that (i) the robust cover inequalities (2) can be efficiently separated in the context of our branch-and-cut algorithm, (ii) robust solutions can be obtained with a computational effort similar to that for deterministic solutions, and (iii) the “price of robustness” is small; for the instances we considered, it was less than 2% on average.

References


Solving the robust CVRP under demand uncertainty

Artur Pessoa
Department of Industrial Engineering,
Universidade Federal Fluminense, Niteroi, Brazil
Email: artur@producao.uff.br

Michael Poss
UMR CNRS 5506 LIRMM
Université de Montpellier, France

Ruslan Sadykov, François Vanderbeck
Institut de Mathématiques de Bordeaux
Université de Bordeaux and Inria Bordeaux, Talence, France

1 Introduction

We consider the following classical Capacitated Vehicle Routing Problem (CVRP). Let $G = (V, A)$ be a complete digraph with nodes $V = \{0, 1, \ldots, n\}$ and arcs $\{(i, j) \in V \times V : i \neq j\}$. Node 0 ∈ V represents the unique depot, and each node $i \in V' = V \setminus \{0\}$ corresponds to a customer with demand $d_i \in \mathbb{R}_+$ (let $d_0 = 0$). The depot is the departure and return base for a fleet $m$ homogeneous vehicles of capacity $Q$. The set of vehicles is denoted as $K = \{1, \ldots, m\}$. Each vehicle incurs a transportation cost $c_{ij} \in \mathbb{R}_+$ if it traverses the arc $(i, j) \in A$. We define a route as a depot-base-walk in graph $G$, i.e., a sequence of nodes visited by a vehicle such that the first and the last node in this sequence is the depot. Let $a_i^r$ be equal to the number of times a node $i \in V'$ is visited in a route $r$. A route $r$ is demand-feasible if the total demand of visited customers does not exceed the vehicle capacity: $a_i^r d_i \leq Q$. Let $\Omega$ be the set of all demand-feasible routes. We assume here that demand-feasible routes may visit a node more than once. The cost $c^r$ of a route is the total transportation cost incurred by traversing the route arcs. The problem consists in finding a set of demand-feasible routes of the minimum total cost such that every customer is visited exactly once. By introducing a binary variable $\lambda_r$ for every $r \in \Omega$, the problem
can be modeled by the following set partitioning formulation.

\[
\begin{align*}
\text{min} & \quad \sum_{r \in \Omega} c^r \lambda_r, \\
\text{s.t.} & \quad \sum_{r \in \Omega} a^r_i \lambda_r = 1, \quad i \in V', \\
& \quad \sum_{r \in \Omega} \lambda_r \leq m, \\
& \quad \lambda_r \in \{0, 1\}, \quad r \in \Omega.
\end{align*}
\]

2 Demand uncertainty

We consider that the demand vector \(d\) is uncertain and can take any value in a given polytope \(D\) that is included in the box \([\underline{d}, \underline{d} + \bar{d}]\) defined by the vectors \(\underline{d}, \bar{d} \in \mathbb{R}_+^n\), where the components of \(\underline{d}\) represent the nominal values, while those of \(\bar{d}\) are the deviations.

Observe that considering downward deviations of \(d\) is not of any use: they would not impair the feasibility given that demand values only play a role in the capacity constraints.

Our study focuses on two definitions of polytope \(D\). The first one is the budget polytope introduced in [1], and widely used in the robust combinatorial optimization literature since then. Given \(\Gamma \in \mathbb{R}_+\), the budget polytope is given by

\[
D^{\text{knap}} = \left\{ d \in \mathbb{R}_+^n : d_i = \underline{d}_i + \eta_i \hat{d}, i \in V' \cap V^0, \sum_{i \in V^0} \eta_i \leq \Gamma, \ 0 \leq \eta \leq 1 \right\}.
\]

The second polytope we consider is that of [2, 3]. Therein is defined a customer partition \(V_C = V_1 \cup \cdots \cup V_s\) and associated budgets \(b_1, \ldots, b_s \in \mathbb{R}_+\), yielding polytope:

\[
D^{\text{prop}} = \left\{ d \in \mathbb{R}_+^n : d_i = \underline{d}_i + \xi_i, i \in V^0, \sum_{i \in V_k} \xi_i \leq b_k, k = 1, \ldots, s, \ 0 \leq \xi \leq \bar{d} \right\}.
\]

Considering the change of variable \(\xi_i = \hat{d}_i \eta_i\) underlines that \(D^{\text{knap}}\) constrains the number of elements of \(d\) that deviate simultaneously from their nominal values, while \(D^{\text{prop}}\) constrains the total amount of deviation within each subset of customers.

The demand uncertainty only affects the feasibility of the routes \(r\) used in the set partitioning formulation. Therefore, the robust counterparts of the above formulation result in set partitioning formulations that involve the set of routes that are demand-feasible for all values of \(d\) in \(D^{\text{knap}}\) or \(D^{\text{prop}}\). We denote these sets of routes by \(\Omega(D^{\text{knap}})\) and \(\Omega(D^{\text{prop}})\), respectively.
3 Robust algorithm

The contribution of our work is to extend the classical algorithms developed for the deterministic CVRP to their robust counterpart. Specifically, Branch-Cut-and-Price algorithms are the state-of-the-art approaches to solve this set partitioning formulation [5]. Therein, the linear relaxation of the formulation is solved by column generation. The pricing problem is solved typically by dynamic programming using a forward labeling algorithm. To improve the quality of column generation lower bounds, set $\Omega$ is restricted to the set of ng-paths. Rounded capacity cuts and limited memory rank-1 cuts are used to further strengthen the root bound. In this work, we use the Branch-Cut-and-Price algorithm from [7]. It has the advantage to efficiently handle instances with real value demands, which is important in the aforementioned robust variant of the problem.

In adapting the deterministic algorithms one must essentially specialize the pricing oracle to generate routes in $\Omega(D^{knap})$ or $\Omega(D^{prop})$. This amounts to solving a minimum cost ”robust” constrained shortest path problems. Using well-known results from the robust optimization literature (see [1, 4, 6] for details) these problems are equivalent to solving a set of deterministic constrained shortest path problems with different weights, and taking the best of them. Specifically, the sets $D^{knap}$ and $D^{prop}$ require to solve $H(D^{knap}) = \lceil \frac{n+1}{2} \rceil + 1$ and $H(D^{prop}) = 2^s$ deterministic problems, respectively. Let $\Omega_h(D^{knap})$ and $\Omega_h(D^{prop})$ denote the sets of routes that can be generated by the $h$-th deterministic constrained shortest path problem associated to the sets $D^{knap}$ and $D^{prop}$, respectively. In the proposed set partitioning formulation, we just set $\Omega = \bigcup_{h=1}^{H(D)} \Omega_h(D)$, for both $D = D^{knap}$ and $D = D^{prop}$, transforming the robust problem into a deterministic equivalent CVRP.

Other important modifications to the algorithm of [7] are the extension of the valid inequalities to the robust context and the implementation of a dedicated heuristic to find primal solutions. We report on our iterated local search (ILS) heuristic with variable neighborhood search (VNS).

4 Results

We performed preliminary computational tests on the instances with up to 150 customers proposed by [2, 3] for uncertainty set $D^{prop}$. Our results indicate that (i) our ILS-VNS heuristic algorithm obtains significantly better solutions than the AMP heuristic proposed by [2] using a small fraction of its CPU time, and (ii) our exact algorithm is orders of magnitudes faster than [3], solving all instances to optimality.
References


Two-echelon distribution with city hub capacity management

Pamela C. Nolz
AIT Austrian Institute of Technology, Vienna, Austria
Email: pamela.nolz@ait.ac.at

Diego Cattaruzza
Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL, Lille, France

Nabil Absi, Dominique Feillet
Ecole Nationale Supérieure des Mines de Saint-Etienne, Gardanne, France

1 Problem Description

In this work, we present an innovative distribution scheme aiming at improving the environmental footprint of companies operating in a city logistics context. More specifically, we investigate the daily operations of a parcel delivery company settled in the urban area of Vienna. We propose a two-echelon distribution scheme where goods are not transported directly from their origin (a depot or a warehouse) to their final destination (end customers), but are initially moved into an intermediate platform (a city hub), from where their final distribution is performed. The city hub is located near the city center, close to the final destination of the goods, allowing for a more efficient and environmentally friendly last mile transport.

On the first echelon, trucks transport goods, i.e. parcels, from the depot outside the city to the city hub, where goods are transferred to tricycles, operating on the second echelon between the city hub and the final customers. Tricycles are able to access narrow streets, are not exposed to the problem of limited parking space and decrease the amount of emissions produced through delivery operations inside city centers. However, since not all parcels fit the cargo space of a tricycle, trucks are allowed to visit final customers if these customers require a quantity exceeding the capacity of a tricycle.

All parcels are present at the depot in the morning, when the trucks can pick-up a certain amount of goods and deliver them to the final customers and the city hub. Being located in the inner-city area, the storage capacity of the city hub is limited, which implies that not all goods can be transported from the depot to the city hub at once. Instead, the trucks are obliged to transfer parcels on the first echelon performing multiple trips, while
respecting the capacity of the city hub and the final customer time windows. On the second echelon, tricycles can only start their delivery tours as soon as the goods have arrived to the city hub, thus freeing city hub capacity as parcels are picked-up on multiple trips. We specify time windows (TWs) to indicate the earliest and latest time a final customer demands to receive an order (this trend has become very popular in restaurants or small stores, where the availability of goods at a specific time is crucial for their operation). Nonetheless, TWs can also be used to regulate the access to certain areas at specific times due to public regulations.

Organizing the described distribution scheme is a challenging task, since the coordination of first- and second-echelon trips is necessary: all second-echelon trips depend on the delivery of goods on the first echelon, which is limited by the city hub capacity. We call the problem of optimizing vehicle routes in this distribution scheme the Capacitated Two-Echelon Vehicle Routing Problem with Time Windows (C2E-VRPTW).

The goal of the C2E-VRPTW is to properly determine a set of vehicle routes to satisfy customer demands at a minimum travel cost, while respecting capacity and time constraints.

2 Solution Method

To cope with the challenges encountered in this real-world problem, we propose an iterative three phase solution method aiming at efficiently eliminating infeasible solutions.

2.1 First phase: solving the second-echelon problem

The second-echelon problem corresponds to the final leg of distribution performed by (electric) tricycles. In our characterization, the tricycles cannot start their trips before the goods of the customers to be served by those trips arrive to the city hub. However, for this phase (at least for the first iteration) we assume that all the goods are permanently available in the city hub; meaning that the tricycles can start their trips without waiting for the goods to arrive. Hence, the problem is seen as a Multi Trip Vehicle Routing Problem with Time Windows (MTVRPTW). The first echelon of the problem is not yet considered.

To solve this subproblem, the population-based algorithm developed by Cattaruzza et al. [1] for the MTVRPTW with release dates is used. The method proposed in Cattaruzza et al. [1] is a population-based algorithm using the giant tour decomposition scheme introduced by Prins [2], coupled with local search operators.

Each solution indicates a set of trips plus the order and assignment of every trip to the tricycles, in such a way that the demand of every customer is satisfied during the imposed TW. The starting time, load, and sequence of nodes to be visited, is given by the
solution for each one of the trips. Moreover, considering that this might not be the final arrangement of the second-echelon trips, load and time window violations are permitted but signalized in the solution.

2.2 Second phase: defining the visits to the city hub

After having an initial solution for the second-echelon problem, the next step is to generate an input for the first echelon, based on the information obtained from the previous phase. To accomplish this, a set of visits to the city hub is introduced. These visits are meant to indicate the services that the first-echelon vehicles will need to do to the city hub with the goods for the second-echelon customers, throughout one working day.

The input for this process is the starting time, load, and sequence of the second-echelon trips, together with the city hub capacity. A visit is defined for every second-echelon trip and the demand of the visit corresponds to the load of the related trip. A greedy algorithm is used to determine the visiting time windows for the city hub, respecting its limited capacity.

2.3 Third phase: solving the first-echelon problem

Taking advantage of the fact that the visits defined in the previous phase share the same characteristics as the first-echelon customers, a set of vertices that indifferently refers to the first-echelon customers given by the problem, and the visits defined during the second phase, is fixed.

Through this characterization, the first-echelon problem can further be reduced to an instance of the MTVRPTW, by fixing release dates to 0 for all the vertices. Thus, it is solved resembling the first phase of the method, using the algorithm proposed by Cattaruzza et al.[1].

In this case, the input for the algorithm will consist in the set of vertices (visits plus first-echelon customers) to be served by trucks in the first echelon with their related demands, time windows, and release dates.

2.4 Iterative Process

From the input defined in the third phase, the algorithm will provide a first-echelon solution. Yet, it might be impossible for the trucks to perform the visits within the established TWs, meaning that the goods of the trip related to that infeasible visit will arrive after the stipulated starting time. With this in mind, a new starting time \( sT^* \) given by the actual arrival of the goods to the city hub is introduced. This \( sT^* \) implies a delay in the service of the second-echelon customers related to that trip. Thus, we also introduce an actual service time \( aT \) to refer to this new service time.
After delaying the second-echelon trips, the $aT$ obtained for every customer, may (or may not) violate the TW of the customer. By saying this we imply that even if a trip is delayed, it might still serve some customers in their respective TW; or it can indeed violate their TW. Thus, we make a distinction between these 2 types of customers: customers with an $aT$ that is still within their associated TW, and customers with a TW violation given by the delay of the trip. From this distinction we proceed to update release dates of the second-echelon customers in the first phase, giving way to the iterative process.

3 Results and Conclusion

For testing the performance of our algorithm we use two types of test instances. As a first set of data we use adapted Solomon instances for the VRPTW, while the second set of data is based on real-world information from a parcel delivery company operating in the city of Vienna.

We will present experiments on sensitivity analysis with respect to relevant parameters. First, we vary the capacity of the city hub ranging between half and twice the capacity of the first-echelon vehicles. Second, we investigate the effects of variations in the distance between the depot and the city hub. Results include the CPU time in seconds, the number of used vehicles, the number of first- and second-echelon routes and distance traveled, and the number of iterations of the method.

These parameters act as key performance indicators for decision makers aiming at implementing an efficient two-echelon distribution system with limited storage capacity.

Acknowledgements

This work received funding from the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) in the framework of the research programme ”Stadt der Zukunft” and the Austrian Federal Ministry of Science, Research and Economy (BMWFW) under grant agreement no. 854921 (CIVIC), as well as from the European Union’s Horizon 2020 research and innovation program, and from the Austrian Research Promotion Agency (FFG) under project no. 845100 (EMILIA).

References


Incentive Schemes for Same-Day Delivery Routing

Marlin W. Ulmer
Technische Universität Braunschweig,
Braunschweig, Germany
m.ulmer@tu-braunschweig.de

1 Motivation

Same-day delivery (SDD) is a powerful tool for online retailers to increase sales. SDD is convenient because customers can order online and do not need to go to the store and wait in lines. Further, customers receive their good within a few hours. Thus, SDD narrows the gap of instant gratification compared to brick and mortar shopping (Anderson 2015). As a result, SDD experiences high two-digit growth rates per year (Yahoo! Finance 2016). Further, the majority of customers is willing to pay delivery fees for SDD (eMarketer 2015). Many retailers offer a set of SDD options differing in delivery speed and price (Grösch 2016). Often, SDD is promised within four-hour delivery deadlines but in some cities like Berlin, Amazon already offers two-hour delivery and even partially one-hour express delivery dependent on products and customer locations (Benedikt et al. 2016).

The combination of SDD and narrow deadlines leads to significant economic challenges for service providers (Ram 2015). Conventional last-mile delivery already causes a majority of the overall delivery costs (Bernau et al. 2016). As Punakivi and Saranen (2001) and Ulmer (2017) show, delivery time commitments additionally increase delivery costs and/or reduce the potential of serving many additional customers and gaining additional revenue. Hence, service providers price different delivery options differently. In their pricing decisions, service providers have two goals in mind. On the one hand, they aim on maximizing the overall obtained delivery fees per day to compensate for the delivery costs. On the other hand, because SDD leads to near-instant gratification it may increase the number of orders in the future (Anderson 2015). As a result, the service providers aim on selecting delivery prices leading to both high revenue in delivery fees and a large number of same-day deliveries. Suitable prices may therefore be dynamically adapted with respect to resources available and customer demand as common in many business models like airline ticketing or gasoline retail (Borenstein 1996, You 1999). Recently, delivery services draw on incentive pricing schemes to control the customer behavior by giving incentives for “efficient” delivery options. This is generally conducted in the field of attended home
delivery (Asdemir et al. 2009, Agatz et al. 2011, Yang et al. 2014, Yang and Strauss 2016). These problems differ significantly from SDD because, in SDD, customers order while the vehicles are on the road.

In this research, we transfer the concept of incentive pricing schemes to same-day delivery. We consider a SDD business model common in e-commerce (e.g., Amazon Prime Now): During the day, customers log on to the retailer’s website and select a set of goods. After finishing the selection, the customer proceeds to the checkout page. At the checkout page, the customer selects the delivery address. Based on the address, the retailer instantly offers a set of different delivery options. These options may comprise SDD with different delivery deadlines (e.g., one-hour, two-hours, four-hours) and a conventional (next-day) delivery. Each option is associated with a price. These prices are individual for each customer and may be used to incentivize customers to select a specific option. Conventional (next-day) delivery is usually free. The customer selects a delivery option based on his or her preferences. If a SDD option is selected, the service provider then assigns the order to a vehicle from the delivery fleet to pick up the goods at the warehouse and to deliver the goods to the customer.

2 Problem Definition

We call the problem the dynamic routing and pricing problem for same day delivery (DRPSDD). During a shift, a fleet of vehicles delivers goods from a depot to customers. The problem is on the operational level. Thus, the drivers have fixed shifts and are already paid. During the shift, customers request orders. These customers are unknown before the time of their order. For each ordering customer, a set of same-day delivery options, more specific, delivery deadlines is provided. For each delivery option, the provider presents a price. This price may differ for each customer and deadline. The customer selects a delivery option or rejects the same-day delivery option (and selects conventional delivery). This choice is based on the customer’s willingness-to-pay (WTP) for a deadline. In SDD, the WTP depends on the ordered product and the customer’s demographics. Thus, the WTP-values for a customer and different deadlines are correlated. If the customer selects a same-day delivery option, a vehicle picks up the order at the depot and delivers it within the delivery deadline. The objective is to determine a dynamic routing and pricing policy maximizing the expected revenue per shift.

3 Solution Method and Results

The challenges for the DRPSDD are manifold since it combines stochastic dynamic routing and dynamic pricing. We experience uncertainty in both customer requests and customer choice behavior. Particularly challenging is the instantaneous determination of suitable,
state-dependent prices with respect to the choice behavior, vehicle routing, and future customer requests. Suitable prices should generate revenue but should also incentivize customers to select options efficiently to fulfill (Agatz et al. 2013). A suitable pricing rule should therefore be state-dependent and should consider both the instant revenue and the impact of the fleet’s flexibility to generate future revenues. This impact is quantified in the opportunity costs meaning the difference in future revenue for the routing resource consumption in case the customer accepts an option or not. In this research, we present an anticipatory pricing and routing policy (APRP) incentivizing customers to select a delivery option with low opportunity costs. If the opportunity costs of an option are low, APRP offers an budget price for the delivery option. If they are high, APRP offers a higher price reflecting the opportunity costs. We determine suitable budget prices by policy search based on a restricted class of policies. To approximate the opportunity costs, we present a value function approximation (VFA, Powell 2011), an offline method of approximate dynamic programming (ADP). For each state and option, we approximate the opportunity costs based on a set of state features. These features reflect the fleet’s flexibility to serve future customers in case the option is selected. We compare the VFA with static pricing and conventional pricing methods based on geography and time for a variety of instance settings. Figure 1 shows the average improvement in revenue and number of customers of the policies compared to static pricing. APRP outperforms the benchmark policies significantly with respect to both revenue and number of same-day deliveries per day.

4 Contributions

Our contributions are as follows. This work is the first combining incentive schemes and dynamic delivery routing. The only work combining dynamic pricing with dynamic vehicle routing is presented by Figliozzi et al. (2007) and Topaloglu and Powell (2007). Both of these works do not consider SDD but different routing problems. For the DPPSDD, we present an anticipatory dynamic pricing and routing policy based on offline ADP. Our
policy provides suitable prices instantly and achieves excellent results in comparison to conventional pricing methods. Our work further presents the first offline ADP-method for a dynamic vehicle routing problem with temporal commitments, namely, delivery deadlines. The proposed VFA accounts for a fleet’s flexibility to efficiently serve future requests. The methodology may therefore be transferable to a variety of related problems with deadlines such as food delivery or dial-a-ride.

References


Tactical planning for Two-Tier City Logistics Systems under disturbances

Pirmin Fontaine
TUM School of Management, Technical University of Munich, Germany
Email: pirmin.fontaine@tum.de

Teodor Gabriel Crainic
School of Management, Université du Québec à Montréal, and CIRRELT, Canada

Ola Jabali
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

Walter Rei
School of Management, Université du Québec à Montréal, and CIRRELT, Canada

1 Introduction

The transportation of goods in urban areas is a complex activity and essential to the economic and social life of any city. It is also, however, a major contributor to significant nuisances, e.g., congestion, emissions, noise, and excessive consumption of fossil fuels. The complexity and impact of freight transportation in cities is further amplified by two heavy trends observed world-wide: increasing urbanization and e-commerce.

New organization and business models are proposed to address these issues under the name of City Logistics (CL). By consolidating flows in the city into the same vehicles, CL aims to improve the utilization of the means of transport and reduce their presence within the city. We focus on this CL strategy in this paper, aiming to provide a significantly enhanced methodology, compared to the state-of-the-art, for the planning of the corresponding operations, services, and possibly shared resources.

Statistical analysis shows that both public transport as well as road networks are facing congestion and unexpected delays because of disturbances. While nowadays congestion can be modeled through time-dependent travel times, disturbances are unexpected events with low probabilities that influence the travel time on a few arcs in the surrounding. However, the effects and probabilities vary, for example, between road networks and a tram lines. Therefore, the selection of transportation mode (e.g., cargo trams and trucks)
can significantly influence the service level of the system and a resilient plan, which assigns resources under consideration of disturbances, is important.

The contributions of this work are as follows: (1) We define a two-tier city logistics model which considers different transportation modes, disturbances, and resource management, (2) we propose an efficient solution method for solving the problem, and (3) we identify the effects of disturbances in such a system.

2 Problem definition and notation

We consider a schedule length which is divided into \( t = 1, \ldots, T \) periods. The city logistics network is defined as follows: Incoming goods arrive at a set of city distribution centers \( \mathcal{E} \), where they are consolidated, sorted and loaded into the urban vehicles. These urban vehicles transport the goods to set of satellites \( \mathcal{Z} \), from where the final distribution is done. Both, the city distribution centers and the satellites are divided into separate subsets \( \mathcal{E}^m \) and \( \mathcal{Z}^m \) for each considered mean of transport \( M \) (e.g., trucks, trams and subways).

We define \( T_m \) as the set of all available urban-vehicle types of the different means of transport \( m \). Then, the set of all urban-vehicle types is given by \( T \). Further, let \( n_{e\tau} \) be the fleet size at each CDC \( e \) and \( u_{c\tau} \) the corresponding capacity of urban-vehicle type \( \tau \). The capacity is further specified by the number of compartments is \( n_{c\tau} \) and the compartment capacity \( u_{c\tau} \). Each urban-vehicle has a unique depot \( e_{\tau} \in \mathcal{E} \), where it starts and finishes its operation at beginning and end of the planning horizon.

For each satellite \( z \), we consider three different types of capacities: (1) the number of urban vehicles for each period \( u^{\tau}_{zt} \), (2) the number of urban vehicles of transportation mode \( m \in M \) \( u^{m\tau}_{zt} \), and (3) total volume of goods assigned to satellite \( z \) in period \( t \) \( V_{zt} \). For trucks \( u^{m\tau}_{zt} \) is the actual number of trucks, while for trams or subways the number of cars in a tram can also be the limitation.

To reflect the stochastic nature of the disturbances, we consider a set of scenarios \( \Pi \). Each scenario \( \pi \in \Pi \) has a probability of \( p_{\pi} \). Then the time-dependent travel time of scenario \( \pi \) between two points \( i \) and \( j \) at time \( t \) in the network is defined by \( g_{ij}^{\pi}(t) \). The service time for loading and unloading an urban-vehicle \( \tau \) is \( h_{\tau} \).

The set of demands is defined as \( D \). In the first-tier, the destination of the demand is not the customer location but a satellite, from where the final distribution is done. The set of potential satellites is given by \( \mathcal{Z}(d) \subset \mathcal{Z} \). For each demand \( d \in D \), the customer location and a volume \( v_d \) are given. Let \( f_{de} \) be the assignment costs for selecting CDC \( e \in \mathcal{E} \) for demand \( d \in D \). Moreover, each demand can have a time-window: when it will be available at the CDC \( [a_d^o, b_d^o] \), and when it can be delivered to the destination \( [a_d^d, b_d^d] \).

As proposed by Fontaine et al. (2017), we use approximated final distribution costs for the second tier transportation \( s_{dzt} \) for demand \( d \) from satellite \( z \) in period \( t \). Besides
operating and handling costs also costs for disturbance through freight activities during the operation time of the service are included. However, if the first-tier service is delayed because of disturbances and time-windows are not met anymore, a non-linear penalty \( h(y) \) for the cumulative delay of all shipments by a service in a satellite. These penalties reflect the extra costs for faster shipments in the second tier, which can occur by choosing a faster more expensive transportation mode or by operating shorter tours.

The tactical planning problem aims to select a set of urban vehicle services out of the set of all available urban vehicle services \( \mathcal{R} \). Each service \( r \) starts at CDC \( e_r \in \mathcal{E} \), visits a set of satellites, and returns to the same CDC again. \( \sigma^r = \{ z_i^r \in \mathcal{Z}, i = 1, \ldots, |\sigma^r| \} \) defines the ordered set of visited satellites, such that if \( r \) visits satellite \( i \) before satellite \( j \) then \( i < j \). For a given service \( r \), an urban vehicle of type \( \tau_r \), which is of mode \( m_r \), operates service \( r \) and its associated costs are given by \( k_r \). The costs include again not only the operating costs of the route and unloading or loading activities, but also disturbance factors for freight activities during the operation time of the service. Departure times at CDCs and satellites and possible waiting and handling times are all included in the definition of \( \mathcal{R} \). To reflect the different compartments of the services \( r \), \( \mathcal{R}^C(r) \) defines the set of services, where each element reflects a service of a compartment. For services with one compartment, obviously \( |\mathcal{R}^C(r)| = 1 \). If the service \( r \in \mathcal{R} \) is operated, all its compartment services are operated.

### 3 Mathematical formulation

We introduce a new formulation for the tactical planning problem with stochastic travel times, which builds on the formulation of Fontaine et al. (2017). For this formulation, we use the same binary decision variable \( \rho_r \) that indicates if the urban vehicle service \( r \in \mathcal{R} \) is selected or not, and the binary decision \( x_{r^c,d,z} \) which takes the value of one, if demand \( d \in \mathcal{D} \) is assigned to compartment service \( r^c \in \mathcal{R}^C \) and satellite \( z \in \mathcal{Z} \) and zero otherwise. Using the introduced notation and the defined decision variables, we can formulate the problem as follows:

\[
\begin{align*}
\min \sum_{r \in \mathcal{R}} k_r \rho_r + \sum_{d \in \mathcal{D}} \sum_{r \in \mathcal{R}} \sum_{r^c \in \mathcal{R}^C(r)} \sum_{z \in \mathcal{Z}} (s_{d,z,t} + f_{d,e_r}) x_{r^c,d,z} + \sum_{\pi \in \Pi} \sum_{r \in \mathcal{R}} \sum_{z \in \mathcal{Z}} p_{\pi} h \left( \sum_{d \in \mathcal{D}} y_{\pi}^{d,z} \right) \\
\text{subject to} \\
\sum_{z \in \mathcal{Z}} \sum_{r \in \mathcal{R}} \sum_{r^c \in \mathcal{R}^C(r)} x_{r^c,d,z} = 1 & \quad \forall d \in \mathcal{D} \\
\sum_{z \in \mathcal{Z}} \sum_{d \in \mathcal{D}} v_{d,r^c,d,z} \leq u_{c,r}^c & \quad \forall r^c \in \mathcal{R}^C(r), r \in \mathcal{R} \\
\sum_{r \in \mathcal{R}(t,\tau,e)} \rho_r \leq n_{e\tau} & \quad \forall \tau \in \mathcal{T}, e \in \mathcal{E}, t = 1, \ldots, T
\end{align*}
\]
\[
\sum_{t'=t-h+1}^{t} \sum_{r \in R(z,t')} \rho_{r} \leq u_{zt}^r \quad \forall z \in Z, t = 1, \ldots, T \tag{5}
\]

\[
\sum_{t'=t-h+1}^{t} \sum_{r \in R(z,t'),m} \rho_{r} \leq u_{zt}^m \quad \forall z \in Z, m \in M, t = 1, \ldots, T \tag{6}
\]

\[
\sum_{r \in R(t,z)} \sum_{r_c \in R^C(r)} \sum_{d \in D} v_d x_{r_c,d,z} \leq u_{zt}^V \quad \forall z \in Z, t = 1, \ldots, T \tag{7}
\]

\[
\sum_{r_c \in R^C(r)} t_{z}^{r_c} x_{r_c,d,z} \leq b_d^d + y_{d rz}^\pi \quad \forall r \in R, z \in Z, d \in D, \pi \in \Pi \tag{8}
\]

\[
\rho_{r} \in \{0, 1\} \quad \forall r \in R \tag{9}
\]

\[
x_{r_c,d,z} \in \{0, 1\} \quad \forall d \in D, r_c \in R^C(r), z \in Z \tag{10}
\]

\[
y_{d rz}^\pi \geq 0 \quad \forall d \in D, r \in R^C(r), z \in Z, \pi \in \Pi \tag{11}
\]

The objective function (1) is minimizing the service costs, plus the approximated final distribution costs, plus the CDC selection costs. Moreover, the recourse costs which are caused by delays are added to the total costs. Constraint (2) ensures that each item is assigned exactly to one compartment. The demand capacities are ensured by linking constraint (3). Constraints (4) limit the maximum number of available vehicles of one type in a CDC. Constraints (5) and (6) limit the number of urban vehicles at a satellite in each period in total and per transportation mode respectively. Constraints (7) limit the maximum amount of demand which can be unloaded or loaded at a satellite in each period. Finally, constraints (8) compute the delay of a demand at a satellite and a service.

We approximate the delays using discrete intervals. Therefore, the non-linear penalty function \( h(y) \) can be approximated by a piece-wise linear convex function.

### 4 Conclusion

We will present a new tactical planning model for CL under disturbances. We are developing an exact solution method based on Benders decomposition to solve the problem efficiently. The full details of the model, method and numerical experiments will be presented at the conference. Other than the efficiency of the algorithm, we are interested in evaluating the selection of different transportation modes depending on the stochasticity and demand type in a CL system.

### References

Flexibility Schemes for Attended Home Deliveries

Charlotte Köhler
Jan Fabian Ehmke
Management Science Group,
Otto von Guericke University, Magdeburg, Germany

Ann Melissa Campbell
Department of Management Sciences
University of Iowa, Iowa City, United States

Email: charlotte.koehler@ovgu.de

1 Introduction

Many of the products that are ordered online require delivery to the home or office when the customer is present. Examples include groceries, appliances, furniture, and restaurant meal deliveries. For these attended deliveries, a retailer offers a selection of time windows on their website, and customers choose one of these windows. Creating the set of delivery time windows is challenging, since customers want convenient short time windows, but short time windows can significantly restrict the ability to accept future requests and decrease the flexibility of the route plan and the route plan’s efficiency (see [1] and [2]).

Deciding which time windows can be used to serve a particular customer is also challenging. Solving a Vehicle Routing Problem with Time Windows for every incoming delivery request and time window option is simply too time consuming and due to dynamic customer requests, a short-term good solution may be useless in the end. The literature offers various approaches for order acceptance of attended home deliveries [1], e.g. dynamic customer acceptance using a fast insertion heuristic to check whether the offering of a particular time window is feasible [3]. However, these approaches do not consider the flexibility of a route plan in the provision of customer-specific time window offerings.

In this work, we introduce the idea of flexible time window management. We consider two time window lengths, long and short windows. The largest number of deliveries are possible with long windows, but customers prefer the better service of the short windows. Here, we still want to maximize the number of deliveries made, but offer short windows to customers at times where it will not restrict the ability to accept additional requests. This flexibility in time window offerings should improve the service to customers while
remaining cost-effective. We present several schemes to help make this time window
offering decision based on customer characteristics (location and time window preferences)
as well as characteristics of the evolving route plan (e.g., available capacity). Our schemes
are easy to adapt by an online retailer and focus on quick and simple decision rules. We
investigate the presented schemes with a case study for an online supermarket.

2 Flexible Time Window Management

A customer on a retailer’s website will be offered a customer-specific set of time windows.
The time windows will have no delivery costs, and at least one time window will always
be offered to a customer if the request can be inserted feasibly within our tentative route
plans. We decide for each customer if a short or long time window (or no time window) is
offered for each time in the day. While long time windows may be less likely to be accepted
by a customer, they provide greater flexibility for accommodating future requests.

More formally, for each request \( j \), we want to create an offer set \( O_j \) that is based on
spatio-temporal customer as well as route plan information. We assume each customer has
a required service time \( u_j \). We consider two sets \( A \) and \( B \), each containing time windows
that are non-overlapping and consecutive. Set \( A \) contains short time windows with a
length of \( a \), and set \( B \) long time windows with a length of \( b \), respectively. The start and
finish time of each time window are denoted \( a_{i,j}^S \in A, b_{j}^S \in B \) and \( a_{i,j}^F \in A, b_{j}^F \in B \). Our
algorithm maintains a tentative route plan for each vehicle \( v \in V \) based on the already
accepted requests \( C \{c_1...c_q \} \). Our algorithm creates the time window offer set \( O_j \) for each
potential insertion point in the tentative routes in three steps:

I. Compute Feasible Time Windows. We use an insertion-based heuristic as
presented in [3] to evaluate the feasibility of inserting the new request \( j \) within our tentative
route plan. Since we want to find all feasible delivery time window options we can display
to a customer, we compute time spans based on the concept of slack introduced by [4]
that reflect the range start of service times for request \( j \) at the insertion position between
customer \( i \) and \( i+1 \) on vehicle \( v \). We define the earliest possible time service can begin as
\( e_{i,i+1}^v = e_i^v + u_i + t_{i,j} \) and the latest possible time service can begin as
\( l_{i,i+1}^v = l_{i+1}^v - u_j + t_{j,i+1} \).

If the size of the resulting time span \( s_{i,i+1}^v = l_{i,i+1}^v - e_{i,i+1}^v \) is equal or larger than zero,
the insertion position is feasible. For each feasible insertion position, we create sets that
contain all feasible time windows \( a_m \in A \) and \( b_n \in B \), formalized as follows:

\[
A_{i,i+1}^w = a_m \mid a_m^S \leq e_{i,i+1}^v \leq a_m^F \quad \text{or} \quad a_m^S \leq l_{i,i+1}^v \leq a_m^F \quad \text{or} \quad (e_{i,i+1}^v \geq a_m^S \& a_m^F \leq l_{i,i+1}^v)
\]

\[
B_{i,i+1}^w = b_n \mid b_n^S \leq e_{i,i+1}^v \leq b_n^F \quad \text{or} \quad b_n^S \leq l_{i,i+1}^v \leq b_n^F \quad \text{or} \quad (e_{i,i+1}^v \geq b_n^S \& b_n^F \leq l_{i,i+1}^v).
\]

II. Offer Tight Time Windows. To decide which of the feasible time windows to
offer, we present four customer acceptance schemes that differ in the way they consider
spatio-temporal information of the tentative route plans and create the time window of-
ferring sets \( A_{i,i+1}^o \) and \( B_{i,i+1}^o \). The schemes are as follows.
**Long**⇒**Short** (LS) With the LS scheme, we maintain a high level of routing flexibility in the beginning of the booking process, so we offer (feasible) long time windows only to the early arriving customers. We switch to offering (feasible) short time windows when a large portion of the route plan is defined. We quantify this by measuring the current utilization of our service capacity and compute how much of the available service time $T$ for the set of available service vehicles $V$ has already been consumed by the set of accepted customers. When a certain percentage $x^{LS}$ of the available time has been utilized, we offer only short time windows for that insertion position. More formally,

$$x^{LS} < (t_{0,1} + \sum_{i=1}^{q-1} (t_{i,i+1} + u_i) + t_{q,0})/(|V| \ast T)$$ offer $A_{i,i+1}^{nu} = A_{i,i+1}^{nu} = B_{i,i+1}^{nu} = \emptyset$, else offer $A_{i,i+1}^{nu} = \emptyset, B_{i,i+1}^{nu} = B_{i,i+1}^{nu}$.

**Short**⇒**Long** (SL) The SL scheme is quite similar to the LS scheme, but begins with offering short time windows in the booking process:

$$x^{SL} \geq (t_{0,1} + \sum_{i=1}^{q-1} (t_{i,i+1} + u_i) + t_{q,0})/(|V| \ast T)$$ offer $A_{i,i+1}^{nu} = A_{i,i+1}^{nu} = B_{i,i+1}^{nu} = \emptyset$, else offer $A_{i,i+1}^{nu} = \emptyset, B_{i,i+1}^{nu} = B_{i,i+1}^{nu}$.

**Travel Time (TT)** With the TT scheme, we want to examine offering sets of short time windows only to customers that are located in the vicinity of customers in our tentative route plan. To this end, we check if a new request is in the neighborhood of an existing customer. If travel time from an accepted customer to the location of the new request is within a threshold value $x^{TT}$, the new request should not impact the flexibility of the tentative route:

$$x^{TT} \geq t_{i,j} \text{ or } x^{TT} \geq t_{j,i+1}$$ offer $A_{i,i+1}^{nu} = A_{i,i+1}^{nu} = B_{i,i+1}^{nu} = \emptyset$, else offer $A_{i,i+1}^{nu} = \emptyset, B_{i,i+1}^{nu} = B_{i,i+1}^{nu}$.

**Insertion Time & Span (TS)** Since routes are evolving during the booking process, customer acceptance exclusively based on myopic insertion costs may be misleading. Hence, the TS scheme combines information about the proximity of a new request with information on how likely the affected part of the route plan is subject to change (size of the span). More formally,

$$x^{TS} \geq t_{i,j} + t_{j,i+1} - t_{i,i+1} \text{ and } x^{TS} \geq s_{i,i+1}$$ offer $A_{i,i+1}^{nu} = A_{i,i+1}^{nu} = B_{i,i+1}^{nu} = \emptyset$, else offer $A_{i,i+1}^{nu} = \emptyset, B_{i,i+1}^{nu} = B_{i,i+1}^{nu}$.

**III. CREATE OFFER SET.** In the last step of the algorithm, we join all time window options from step II and create a single offer set $O_j$ to display to the new request $j$:

$$O_j = A_{i,i+1}^{nu} \cup \ldots \cup A_{i,i+1}^{nu} \cup B_{i,i+1}^{nu} \cup \ldots \cup B_{i,i+1}^{nu}.$$

We repeat this process for each insertion point on the route for each vehicle $v \in V$ to get a final set $O_j$. We assume that each requesting customer has a randomly chosen preference $\rho$ for either early ($\rho \leq 0.5$) or late ($\rho > 0.5$) time windows and define the early time windows as those that occur in the first half of the day and the remaining as late time windows. If the final offer set $O_j$ contains time windows within the preferred time of the
day, short time window options are always favored by a customer. If only a long time
window is offered, the requesting customer will accept it with a rate of $p$ or cancel the
booking process.

3 Insights & Outlook

We investigate the effectiveness of these schemes by a simulation of the booking process
for an online supermarket in Berlin, Germany. We set the lengths of short time windows
to 30 minutes and long windows to 240 minutes. We use real travel times provided by
OpenStreetMap and generate 100 customer requests for 1000 independent runs.

Table 1 shows an excerpt of the results based on the average of all runs for the four
presented schemes tested with the specified values for $x^{LS}$, $x^{SL}$, $x^{TT}$ and $x^{TS}$. When
customers are more likely to accept long time windows ($p = 75\%$), there is a clear trade-off
between serving many customers and promising many short time windows. For example,
the LS scheme can accept a maximum of 72 customers in total, but five within a short
time window for a threshold of $x^{LS} = 90$. For a threshold of $x^{LS} = 10$, we can only serve
64 customers in total but increase the number of customers accepted within a short time
window to 57. It is also interesting to see how $x^{SL}$ reveals that early acceptance with
short time windows can impact the number of total customers accepted, as indicated by
lower totals than with $x^{LS}$. When the acceptance rate of long time windows is set to
$p = 50\%$, the higher the chances of a customer canceling the booking process, and hence
fewer customers can be accepted. Across both probabilities, the TT scheme does a good
job of accepting a total number of deliveries close to the maximum while offering many
short time windows to customers.

In our presentation at the conference, we will provide more detailed results for the four
flexibility schemes, tested with balanced and imbalanced demand for time windows (based
on real order data). We show that our schemes work well, especially when more customers
are willing to accept long time windows or when demand is imbalanced. Future work
will consider incorporating a more sophisticated customer demand model that anticipates
real-world customer locations and preferences.

<table>
<thead>
<tr>
<th>$x^{LS}$</th>
<th>90</th>
<th>70</th>
<th>50</th>
<th>30</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>72 (5)</td>
<td>70 (18)</td>
<td>68 (30)</td>
<td>66 (44)</td>
<td>64 (57)</td>
</tr>
<tr>
<td>$x^{SL}$</td>
<td>10</td>
<td>30</td>
<td>50</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>70 (7)</td>
<td>68 (20)</td>
<td>66 (32)</td>
<td>64 (45)</td>
<td>62 (57)</td>
</tr>
<tr>
<td>$x^{TT}$</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>72 (0)</td>
<td>69 (19)</td>
<td>68 (37)</td>
<td>67 (51)</td>
<td>66 (57)</td>
</tr>
<tr>
<td>$x^{TS}$</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>72 (0)</td>
<td>72 (11)</td>
<td>68 (35)</td>
<td>66 (50)</td>
<td>65 (55)</td>
</tr>
</tbody>
</table>

Table 1: Results (Excerpt): I. Total accepted Customers (II. Accepted Customers within
Short Time Window)
References


Planning Deliveries in Disaster Relief by Truck and Drone

Alena Otto  
Department of Management Information Science,  
University of Siegen, Siegen, Germany  
alena.otto@uni-siegen.de

Stefan Poikonen  
Robert H. Smith School of Business,  
University of Maryland, College Park, USA  
Email: spoikone@math.umd.edu

Bruce Golden  
Robert H. Smith School of Business,  
University of Maryland, College Park, USA  
bgolden@rhsmith.umd.edu

1 Introduction

In the aftermath of disasters, some roads may become impassable. Delivery of supplies to hospitals, shops, and households as well as search-and-rescue operations may be hindered. Moreover, the state of the roads is often unknown. In such situations, constructing robust routes, which take into account the probability that each road segment is passable, is desirable.

Additionally, the use of unmanned aerial vehicles (UAVs), also called drones, to supplement a truck-only fleet may also be considered. Drones have been already regularly used in disaster management for surveillance, mapping and search operations [3]. In addition, drones may perform deliveries. Modern parcel delivery drones can carry a 3-kg (6.5-pound)-package for a 15 km distance [1, 5], whereas special disaster relief drones, such as the Windhorse’s Perceo, will be able to deliver up to 50 kg (110 pounds) within the 40 km reach [2]. Because of the limited flight duration of drones, it is advantageous to let a truck serve as a mobile depot for the drone. The drone can visit the truck to pick up a package and to replenish its energy by replacing or recharging the battery [4] (see Figure 1 for illustration).

In this presentation, we study operations of a truck-and-drone tandem in disaster relief, when the state of the roads is unknown and a set of supplies packed in packages of standard size has to be delivered to the customers. We provide a theoretical analysis of several intuitive deployment strategies of the truck and the drone. We also construct a flexible heuristic framework to generate routes under varying operational assumptions. This framework accounts for the uncertain status of the passability of the roads and the information gained as the vehicles progress.
2 General problem statement

Undirected graph $G = (D, V, E, c^t, c^d, p)$ describes a street network. The set $D$ is the vertex set of the street network, $V \subseteq D$ is the set of customers, each demanding 1 homogeneous package, $v_0 \subseteq D$ denotes the origin depot, $E$ is the edge set. We use $c^d(i, j)$ and $c^t(i, j)$ to denote traveling times of the drone and the truck along an edge $(i, j) \in E$.

The set of impassable edges is $\mathcal{E} \subseteq E$. However, we do not know $\mathcal{E}$ when truck and drone operations begin. Note that the truck cannot traverse a damaged edge, but the drone can safely fly over it at its original speed. Edge labels $p(i, j)$ describe the probability that an edge is impassable. To learn the status of edge $(i, j) \in E$, the truck or the drone must visit one of its adjacent nodes $i, j \in D$. We may model limited visibility along long edges by subdividing the edge into multiple segments (see Figure 2).

We look for feasible walks for the truck and the drone:

- The truck and the drone start from the depot, traverse only edges in $E$, and have to return to the depot in the end of their operations.
- The truck walk should not contain damaged edges.
- Since the truck has to park in a safe location during the launch or land of the drone, we assume that the drone can only launch from the truck or return to the truck in one of customer nodes $V$. 

Figure 1: Illustration of deliveries by a truck-and-drone tandem. Straight arrows depict the truck's trajectory, heading dashed arrows depict UAV's trajectory, small homes depict customer locations, and the larger facility corresponds to the depot.

Figure 2: Illustrative example: road segments between customers $v_1$, $v_2$, and $v_3$. Red nodes depict customer locations, non-customer nodes are black. The road segment between $v_1$ and $v_2$ has low visibility, whereas the way to $v_3$ is fully visible in node $v_2$. 

180
The drone can carry only one package at a time, therefore it has to visit either depot $v_0$ or the truck between each two consecutive delivery nodes in its walk.

Walks of the truck and the drone have to be synchronized in their launch and return nodes. Two subsequences of nodes—a subsequence of the truck walk starting at some launch node and ending at the next return node, and a subsequence of the drone walk between the corresponding launch and return nodes—form an operation. During an operation, the truck may have to wait for the drone or, vice versa. Between two consecutive operations, the truck and the drone travel together at the speed of the truck with the drone parked on the truck.

Consider the example in Figure 1. The truck and the drone travel synchronized feasible walks $\pi^t := (v_0, \overline{v}_1, \overline{v}_3, \overline{v}_4^L, \overline{v}_6^R, v_0)$ and $\pi^d := (v_0, \overline{v}_1^L, \overline{v}_2, v_4^R, v_5, v_6^R, v_0)$, respectively. Subscripts $L$ and $R$ mark launch and return nodes, respectively, and delivery nodes are overlined. The truck and the drone perform two operations $\{ (v_1, v_3, v_4), (v_1, v_2, v_4) \}$ and $\{ (v_6, v_0), (v_6, v_5, v_6) \}$. In the first operation, which has duration of $\max \{ c^t(v_1, v_3) + c^d(v_3, v_4), c^d(v_1, v_2) + c^t(v_2, v_4) \} = \max \{ 4 + 2, 3 + 2 \} = 6$, the drone has to wait for the truck for 1 time unit in node $v_1$. In the second operation of $\max \{ 0, 2 \} = 2$, the truck has to wait for the drone. The makespan of the truck-and-drone tour is $2 + 6 + 2 + 2 + 2 = 14$ time units.

Our objective is to find feasible synchronized walks of the truck and the drone that deliver packages and minimize the (expected value of) the makespan.

3 Solution Method

We first analyze the theoretical competitive ratios of several intuitive solution strategies by comparing their result with the optimal objective value in the full information case. These inspired a number of heuristic solutions.

In each variant of this problem, the basic structure of the heuristic solution method remains unchanged. The key idea is that every time we obtain new information, we re-simulate the state of the network. In simulating the network, we find a way to ‘price’ each edge, according to the expected rerouting cost, if that edge is not passable. We may describe the overall structure of our solution method as follows.

1. For each edge $(i, j)$:
   - For $\text{iter} = 1$ to $\text{MAX}$:
     - Randomly simulate the state of all edges, except $(i, j)$, according to probability set $p$.
     - Temporarily assume $(i, j)$ is undamaged.
     - For each destination $T$, compute the shortest path from $i$ to $T$, $\text{distNoDam}(i, T)$.
     - Temporarily assume $(i, j)$ is damaged.
     - For each destination $T$, compute the shortest path from $i$ to $T$, $\text{distDam}(i, T)$.
     - Set $\delta_{\text{iter}}(i, T) = \text{distDam}(i, T) - \text{distNoDam}(i, T)$.
   - Set $\text{ExpectedRerouteCost}(i, j) = \sum_{\text{iter}=1}^{\text{MAX}} \delta_{\text{iter}}(i, T)/\text{MAX}$

2. Construct modified edge weights $c_+^i$, such that $c_+^i(i, j, T) = c^i(i, j) + (1 - p(i, j)) \times \text{ExpectedRerouteCost}(i, j, T)$
3. For each $v_a, v_b \in V$, compute the shortest path over the graph with vertex set $D$, edge set $E$, and cost of traversing an edge $(i, j)$ as $c_+(i, j, v_a)$.

4. Solve a truck-and-drone routing problem, utilizing the expected distances between customer sites from Step 3. (The details of this step vary depending on exact model assumptions.)

5. Move the vehicles forward according to the solution from Step 4, until new information is realized at a node. Update $p$ according to the newly realized information.

6. If we have completed all deliveries, quit algorithm. Otherwise, go to Step 1.

To evaluate the heuristic algorithms, we compare the following: 1) the average objective value under full prior information; 2) the average objective value of our heuristic solution; and 3) the average objective value if we ignore the chance that edges in our path are impassable, and only re-route when our original path is blocked.

![Figure 3: Illustration of the truck walk for our problem. In red are segments that have already been traversed by a truck. In blue is the segment currently being traversed by the truck. Thicker black or grey dotted lines indicate the planned path of the truck. Thinner black or grey dotted lines indicate possible alternate routes. Pure black indicates a segment that is certainly passable, whereas lighter grey colors are unlikely passable. At left, is the state of the problem earlier in the route. At right, we see the truck has discovered a segment was passable, so the planned route has been updated according to this new information.](image)

4 Next steps

We are currently evaluating our algorithms in computational experiments on data sets motivated by the infrastructure of real-world cities. We are also working on effective integer programs for useful special cases of the general problem. The results of these studies will be included in our presentation at the Odysseus workshop.
References


Operational Challenges of a Food Bank in a Gleaning Network

Ohad Eisenhandler, Michal Tzur

Department of Industrial Engineering
Tel Aviv University, Tel Aviv, Israel

Email: ohadeize@mail.tau.ac.il, tzur@eng.tau.ac.il

1 Introduction

Gleaning is the act of collecting crops from fields that are not economically profitable to harvest. Farmers usually allow their fields to be gleaned when the produce cannot be sold in the market, because of either low revenues or cosmetic impairments (that do not damage its nutritional quality). In these cases, unless the crops are collected, they are left to rot or to be plowed under. Consequently, it is more cost-effective for farmers to have their crops gleaned than to pay pickers to go back through the fields; tax benefits and social responsibility are further motivations.

The practice of gleaning dates back to Biblical times, when crops were directly gleaned by needy families themselves. Nowadays, a gleaning network is typically managed by a food bank, which is a nonprofit charitable organization that acts as the central decision maker of the network. The food bank uses a fleet of vehicles and employs specialized gleaners in order to collect produce from the donating farmers. The collected produce is sent back to a logistic center, where in the next day it is sorted for quality and organized in distribution boxes. It remains in proper storage conditions in the logistic center until it is delivered to one of the welfare agencies in the food bank's network, which serve it to end-beneficiaries. The problem we study is motivated by the activity of Leket Israel (Hebrew for Gleaning for Israel), which is one of two major food banks in the country, with a gleaning network serving 175,000 individuals through 200 welfare agencies nationwide. Similar gleaning networks operate in the U.S. and in Europe.

The problem faced by the food banks that manage such gleaning operations shares its characteristics with several classes of problems that are familiar from the vehicle routing literature, such as pickup and delivery problems, inventory routing problems and routing problems with profits. However, it differs from them in the objective function, since the primary goal of the food bank is humanitarian, i.e., reducing hunger, rather than creating
financial profits. From the point of view of humanitarian logistics, the activity of food banks has attracted a considerable amount of attention in the literature in recent years, e.g., [1], [2], [3] and [4]. However, these studies assume that the food can be immediately distributed from the suppliers to the delivery sites, which is not the case for gleaning activities. As a matter of fact, research on gleaning from an operations research perspective is quite scarce: in [5], a dynamic model was formulated for the volunteer staffing problem which finds the control policy that maximizes the total volume gleaned (minus penalties for turning down donations). The strategic problem of determining which fixed schedule slots to offer every week for potential gleaning requests was studied in [6]. By simulating the stochastic processes of both supply and labor, they aimed to maximize the total expected volume gleaned.

The aim of the current study is to close the gap w.r.t. the logistic challenges of food banks that manage gleaning networks. Since the gleaned produce cannot be immediately distributed, these operations give rise to a problem with special characteristics that have not been considered in previous research about the activity of food banks: (1) separation of vehicle routes to pickup (backhaul) vs. delivery (linehaul) lines; (2) multi-period planning, which considers the fact that supplies do not have to be collected on a specific day, but require time-consuming processing prior to delivery and have a consumption deadline. We also use a humanitarian objective function, which best captures the goals of the food bank.

2 Statement of the Problem

The setting that was described above leads to the following logistic problem over a planning horizon of $T$ days. Every day, farmers inform the food bank of gleaning opportunities, which have to be performed within a given number of days. The amount available for collection, the number of gleaners required for the task, and a spoilage due date, following which the produce becomes inedible and can no longer be supplied, are all given. The crops are collected by capacitated vehicles, and the daily work time of drivers is limited. We assume that a vehicle that is assigned to a gleaning opportunity is occupied for an entire workday, since produce is continuously loaded into it.

After collection, the produce is brought to the depot, where it undergoes processing the following day, and it is ready for distribution two days after it is harvested. Every day, vehicles that are not assigned to gleaning may be used for distribution of produce from the depot to welfare agencies. Each agency is assigned with a fixed delivery quantity per visit (which is determined by the food bank in advance, proportionally to the number of individuals served by each agency), and may only be visited if this quantity can be supplied. Certain agencies may
not be operating every day of the week, and may not be supplied on such days. In addition, a record of previous visits starting from a specific epoch is kept.

The food bank uses an in-house fleet of vehicles and workers, professional gleaners and drivers. However, if on a certain day, larger amounts of vehicles or gleaners are required, a weekly outsourcing budget can be used to increase the amounts of these resources.

Given this input, the problem is to determine which gleaning opportunities to accept and on what day to perform them; and to design vehicle routes for the distribution of food that is available in the depot to welfare agencies. The problem is solved in a rolling horizon framework, such that the solution for the current day is implemented and used to update the service history and the amount of food available in the depot for the following workday.

3 The Objective Function and the Solution Method

While logistic problems in many supply chains amount to minimizing costs or maximizing profits, the setting of the problem leads to different goals that need to be achieved. Specifically, financial considerations are taken into account in the budget constraint mentioned above, and thus give rise to other considerations that need to be included in the objective function. Previous work on food banks suggested that the objective function should promote both effectiveness, i.e., getting as much food as possible to agencies, and equity, i.e., maintaining fair allocations among the different agencies w.r.t. the number of individuals each of them serves. Specifically, a novel objective function was presented and analyzed in [2], with the aim of balancing the tension between these two goals, w.r.t. the allocation of food to all agencies. The measure of equity they used was the equity version of the well-known Gini coefficient, which can be computed as \( 1 - \frac{\sum_{i \in D, i < j} |q_i Y_i - q_j Y_j|}{\sum_{i \in D} Y_i} \), where \( D \) is the set of agencies, \( q_i \) is the share of population served by agency \( i \in D \), and \( Y_i \) is its allocation. The authors showed that by multiplying this expression by the measure of effectiveness, i.e., \( \sum_{i \in D} Y_i \), a linear expression, \( \sum_{i \in D} Y_i - \sum_{i, j \in D, i < j} |q_i Y_i - q_j Y_j| \), which promotes both goals adequately, can be obtained.

However, in our problem the source of inequity is the number of delivery visits to each agency rather than the delivery quantities, since the latter are pre-determined, unlike in [2]. As a consequence, it is advisable to balance the total number of visits to all agencies over the entire planning horizon, and its equity w.r.t. to the different agencies. More formally, we define the decision variable \( V_i \) as the number of visits to agency \( i \in D \). Then, an appropriate objective function for our problem is the following: \( \sum_{i \in D} V_i - \frac{1}{|D|} \sum_{i, j \in D, i < j} |V_i - V_j| \). This function aims
to simultaneously promote effectiveness w.r.t. to the total number of visits to all agencies, as well as fairness in the dispersion of them across agencies. It can be shown that the problem described, including this objective function, is NP-Hard.

We develop a solution method for the problem, consisting of two phases. In the first phase, the day in which each gleaning opportunity is performed (or the decision to reject it) is determined by dynamic programming. The second stage determines the distribution routes for each vehicle (that is not used for collection of produce) on each day in the planning horizon. This is done by heuristically choosing a set of routes, generated in an offline pre-processing stage, that do not violate the time and capacity constraints. The solution method will be tested based on real life instances provided by the food bank we are cooperating with.

Acknowledgments

This research is supported by the Israel Science Foundation (ISF) grant no. 463/15 and by the Manna Center Program for Food Safety and Security at Tel Aviv University.

References

Optimizing access to drinking water for remote populations affected by the Nepal earthquake

Selene Silvestri
Department of Logistics and Operations Management,
HEC Montréal, Montréal, Canada
Email: selene.silvestri@hec.ca

Gilbert Laporte
Department of Management Sciences,
HEC Montréal, Montréal, Canada

Marie-Ève Rancourt
Department of Logistics and Operations Management,
HEC Montréal, Montréal, Canada

1 Introduction

This study is a collaborative work with three National Red Cross Societies (Nepal, NRCS; Austria, AutRC; and Switzerland, SRC). The purpose of this work is to propose an optimization tool for the water network rehabilitation (CWNR) of remote populations in Nepal. Every year earthquakes, landslides and various other disasters cause severe casualties, deaths and damages in Nepal. On 25 April, 2015, a 7.8 magnitude earthquake killed more than eight thousand people and destroyed hundreds of thousands of houses. The water, sanitation and hygiene (WASH) sector also suffered considerable damages. More than 1.1 million people were left without access to protected water sources, due to the destruction of water supply systems (WSSs) across 14 districts [1]. Many of the most affected areas were rural, with some of them difficult to reach (remote). After the initial emergency response phase, a recovery plan targeting the WASH sector was established by the Red Cross Societies. As part of this program, the rehabilitation of 18 damaged WSSs was planned by the AutRC. In this context, its main goal was to locate community water taps (WTs) such that people can access them according to the NRCS WASH technical standards. These standards fix the maximum horizontal and vertical distances between households (HHs) and WTs. In exceptional cases, these standards can be widened.

In this paper, we aim to propose a solution to restore the WSSs in two Village Development Committees (VDCs) of Nepal, i.e., locate WTs and connect them to the water
sources (pipelines). The primary objective consists of meeting the AutRC’s main goal, while minimizing the number of WTs. The secondary objective consists of minimizing the cost to connect the WTs to the pipeline network. This collaborative work aims at developing a well-documented systematic approach to support and facilitate decision making for humanitarian WASH programmes. We assume that all HHs in the region of interest (RoI) have no access to drinking water, and that the water sources considered during the rehabilitation plan are the only ones available in these two VDCs. Moreover, since the WSS in the RoI is a gravity-fed system, the elevation of the water sources, potential WT locations, and HHs have to be taken into consideration. Data provided by the AutRC and satellite imagery are used to determine the parameters of the mathematical model that represents the problem at stake. Due to the size of this problem, we propose a matheuristic approach to solve it.

2 Mathematical model

The community water network rehabilitation problem (CWNRP) is defined on a directed graph $G = (V, A)$, where $V$ is the node set and $A$ is the arc set. A feasible solution for the CWNRP is a set of spanning arborescences (arborescence forest), connecting the water sources to the WTs. WTs are located on the vertices of the forest, however some of the vertices only represent a branch in the WSS. The graph $G$ is obtained by dividing the RoI using a grid of equal size cells and associating each cell to a node of the graph. Let $H \subseteq V$ denote the set of nodes (cells) containing at least one HH, and let $L \subseteq V$ denote the set of potential locations for WTs. If $(\Delta, \Sigma)$ represent the maximum horizontal and vertical distances of the WASH standards, and $(\bar{\Delta}, \bar{\Sigma})$ the widened distances allowed in some exceptional cases, we consider a cell to be a potential WT location when it is within the horizontal radius $\bar{\Delta}$ of at least one HH. Figure 1 depicts an illustration of the RoI in this context. We can observe that $H \subseteq L$ and that a cell $l \in L$ might be a non feasible location due to its vertical distances from all HHs. The set of nodes $N$ also contains cells of the RoI where only a branch in the WSS can be installed. With $\delta_{hl}$ representing the
horizontal distance from $h$ to $l$ and $\sigma_{hl}$ the vertical distance between the highest and the lowest cell in the path from $h$ to $l$, the set $E$ contains all arcs $(h, l)$ such that $\delta_{hl} \leq \bar{\Delta}$ and $\sigma_{hl} \leq \bar{\Sigma}$. Moreover, if $\epsilon_v$ represents the elevation of $v \in V$, the set $E$ also contains all arcs $(v, u)$ such that $\epsilon_v \geq \epsilon_u$.

For each $l \in L$, let $w_l$ be a binary variable equal to 1 if and only if a water tap is located in $l$. For each $h \in H$, $l \in L$, let $s_{hl}$ be a binary variable equal to 1 if and only if HHs in $h$ are assigned to the water tap in $l$, and let $t_h$ be a binary variable equal to 1 if and only if $h$ remains isolated. For each $v \in V$, let $z_v$ be a binary variable equal to 1 if and only if a branch in the WSS is installed in $v$. Let $P$ represent the set of all water sources available. For each $p \in P$, $v \in V$, the binary variable $y^p_v$ indicates whether cell $v$ receives water directly from $p$, while the non-negative flow variable $f^p_v$ represents the amount of water from $p$ to $v$. For each $v, u \in V$, let $x_{vu}$ be a binary variable equal to 1 if and only if there is flow of water from $v$ to $u$, and let the non-negative variable $f_{vu}$ indicate the corresponding amount of water. The problem can be formulated as follows.

minimize $z = \sum_{l \in L} \eta_l w_l + \sum_{v, u \in V} \nu_{vu} x_{vu} + \sum_{h \in H} \sum_{l \in L} \Theta_{hl} s_{hl} + \sum_{h \in H} \Phi_h t_h$ \hspace{1cm} (1)

subject to

\[
\begin{align*}
\sum_{l \in L} s_{hl} &= 1 - t_h & h \in H \\
&\leq w_l & h \in H, l \in L \\
\mu w_l &\leq \sum_{h \in H} \gamma_{hl} s_{hl} & l \in L \\
w_l &\leq \sum_{u \in V} x_{ul} + \sum_{p \in P} y^p_l & l \in L \\
z_v &\leq \sum_{u \in V} x_{uv} + \sum_{p \in P} y^p_v & v \in V \\
x_{ul} &\leq w_l + z_l & l \in L, u \in V \\
x_{uv} &\leq z_v & v \in V \setminus L, u \in V \\
w_l + z_l &\leq 1 & l \in L \\
x_{vu} + x_{uv} &\leq 1 & b v, u \in V \\
\sum_{u, v \in V} x_{uv} &= \sum_{l \in L} w_l + \sum_{v \in V} z_v - \sum_{p \in P} \sum_{v \in V} y^p_v & V \subseteq V, |V| \geq 2 \\
\sum_{u, v \in V} x_{uv} &\leq \sum_{v \in V \setminus \bar{V}} z_v + \sum_{l \in (V \setminus L) \setminus \bar{V}} w_l & V \subseteq V, |\bar{V}| \geq 2 \\
\sum_{v \in V} f^p_v &\leq \varphi_p & p \in P \\
y^p_v &\leq f^p_v & v \in V, p \in P \\
f^p_v &\leq \varphi_p y^p_v & v \in V, p \in P \\
\sum_{u \in V} f_{uv} + \sum_{p \in P} f^p_v - \sum_{r \in V} f_{vr} &= 0 & v \in V \setminus L
\end{align*}
\]
\[ \sum_{u \in V} f_{ul} + \sum_{p \in P} f^p_l - \sum_{r \in V} f_{lr} = \omega \sum_{h \in H} \gamma_h s_{hl} \quad l \in L \]  

\[ w_l, s_{hl}, t_h \in \{0, 1\} \quad h \in H, l \in L \]  

\[ z_v, y^p_v, x_{uv} \in \{0, 1\} \quad v, u \in V, p \in P \]  

\[ f^p_v, f_{uv} \geq 0 \quad v, u \in V, p \in P \]

In this formulation, constraints (2) impose that each cell containing HHs is served by at most one WT, unless it remains isolated. Constraints (3) ensure that a cell containing HHs cannot be assigned to a potential WT location cell if no WT is located in it. Constraints (4) guarantee that a WT can be opened in \( l \) only if it covers at least \( \mu \) HHs. Constraints (5)–(8) link variables \( x \) and \( y \) to variables \( w \) and \( z \), ensuring that WTs or branches in the WSS can be located in a given cell if and only if that cell receives water, either from another node or directly from the water sources. Constraints (9) ensure that \( l \) is not both a WT and a branch cell, while constraints (10) ensure that at most one arc between \((v, u)\) and \((u, v)\) belongs to any feasible solution. Constraints (11) guarantee that an arborescence forest is built. Constraints (12) are a readapted version of the classical Dantzig, Fulkerson and Johnson \cite{2} subtour elimination constraints. Constraints (13)–(15) link variables \( y \) with variables \( f \), ensuring that the flow sent from a water source \( p \) does not exceed its capacity. Finally, constraints (16) and constraints (17) are flow conservation constraints. The objective (1) requires the minimization of the WTs and of the cost to connect them to the pipeline network, while ensuring that no HHs remain isolated or are assigned to locations that are within the widened distances, unless no better solution exists.

### 3 Matheuristic approach and Conclusions

Due to the large size of the problem, the CWNRP cannot be solved exactly, therefore we develop a matheuristic approach to address it. This approach is based on the decomposition of the mathematical model into two problems, the WTL problem and the WDN problem. The WTLP is an uncapacitated facility location problem and it aims to identify the optimal locations for WTs. The latter uses constraints (2)–(4) and constraints (18), and its objective requires the minimization of the first, third and forth term of (1). Let \((w^*_l, s^*_{hl}, t^*_h)\) represent its optimal solution. The WDNP consists of identifying the min-cost WSS to connect the WTs identified by solving the WTLP to the water sources. The WDNP uses constraints (5)–(17) and constraints (19)–(20), where variables \((w_l, s_{hl}, t_h)\) are replaced by the optimal values \((w^*_l, s^*_{hl}, t^*_h)\). The objective of the WDNP requires the minimization of the second term of (1). This is a work-in-progress project, and detailed computational results obtained by using real data will be provided at the conference.

### References


The Inventory Routing Problem with Demand Moves

Annelieke C. Baller
Department of Information, Logistics and Innovation,
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
Email: a.c.baller@vu.nl

Guy Desaulniers
Department of Mathematics and Industrial Engineering,
École Polytechnique and GERAD, Montréal, Canada

Said Dabia
Department of Information, Logistics and Innovation,
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Wout E.H. Dullaert
Department of Information, Logistics and Innovation,
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

1 Introduction

The Inventory Routing Problem (IRP) combines the optimization of inventory management and routing of the replenishments for a set of customers. In the case of ATMs, each ATM faces a certain demand from clients. In large cities ATMs are often located in close proximity which provides the opportunity to redirect clients from one ATM to another in case of a cash shortage, hence, to move demand between ATMs.

The possibility of redirecting clients can be incorporated in the optimization of the replenishments to reduce total costs. A cash management company in the Netherlands considers this a realistic option to reduce their ATM replenishment costs. Moreover, in the future it might be possible to provide clients with information upfront via a mobile application and give them a small discount if they withdraw cash from an ATM such that cash withdrawals are more efficient from the point of view of the cash management company.

We model the IRP with Demand Moves (IRPDM) in which demand of a customer can (partially) be satisfied by another customer. For each demand move, a service fee/cost
is incurred which is proportional to the size of the redirected demand and the distance between the two customers involved. We assume demand can only be moved if there is no remaining inventory. For each customer, there is a limited subset of other customers to which clients can be redirected, for example the closest customers.

1.1 Literature

The IRPDM is related to the IRP with Transshipment (IRPT) introduced by Coelho et al. (2012) [1]. In the IRPT, one can move goods between customers and also from the depot to customers, for example in order to redistribute merchandise between stores. It is assumed that these transshipments are performed by an outsourced carrier. In [1] a formulation for this problem is proposed and an ALNS solution method is used to solve it. Coelho and Laporte (2013) [2] use a Branch-and-Cut algorithm without problem specific valid inequalities to solve the same problem.

On the one hand, the IRPDM is a special case of the IRPT. First, in the IRPT it is possible to transship goods and store the goods at the destination customer to be used during multiple periods. In the IRPDM the goods are not transshipped, but the demand is moved. Therefore, a demand move in one direction, is equivalent to transshipment of goods in the other direction that is consumed immediately. Second, in the IRPDM demand moves to the depot are not possible while in the IRPT goods can be moved directly from the depot to a customer.

On the other hand, the IRPDM contains some features that generalize the IRPT. First, we handle the multiple vehicle case, while in [1] and [2] only the single vehicle case is considered. Second, we restrict for each customer the set of other customers to which demand can be moved. Both in [1] and [2] the set of customers from which goods can be transshipped is limited to a subset of the customers. This subset is determined for the entire instance, and from a customer in this set, goods can be moved to any other customer without restrictions. In the IRPDM a large distance between customers would make a demand move impractical. Therefore, we restrict demand moves for each customer to a small subset of other customers and this subset can be different for each customer. Although the restriction to a general subset of source customers is modeled in [1], the feature does not seem to be used in the computational tests.

2 Mathematical formulation

To model demand moves in the IRP we extend the IRP formulation as introduced by Desaulniers et al. (2016) [3]. In [3] a Branch-Price-and-Cut algorithm is proposed to solve the IRP. In the model, columns represent a combination of a route and a so-called Route Delivery Pattern (RDP) specifying the quantity delivered to each customer along
In the Master Problem the optimal combinations of routes and RDPs are selected to minimize total routing and inventory holding costs. In the Pricing Problem routes and associated RDPs are generated based on the reduced costs retrieved from the Master Problem.

In short, to introduce the demand moves in the described model we adjust the RDPs to include demand moves. Besides that, in the IRP it can be determined upfront in which periods of the planning horizon the initial inventory will be used to satisfy demand based on the first-in, first-out principle (FIFO). In the IRPDM the initial inventory needs to be handled differently, since initial inventory can now be used to satisfy moved demand. Furthermore, in the IRPDM we add an additional cost term in the objective representing the costs of the demand moves. Finally, we add constraints that enforce that demand can only be moved if there is no stock left at a customer. This coincides with the FIFO principle.

More specifically, we introduce for each customer $i \in N$ a set of customers $\mathcal{N}_i \subseteq N \setminus i$ for which demand can be satisfied, i.e., demand can be moved from $j \in \mathcal{N}_i$ to $i$. This set can contain for example the customer(s) closest to $i$. Because we restrict the set of customers for which demand can be satisfied, it is possible that in one period a demand move occurs from customer $i$ to customer $j$ and also from customer $j$ to customer $k$ if a demand move from $i$ to $k$ is not possible. We prevent this kind of moves, since they are not desirable from a practical point of view.

In [3], for a visit to customer $i$ in period $p$, an RDP $w$ specifies the quantity $q^s_{wi}$ delivered to $i$ to satisfy demand in periods $s = p, p+1$ etc. To include demand moves, an RDP is extended to specify both the quantities $q^s_{wi}$ delivered to $i$ to satisfy demand of $i$ and the quantities $q^s_{wij}$ delivered to customer $i$ dedicated to satisfy demand of customers $j \in \mathcal{N}$ in periods $s$. To use the initial inventory at a customer to satisfy demand of another customer, we introduce decision variables $i_{ij}^p$ indicating the number of units of initial inventory at $i$ used to satisfy the demand of customer $j$ in a period $p$.

The Pricing Problem is defined per period of the planning horizon. Given a period, the Pricing Problem is the combination of an Elementary Shortest Path Problem with Resource Constraints (ESPPRC) and a linear relaxation of a knapsack problem. The knapsack problem has a more complicated cost structure than the problem in [3] because of the demand moves and the related costs and dual variables. The Pricing Problem is solved using a heuristic and exact labeling algorithm. When extending a label in the Pricing Problem from customer $i$ to customer $j$ for period $p$, multiple labels are created each representing different delivery quantities for $p$ and the periods following $p$.

To strengthen the formulation, we add several valid inequalities. Note that, because of the introduction of demand moves, it is now possible to satisfy the demand of a customer with less replenishments than in the IRP. This influences for example the valid inequalities...
that put a lower bound on the minimum number of visits needed per customer formulated in [3]. Existing valid inequalities from [3] are adjusted to handle the demand moves, including the capacity inequalities which were newly introduced for the IRP in [3]. For example, to account for demand moves in the valid inequality on the minimum number of visits per customer, a move of demand to another customer can also be seen as a visit, hence contributing to the equation. Besides the valid inequalities used in [3], we consider the ‘multiperiod capacitated subtour inequalities’ proposed for the IRP by [4] and used by [5]. These inequalities were introduced for a Branch-and-Cut method applied to an arc flow formulation, and we test whether these are interesting for a path flow formulation for both the IRP and the IRPDM.

3 Computational tests

At the time of writing we are implementing the extension of the IRP to the IRPDM and hence there are no results to report yet. We are confident that we will have full computational results at the time of the conference. To analyze the influence of allowing demand moves, we will perform thorough computational tests on both existing and newly generated instances. We consider several measures to select the sets \( N \). Moreover, we test on restricting the demand moves, for example to limit the demand moves to a percentage of the total demand.

References


Fuel delivery in large networks

Paweł Hanczar
Wrocław University of Economics, Wrocław, Poland
Email: pawel.hanczar@ue.wroc.pl

Maciek Nowak  Jacek Kaleta
Loyola University, Chicago, USA  Samat Poland Ltd., Krakow, Poland

Under a vendor-managed inventory (VMI) system, a buyer provides regular inventory-related information to a supplier and the supplier is responsible for maintaining an appropriate inventory level at the buyer. This concept has many different variants in business practice, including the petrochemical industry with fuel distribution within a gas station network. In this scenario, the distributor that owns the network of gas stations is the buyer, providing information on fuel levels at each station, and the carrier that delivers the fuel is the vendor, responsible for planning shipment amounts and delivery dates. Distribution companies expect the carrier not only to deliver specialized transportation services, but also to develop end-to-end delivery plans that meet certain metrics. Carriers accept this approach as taking over additional responsibilities will result in better utilization of their fleet and ultimately make it difficult for the distribution company to change transportation providers. VMI also gives carriers the option of limiting the number and variability of vehicles needed to execute service.

As the carrier has control of both routing of delivery vehicles and inventory levels at distribution points, this takes the form of the well studied inventory routing problem (IRP) [1, 3]. The IRP has been applied in the context of a fuel distribution network in numerous instances [2, 7, 8, 9, 10]. However, the largest networks in previous research consider no more than 200 stations. Many real networks are an order of magnitude greater. In this research, we take advantage of a unique characteristic of the fuel distribution problem to solve several very large scale, multi-commodity IRPs, with almost 2000 gas stations, up to 33 refineries (depots), four commodities and a 3-7 day time horizon. We apply a variable neighborhood search heuristic to this problem, showing the benefits associated with a pooled inventory system and a model in which a vehicle may service multiple customers on one route. We also show how solutions differ when the objective is to minimize the kilometers traveled relative to tons of fuel delivered, as compared to the more traditional cost of transportation and inventory.

In a traditional IRP, the objective is to minimize the sum of transportation and in-
ventory costs. However, given the limited information that a carrier may have about the cost of inventory for a customer, carriers are interested in alternate metrics to evaluate performance. The primary metric used by the carriers that are involved in this study is the total distance traveled relative to the amount of fuel delivered. This is generally designated as the kilometers traveled divided by the tons of fuel delivered (KM/tons) and the objective is to minimize this metric. Combining these two values allows the carrier to evaluate the level of service they are delivering (maximizing the fuel shipped) and the efficiency of that service (minimizing the distance traveled). To the best of our knowledge, this metric has not been used in literature.

With this objective in mind, we determine the amount of each fuel type to deliver to each station via which route on each day of the problem horizon. The specialized tankers used to distribute fuel are most commonly 20-ton semi-trailers with different types of fuel in distinct chambers with capacities from 1,500 to 9,000 liters. For this problem, we allowed for the delivery of four fuel types, or commodities. The primary method of planning fuel supply is to deliver all fuel types with one trailer to one gas station, using a full truckload (FTL). In order to maximize the use of a chamber’s capacity, the sum of the different fuel types supplied to one station should be equal to the total capacity of the vehicle. The multiple compartments allow for a second approach, routing a tanker to deliver fuel to more than one gas station, as in an LTL network [4]. This is less common as it may lead to an increase in the number of visits at a customer. However, gas station operators are willing to consider this routing if it leads to a decrease in cost or an improvement in service. The carriers in this study currently route using a simplified FTL operation, such that when inventory at a station drops to a level allowing sufficient capacity for a full tanker of fuel, a delivery is made from the nearest refinery that fully replenishes the station. As these carriers are very interested in exploring the benefits associated with an LTL system, we model that a carrier may deliver to multiple customers on a route.

The variable neighborhood search begins with a set of routes as constructed by the carrier, with FTL service for each station from the nearest refinery. As we combine customers onto common routes, we take advantage of the fact that a route may consist of at most three stations. With a limited number of chambers on the vehicle and large demand from each customer, a tanker is generally constrained to visit no more than three customers on a route. Cornillier et al. [6] introduced this concept with a two customer route. Given this constraint, we more easily construct a set of feasible routes by combining deliveries that occur on the same day or on consecutive days, assigned to the refinery that is closest to a station on the route. This set of feasible routes is used to construct a supply plan that satisfies demand as it occurs over the course of the problem horizon. We utilize several neighborhood structures for improvement, including swap and 2-opt, iteratively perturbing the solution.
The model is used to solve test instances based on data collected from several gas stations in Poland and abroad. Four real instances are examined as defined in Table 1, with each representing a network operating in Central Europe. Demand data is provided for a seven day period for each network.

Both the carriers and customers are interested in decreasing the inventory levels at the stations as they feel that the current levels are not efficient, determined based on the size of the tanks at the gas stations. As indicated in Coelho and Laporte [5], an inventory level set below maximum capacity can lead to a cost savings of 1-2%, with inventories set at 80-90% of capacity. To evaluate the possibility of adjusting the order-up-to level, the tank capacity was tested with several values below the original capacity. Further, the carriers and their customers have considered a virtual pooled inventory scheme in which each station has a minimum inventory level, but the maximum inventory level is aggregated for the entire network. Pooling inventory in this fashion allows for the aggregation of demand and a greater ability to react to variability. Our model is used to evaluate a network in which the pooled system inventory must remain above a certain level, while each gas station has an independent, lower safety stock level.

Initial results indicate that the average stock within the network increases with an objective of minimizing the metric KM/ton when compared to a traditional metric of minimizing transportation and inventory costs. While this is not surprising, it indicates that carriers that focus on this objective myopically can lead to increased holding costs for their customers. When combined with the limited order-up-to levels, average stock levels decreased while efficiency improved. These limits on the amount of inventory that the customer could carry led to a greater decrease in distance traveled than fuel transported. Also, virtually pooling the inventory led to considerable improvements with very little degradation in service.

As we continue to improve the heuristic used for this analysis, the carrier has begun to implement our recommendations. A comparison of past operations and those in which multiple customers may be served on one route will be presented.

References


A *Branch-and-Price* approach for an inventory and routing problem to address the replenishment of a network of automated teller machines

Cristián E. Cortés  
Civil Engineering Department,  
Universidad de Chile, Santiago, Chile  
Email: ccortes@ing.uchile.cl  

Daniel Herl  
Industrial Engineering Department,  
Universidad de Chile, Santiago, Chile  

Pablo A. Rey  
Department of Industry,  
Universidad Tecnológica Metropolitana, Santiago, Chile  

Alejandro Cataldo  
Industrial Engineering and Systems Department,  
Pontificia Universidad Católica de Chile, Santiago, Chile  

Leandro Coelho  
Department of Operations and Decision Systems,  
Université Laval, Québec, Canada  

1 Introduction

This study was motivated by the distribution of money in a network of automated teller machines (ATMs) located in Santiago, Chile and the routing of vehicles in charge of the replenishment of the different ATMs. The resulting problem turns out to be an inventory and routing problem (IRP). In particular, given a set of customers and a finite time horizon, the objective of the IRP consists of determining the time to visit each client and the amount of money delivered at each visit, with the goal of minimizing the overall cost [1]. We propose the use of a *branch-and-price* framework for the multiperiod IRP problem. One important feature of our approach is that we allow *out-of-stocks* in the inventory of the teller machines, which are penalized. We also allow in the model the possibility of starting any given route with some delay. This last feature generates an additional difficulty due to the existence of *column-dependent-rows*, which means that we dynamically generate columns and rows simultaneously, which creates an extra difficulty for the pricing problem of our approach.

In the literature we found some interesting references related to the replenishment of
ATMs. One first approach presented in [3] proposes an integer programming model for the IRP with pickups and deliveries, which does not consider the possibility of becoming out-of-stocks, although this case incorporates what is denoted recirculation ATMs (which provide customers the capability of both withdrawing and depositing cash) that were gradually replacing the way to operate of ATMs in the Netherlands. A second approach is found in [2] where an arc-based model is presented for a problem with out-of-stocks but with the simplification that the exact moment of each visit, within each period, is not incorporated explicitly in the model.

2 Our Approach

A Branch-and-Price framework consists in performing both column generation and branch-and-bound with the goal of solving large integer problems. The column generation identifies the columns that will most likely be used in the optimal solution, without having to enumerate all the possible columns, which along with the branch-and-bound approach ensures an integer solution to the problem. As usual in a column generation scheme, we identify first a restricted master problem (RMP), which selects, for each period of time, the routes to be used subject to a set of constraints, and then, in the second stage, a pricing model will generate new routes to be used by the RMP. In this scheme, we wrote a pricing problem for each period of time, where we look for routes with negative reduced cost to be added to our master problem. If no negative reduced cost columns are found, we have reached the optimal solution of the current relaxed RMP. Both the restricted master and the pricing models are formulated as mixed integer problems.

The RMP is based on an IRP where every ATM is a client that must be served for at most one route. For its construction, we define a set of static variables related to inventory and out-of-stock levels, meanwhile the variables related to both route selection and starting time of those routes are generated dynamically. Each route establishes the set of ATMs that will be visited and the instant of the visit. When the truck arrives to the ATM, the cassette with money currently at the ATM is replaced by another one full of money. In our problem there are multiple cassettes with different size that can be used anytime in every ATM. The main goal of this model is to minimize the operational cost and penalties associated with the ATMs that incur on out-of-stocks. In addition it is assumed that the rate of consumption of every ATM’s demand is continuous and constant.

Let $I$ be the set of all ATMs ($i = 1, \ldots, I$) in the network, $T$ the set of periods in the problem ($t = 1, \ldots, T$) and $R_{i,t}$ the set of all the routes that visit the $i \in I$ ATM in the $t \in T$ period (thus $R = \cup_{i,t} R_{i,t}$ is the set of all the routes in the RMP). The variable $s_{i,t}$ reflects the inventory for the ATM $i$ at the beginning of period $t$, the variables $y_{i,t}$ and $o_{i,t}$ are a binary and continuous variable that reflects if the ATM $i$ incurs in out-of-stock in the period $t$ and the amount of demand not satisfied in the same ATM and period,
respectively. On the other hand, we denote $x_{r,t}$ and $f_{r,t}$ indicating if the route $r$ is used in the period $t$ and its starting time, respectively. The former variable is binary while the latter is continuous. Thus, the RMP is as follows:

$$\text{minimize} \quad \sum_{r \in R} \sum_{t \in T} c_r x_{r,t} + \sum_{i \in I} \sum_{t \in T} \left( g_i o_{it} + h_i s_{it} + k_i y_{it} \right)$$

subject to

$$-s_{i,t} + s_{i,t+1} - M_i \sum_{r \in R_{i,t}} x_{r,t} - \lambda_{i,t} y_{i,t} \leq -\lambda_{i,t} \quad \forall i \in I, t \in T$$

$$s_{i,t} - s_{i,t+1} - M_i \sum_{r \in R_{i,t}} x_{r,t} \leq \lambda_{i,t} \quad \forall i \in I, t \in T$$

$$-s_{i,t+1} + \sum_{r \in R_{i,t}} (Q_{i,r,t} - \overline{X}_{i,r,t}) x_{r,t} + \lambda_{i,t} \sum_{r \in R_{i,t}} f_{r,t} \leq 0 \quad \forall i \in I, t \in T$$

$$s_{i,t+1} + \sum_{r \in R_{i,t}} (-Q_{i,r,t} + \overline{X}_{i,r,t} + M_i) x_{r,t} - \lambda_{i,t} \sum_{r \in R_{i,t}} f_{r,t} \leq M_i \quad \forall i \in I, t \in T$$

$$-s_{i,t} - \sum_{r \in R_{i,t}} \overline{X}_{i,r,t} x_{r,t} + \lambda_{i,t} \sum_{r \in R_{i,t}} f_{r,t} - o_{i,t} \leq -\lambda_{i,t} \quad \forall i \in I, t \in T$$

$$-\lambda_{i,t} y_{i,t} + o_{i,t} \leq 0 \quad \forall i \in I, t \in T$$

$$\sum_{r \in R_{i,t}} x_{r,t} \leq 1 \quad \forall i \in I, t \in T$$

$$-(1 - d_{r,t}) x_{r,t} + f_{r,t} \leq 0 \quad \forall r \in R, t \in T$$

$$-s_{i,1} = -\theta_i \quad \forall i \in I$$

Constraints (1) to (6) describe the dynamic behind both the inventory management and the variables associated with out-of-stocks, where the parameter $\lambda_{i,t}$ is the demand of the ATM $i$ during period $t$. $M_i$ is the maximum amount of money that can be delivered to ATM $i$, $Q_{i,r,t}$ is the amount delivered by the route $r$ to the ATM $i$ during period $t$ and $\overline{X}_{i,r,t}$ is the remaining demand at the ATM $i$ in period $t$ after the visit of the truck corresponding to route $r$, assuming that the route starts at the beginning of the period. Constraint (7) sets an upper bound on the amount of demand not met, if there is an out-of-stock at the ATM $i$ during period $t$; constraint (8) states that at most one route can visit an ATM during each period, while constraint (9) limits the maximum time that the beginning of a route can be delayed, where $d_{r,t}$ indicates the duration of the route $r$ in period $t$ as a fraction of the period. Finally, constraint (10) sets the inventory level at the beginning of the first period.

As mentioned before, delaying the start time of routes in our RMP generates an extra difficulty to the formulation and solution of the model as we now must deal with column-dependent-rows. In our model, every new route added to the RMP introduces: a variable $x_{r,t}$, a variable $f_{r,t}$ and a constraint of type (9) (with dual variable $\eta_{r,t}$). This is not compatible with the usual column-generation scheme as described in [4] and the authors
outline a procedure to properly address this extra difficulty. The procedure indicates that from the solution of the relaxed RMP, a feasible solution must be obtained; besides, from the dual values of the relaxed RMP, a dual solution has to be constructed (not necessarily feasible; if it is feasible, then it is optimal). Note that in this dual solution at least one constraint is violated, and therefore, it can be introduced in the RMP as a new variable. In our application, the dual problem has two sets of constraints (associated with variables \(x_{r,t}\) and \(f_{r,t}\)) in which, the dual variable associated with constraint (9) arises explicitly. From the constraints linked to the variable \(f_{r,t}\) we can obtain an expression for \(\eta_{r,t}\) that can be used to solve the pricing problem related to variable \(x_{r,t}\). Thus, the optimality condition to stop the search of new routes in our approach is:

\[
\sum_i \left[ Q_{irt} (\gamma_{it} - \phi_{it} - \zeta_{it}) - \lambda_{irt} (\gamma_{it} - \phi_{it} + \delta_{it}) - M_i (\alpha_{it} + \beta_{it} - \phi_{it}) + \mu_{it} \right] - (1 - d_{rt}) \eta_{rt} \leq c_{rt}, \tag{12}
\]

where \(\alpha_{i,t}, \beta_{i,t}, \gamma_{i,t}, \phi_{i,t}, \zeta_{i,t}, \delta_{i,t}, \nu_{i,t}, \mu_{i,t}, \eta_{r,t}\) and \(\pi_{i}\) are the dual variables of constraints (1) to (10), respectively and \(\eta_{r,t}\) can be expressed as \(\sum_i \lambda_{i,t} (\gamma_{it} + \delta_{it} - \phi_{it} - \zeta_{it})\).

### 3 Preliminary results

So far, we have implemented the B&P scheme for a synthetic and controlled instance of 3 ATMs, 2 periods and 2 types of cassettes. The solving time for our instance is 5.74 seconds, the B&B tree generates 59 nodes (29 internal and 30 leaves) and the problem was solved to optimality, with a 13.25% gap between the optimal integer solution and the root LP solution.

### Acknowledgements

The authors wish to acknowledge the support from the Complex Engineering Systems Institute (CONICYT - PIA - FB0816) and CONICYT/FONDECYT/REGULAR/ N.° 1141313.

### References


Combining Optimization and Simulation for Designing Shortsea Feeder Networks

C. A. Benjamin Medbøen  Magnus B. Holm  Kjetil Fagerholt  Peter Schütz
Department of Industrial Economics and Technology Management
Norwegian University of Science and Technology, Trondheim, Norway
Email: peter.schuetz@iot.ntnu.no

1 Introduction

In Norway, the demand for cargo transportation (measured in tonne-kilometres) is expected to grow by 40% until 2030. This expected increase in transportation is met by a political ambition to shift more goods from the road to sea and railroad. However, road-based door-to-door transportation is often less expensive than alternative maritime transportation solutions, while at the same time offering high frequencies and reliability of service. In order to increase the share of waterborne transportation considerably, the competitiveness of the maritime transportation sector needs to improve for short distances. Some of the most important challenges that need to be addressed have been identified as high costs related to port visits and small ports that cannot accommodate large ships.

In an attempt to address these challenges, a novel maritime transportation system has been proposed: Short Sea Pioneer (SSP). The SSP concept builds upon the conventional idea of having several feeder routes served by smaller daughter ships, which are connected to a main route served by bigger mother ships. The relatively small daughter ships can visit ports that are not accessible for large conventional container ships. This can save road transportation of goods from small to large ports. The unique and new aspect with the system is that mother and daughter ships can meet at sea to transship their cargo. Thus, expensive port and storage costs can be avoided, but synchronized schedules for mother and daughter vessels are required.

Synchronized schedules can be sensitive to delays in the system and therefore need to be robust with respect to potential sources of delays. We combine a deterministic liner shipping network design problem with a simulation model that estimates sailing times based on historical weather data, in order to determine the optimal fleet of mother and daughter ships as well as their routes.
2 Problem Description

The problem can be formulated as a Liner Shipping Network Design Problem under uncertain weather conditions. We first provide a short description of the deterministic version of the network design problem, before discussing the effect of weather uncertainty.

2.1 The Network Design Problem

The objective of the problem is to determine a set of routes for mother and daughter ships, such that the sum of charter costs and operational costs is minimized. We consider the case of a Norwegian container shipping company operating between the European continent and Western Norway, as illustrated in Figure 1. The vessels have to visit each port once a week, where continental main ports in Europe can only be visited by the mother ship, coastal main ports, i. e. larger ports, in Norway can be visited by either mother ship or daughter ship (but not both) and local main ports, i. e. smaller ports, can only be visited by the smaller daughter vessels. Transshipment between mother ship and daughter ship takes place in an ocean hub. Ocean hubs are suitable candidate locations at sea for transshipment. We distinguish between north-going and south-going ocean hubs, where a north-going ocean hub is visited by a north-going mother ship and a south-going ocean hub is visited by a south-going mother ship.

![Figure 1: An illustration of the SSP logistics system.](image)

A route sailed by a daughter ship, i. e. the daughter route, can only be sailed by one daughter vessel and must be completed within one week. The main route connecting the
Norwegian west coast with ports on the European continent must be served with either one or two mother ships.

We assume that there is a known and constant weekly cargo demand between the continental main port and each of the coastal ports. All cargo has to be transported. Shipping of cargoes between coastal ports is not considered since the vast majority of the cargoes transported by the case company are either going to or coming from the continental main port. A transshipment can only occur in an ocean hub between a mother and daughter ship. Each cargo can only be transshipped once which means that the ship receiving the cargo will also deliver it to its final destination. We assume that the mother vessel is always chosen large enough to carry all cargo, while there are capacity restrictions for the daughter vessels.

A daughter ship can meet a mother ship once or twice every week for transshipment in the ocean hubs. If a daughter ship meets a mother ship twice, one of the visits must be with a north-going mother ship and the other visit must be with a south-going mother ship. The meeting location must be in the same ocean hub. Furthermore, a north-going mother ship will only deliver cargoes to a daughter ship, whereas a south-going mother ship will only pick up cargoes from a daughter ship.

### 2.2 Effects of Weather Uncertainty

When weather uncertainty is introduced, the time usage for a ship during a round trip can deviate from the original plan, e. g. due to reduced speed in high waves. For a synchronized transshipment to take place in an ocean hub, both the mother ship and the daughter ship have to be present at the same time. If one of the vessels is delayed, we observe a synchronization violation and – as a consequence – both ships will be delayed. Thus, a single delay can propagate throughout the system and lead to delays for the other ships.

If a ship is delayed by too much, it might be unable to complete its route within the maximum allowed duration. These duration violations prevent weekly port visit frequency, and carry delays over into the next week. The SSP logistics system therefore needs robust schedules, e. g. using buffer time windows to account for potential delays, and/or operational flexibility, e. g. the ability to speed up above the design speed of the vessel, to ensure the reliability and regularity of the transportation service.

### 3 Solution Approach

The problem of finding the optimal set of robust routes for mother and daughter ships under uncertain weather conditions is solved by combining the deterministic optimization model with a simulation model. This framework is illustrated in Figure 2. The iterative
process between the master and simulation model is referred to as the *solution triggered feedback approach*.

![Diagram](image)

**Figure 2:** The optimization-simulation framework with solution triggered feedback.

Based on the input data, we generate the set of feasible mother and daughter routes using a dynamic label setting algorithm. The optimization model, i.e. the master problem, determines the optimal set of routes. The solution proposed by the master problem is then evaluated using the simulation model. Using historic observations of wave height along the selected routes, the sailing speed of the vessels is estimated and their arrival time at ports and ocean hubs is calculated. Any delay in the schedule of either mother or daughter ship is logged. If a delay causes the ship to exceed its maximum roundtrip duration, a penalty cost is added to the cost of the route to make the route less attractive for the optimization model.

The updated cost information of the simulated routes is then fed back to the optimization problem in order to determine a new optimal solution. If the new solution contains routes that have not yet been simulated, a new simulation of the selected routes is triggered. This iterative procedure continues until no new (i.e. non-simulated) routes are selected by the master problem. The feedback approach between the master problem and the simulation model guarantees that no better solutions can be found based on the trade-off between operational costs and penalty costs.

## 4 Computational Results

We present and discuss results for the Short Sea Pioneer concept based on real world data from the Norwegian west coast. The results show that our solution approach is able to generate robust schedules for the proposed maritime transportation system.
Node-Based Lagrangian Relaxations for Multicommodity Capacitated Fixed-charge Network Design

Mohammad Rahim Akhavan Kazemzadeh
CIRRELT and Département d’informatique et de recherche opérationnelle,
Université de Montréal, Canada

Tolga Bektaş
School of Management and Centre for Operational Research,
University of Southampton, United Kingdom

Teodor Gabriel Crainic
CIRRELT and Département de management et technologie,
Université du Québec à Montréal, Canada

Antonio Frangioni
Dipartimento di informatica,
Università di Pisa, Italy

Bernard Gendron
CIRRELT and Département d’informatique et de recherche opérationnelle,
Université de Montréal, Canada
Email: Bernard.Gendron@cirrelt.ca

Enrico Gorgone
Dipartimento di matematica ed informatica,
Università di Cagliari, Italy

The multicommodity capacitated fixed-charge network design (MCFND) problem is defined on a directed graph $G = (N, A)$, where $N$ is the set of nodes and $A$ is the set of arcs. For each node $i \in N$, we define the sets of forward and backward neighbours, $N_i^+$ and $N_i^-$, respectively. Each commodity $k \in K$ corresponds to an origin-destination pair such that $d^k$ units of flow must travel between the origin $O(k)$ and the destination $D(k)$. The objective function to be minimized includes a cost $c_{ij}^k \geq 0$ for routing one unit of commodity $k \in K$ through arc $(i, j) \in A$ and a fixed cost $f_{ij} \geq 0$ for using arc $(i, j) \in A$, thus providing a capacity $u_{ij} > 0$ on the arc. A classical model for the MCFND introduces
two sets of variables: $x_{ij}^k$ is the flow of commodity $k \in K$ on arc $(i, j) \in A$, while $y_{ij}$ is 1, if arc $(i, j) \in A$ is used, and 0, otherwise. The model is written as follows:

$$\min \sum_{(i,j) \in A} \sum_{k \in K} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in A} f_{ij} y_{ij} \quad (1)$$

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = b_i^k, \forall i \in N, \forall k \in K, \quad (2)$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}, \forall (i,j) \in A, \quad (3)$$

$$x_{ij}^k \leq s_{ij}^k y_{ij}, \forall (i,j) \in A, \forall k \in K, \quad (4)$$

$$x_{ij}^k \geq 0, \forall (i,j) \in A, \forall k \in K, \quad (5)$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in A. \quad (6)$$

The objective function (1) seeks to minimize the total routing and design costs. Constraints (2) are the usual flow conservation equations ensuring that the demands are routed from the origins to the destinations, where:

$$b_i^k = \begin{cases} +d^k, & \text{if } i = O(k), \\ -d^k, & \text{if } i = D(k), \\ 0, & \text{otherwise,} \end{cases} \forall i \in N, \forall k \in K.$$  

The capacity constraints (3) ensure that the sum of the flows on each arc $(i, j) \in A$ does not exceed its capacity $u_{ij}$. They also play the role of linking constraints because they ensure that no flow is allowed on an arc $(i, j) \in A$, unless that arc is open and its fixed cost is paid. Given that $s_{ij}^k = \min\{d^k, u_{ij}\}$, the strong linking constraints (4) are redundant, but adding them significantly improves the linear programming (LP) lower bound $Z^{LP}$.

The so-called knapsack and shortest path relaxations are obtained by relaxing in a Lagrangian way constraints (2) and (3)-(4), respectively [1]. The Lagrangian subproblem for the knapsack relaxation decomposes by arc, giving rise to a continuous knapsack problem for each arc $(i, j) \in A$. The Lagrangian subproblem for the shortest path relaxation decomposes by commodity, giving rise to a shortest path problem for each commodity $k \in K$.

The model has to be slightly reformulated to derive new node-based relaxations. We use the following notation for each node $i \in N$:

$$K_i^O = \{k \in K | i = O(k)\}, \text{ the commodities for which } i \text{ is the origin;}$$

$$K_i^D = \{k \in K | i = D(k)\}, \text{ the commodities for which } i \text{ is the destination;}$$

$$K_i^T = \{k \in K | i \neq O(k), D(k)\}, \text{ the commodities for which } i \text{ is a transshipment node.}$$
The reformulation exploits the following basic properties: 1) for each $k \in K$, it is well-known that the flow conservation equation at $i = D(k)$ is redundant; 2) because costs are nonnegative, for each arc $(i, j) \in A$, $x^k_{ij} = 0$ if $k \in K^O_j$ or $k \in K^D_i$. We then rewrite the flow conservation equations (2) as follows:

$$\sum_{j \in N^+_i} x^k_{ij} - \sum_{j \in N^-_i} x^k_{ji} = 0, \forall i \in N, \forall k \in K^T_i, \quad (7)$$

$$\sum_{j \in N^+_i} x^k_{ij} = d^k, \forall i \in N, \forall k \in K^O_i, \quad (8)$$

$$x^k_{ij} = 0, \forall (i, j) \in A, \forall k \in K^O_j \cup K^D_i. \quad (9)$$

We relax constraints (7) in a Lagrangian way by introducing the Lagrange multiplier $\pi^k_i$, $\forall i \in N, \forall k \in K^T_i$, for each of these constraints. The following valid inequalities are also added to improve the relaxation:

$$\sum_{j \in N^+_i} x^k_{ij} \leq g^k_i, \forall i \in N, \forall k \in K^T_i, \quad (10)$$

where $g^k_i = \min\{d^k, \sum_{j \in N^-_i} u_{ji}\}, \forall i \in N, \forall k \in K^T_i$.

The resulting Lagrangian subproblem decomposes by node. The subproblem for each node $i \in N$ is then:

$$(P^i_L) \quad Z_i(\pi) = \min \sum_{j \in N^+_i} \left( \sum_{k \in K} c^k_{ij}(\pi)x^k_{ij} + f_{ij}y_{ij} \right)$$

$$\sum_{j \in N^+_i} x^k_{ij} = d^k, \forall k \in K^O_i, \quad (12)$$

$$\sum_{j \in N^+_i} x^k_{ij} \leq g^k_i, \forall k \in K^T_i, \quad (13)$$

$$x^k_{ij} = 0, \forall j \in N^+_i, \forall k \in K^D_i \cup K^O_j, \quad (14)$$

$$\sum_{k \in K} x^k_{ij} \leq u_{ij}y_{ij}, \forall j \in N^+_i, \quad (15)$$

$$x^k_{ij} \leq s^k_{ij}y_{ij}, \forall j \in N^+_i, \forall k \in K, \quad (16)$$

$$x^k_{ij} \geq 0, \forall j \in N^+_i, \forall k \in K, \quad (17)$$

$$y_{ij} \in \{0, 1\}, \forall j \in N^+_i, \quad (18)$$

where

$$c^k_{ij}(\pi) = \begin{cases} 
  c^k_{ij} + \pi^k_i - \pi^k_j, & \text{if } k \in K^T_i \cap K^T_j, \\
  c^k_{ij} + \pi^k_i, & \text{if } k \in K^T_i \setminus K^T_j, \\
  c^k_{ij} - \pi^k_j, & \text{if } k \in K^T_j \setminus K^T_i, \\
  c^k_{ij}, & \text{if } k \in K^O_j \cap K^D_i, \quad \forall j \in N^+_i, \forall k \in K.
\end{cases}$$
This new node-based location relaxation gives rise to $|N|$ subproblems, one for each node $i \in N$, which is a capacitated facility location problem (CFLP) [2] where $K_i^O \cup K_i^T$ and $N_i^+$ are the sets of customers and facilities, respectively. A lower bound on the optimal value of the MCFND is then obtained as follows: $Z(\pi) = \sum_{i \in N} Z_i(\pi)$. The best lower bound is derived by solving the Lagrangian dual: $Z^{LD} = \max_{\pi} Z(\pi)$. Since the CFLP does not have the integrality property [3], we thus obtain a lower bound that improves upon the LP relaxation lower bound: $Z^{LD} \geq Z^{LP}$ and there are problem instances for which this inequality is strict.

We will present two other node-based relaxations based on Lagrangian decomposition, which further improve upon the lower bound $Z^{LD}$ at the expense of having to solve more difficult subproblems. We will show how to solve the Lagrangian duals associated with these relaxations by a bundle method and how to derive Lagrangian heuristics that solve series of multicommodity minimum cost network flow problems within a slope scaling procedure [4], as well as restricted MCFNDs. Computational results on a set of benchmark instances will be presented, allowing us to compare lower and upper bounds computed by the new node-based relaxations with those of the relaxation and heuristic methods from the literature (see [5, 6] and the references therein).

References


A Graph Reduction Heuristic For
Supply Chain Transportation Plan Optimization

Simon Belieres
LAAS-CNRS
Université de Toulouse, Toulouse, France
Email: sbeliere@laas.fr

Nicolas Jozefowiez
LCOMS
Université de Lorraine, Metz, France

Frédéric Semet
Centre de Recherche en Informatique, Signal et Automatique, Lille
Ecole Centrale de Lille, Lille, France

1 Introduction

This paper describes an algorithm for the computation of tri-echelon supply chains transportation plans. The problem is inspired from logistic issues appearing in large-scale restaurant chains. We consider products for which customers have a recurring request through the time horizon. Suppliers product and ship the commodities to customers by direct deliveries, or by way of intermediary hubs.

Freight flows in the supply chain are modelled by time-expanded graphs [3]. Unfortunately, realistic instances in terms of network and time horizon yield to graphs with a huge amount of nodes and arcs. Induced mixed integer linear programs are too large to be solved in a reasonable amount of time with an industrial solver.

We conceived a resolution heuristic able to produce good solutions despite instances increasing size. The method is inspired from Boland and Hewitt [1]. The authors focus on the Service Network Design Problem ($SNDP$). It is a comparable freight transportation problem, reviewed by Crainic [2]. However, the method cannot be applied directly as our problem differs on key elements. Our heuristic consists in generating a subgraph containing nodes and arcs required to transit the optimal flow. When generation is over, we solve a much smaller program induced from the subgraph, and get a not necessarily optimal transportation plan.
2 Problem description

The static network $D = (N, A)$ models our supply chain. The nodes $N$ represents the actors of the supply chain, partitioned in three sets. The set $U$ represents the suppliers, $V$ the hubs, and $W$ the restaurants. We consider a set of commodities $K$. Each supplier $u \in U$ can infinitely provide a subset of products $K^u \in K$. Each restaurant $w \in W$ has a demand $d_{wk}^t \geq 0$ of commodity $k$ at time $t$. Our objective is to determine the minimal cost transportation plan satisfying all the demands.

There is no temporal dimension in static networks, so we represent the supply chain by a time-expanded graph $D_T = (N_T, H_T \cup A_T)$ derived from $D$. The set of nodes $N_T$ is obtained duplicating the physical locations of $N$ through the time horizon. The set of arcs is decomposed into holding arcs $H_T$ - connecting two occurrences of the same physical location - and transportation arcs $A_T$. Note that holding arcs only exist for hubs. Each arc $((i, t), (j, t'))$ has a travel time $t' - t$, a per-unit-of-flow cost $c_{ij} \in \mathbb{R}^+$, a fixed cost $f_{ij} \in \mathbb{R}^+$, and a capacity $u_{ij} \in \mathbb{R}$.

For each demand $d_{wk}^t$ the destination is known, but the origin is unknown. Indeed, any supplier $u \in U$ such that $k \in K^u$ can fulfill the request. The solver must determine which one is the best option. Thus we have no commodities origin constraints, involving an unusual complexity level for transportation problems.

Given a time-expanded network $D_T$, we define $SCNDP(D_T)$ to be our Supply Chain Network Design Problem. Positive integer variables $y_{ij}^{tt'}$ represent the number of trucks used on arc $((i, t), (j, t'))$. Positive continuous variables $x_{ij}^{ktt'}$ model the flow of commodity $k$ on arc $((i, t), (j, t'))$. Note that $x_{ij}^{ktt'}$ is not defined if $k$ cannot transit on the given arc, i.e. if $i \in U$ and $k \notin K^i$. The following is a valid integer programm:

$$
\begin{align*}
\min & \quad z(D_T) = \sum_{A_T} f_{ij} y_{ij}^{tt'} + \sum_{k \in K} \sum_{A_T} c_{ij} x_{ij}^{ktt'} + \sum_{k \in K} \sum_{H_T} c_{ii} x_{ii}^{ktt'} \\
& \quad \sum_{A_T \cup H_T} x_{ij}^{ktt'} - \sum_{A_T \cup H_T} x_{ji}^{ktt''} \quad \forall ((i, t), (j, t')) \in A_T \, \forall (j, t) \in V_T \\
& \quad \geq d_{wk}^t \quad \forall k \in K, \ \forall (i, t) \in N_T \\
& \quad \sum_{k \in K} x_{ij}^{ktt'} \leq u_{ij} y_{ij}^{tt'} \quad \forall ((i, t), (j, t')) \in A_T \\
& \quad x_{ij}^{ktt'} \in \mathbb{R}^+ \quad \forall ((i, t), (j, t')) \in A_T \cup H_T, \ \forall k \in K \\
& \quad y_{ij}^{tt'} \in \mathbb{N}^+ \quad \forall ((i, t), (j, t')) \in A_T
\end{align*}
$$

We seek to minimize the total expenses, i.e. the fixed costs of allocating resources on transportation arcs and the linear costs of transportation and holding flows. Constraint (1) is an adapted Kirchhoff constraint, with no imposed origin on commodities. Constraint (2) ensures that enough trucks are allocated to ship the commodities. Constraints (3) and (4) are the variable domains.
3 Resolution method

3.1 Dynamic Discretization Discovery

Dynamic Discretization Discovery (DDD), method by Boland and Hewitt, is optimal for the Service Network Design Problem (SNPD). This problem is comparable to ours, with two major differences: commodities origins are defined and holding cost are null. The objective function seeks to minimize transportation costs only.

The DDD method consists in generating an initial subgraph $D_T$ under a certain set of properties. Those conditions respected, the subproblem $SNPD(D_T)$ provides a solution not necessarily feasible in the full graph. The subgraph contains arcs under-estimating real travel times, which allow commodities to arrive earlier than feasible in real-life. Therefore the subgraph offers unrealistic flow consolidations. However, the subproblem necessarily provides a lower bound to the initial SNPD.

The DDD method solves the subproblem $SNPD(D_T)$ and detects the use of early-arrival arcs in the solution. In that case, the subgraph is repaired and expanded while keeping its lower bound property. The process stops once a lower bound feasible to the full graph is found, i.e. the optimal solution.

3.2 A heuristic based on two reparation mechanisms

The DDD cannot be used unaltered to our problem, essentially because we consider strictly positive holding costs and our commodities have no predefined origins.

Considering strictly positive holding costs breaks the optimality property of the DDD, as we set an example in which the subproblem no longer provide a lower bound to the full problem. This is because having unrealistic flow consolidations is not sufficient to ensure finding a better solution than possible. Supressing the commodities origins prevent us from generating the initial subgraph similarly to the DDD.

We demonstrate the DDD is optimal to the SCNDP with free holding costs, if the subgraph contain any supplier occurrence $(u, t)$. However we must restrict the set of suppliers size, as including them all makes the method too slow. We initiate the subgraph with the suppliers shipping a non-null amount of commodities in the optimal solution of the SCNDP linear relaxation.

The subgraph generation heuristic is based on two reparation mechanisms. We build the initial subgraph, with early-arrival arcs and free holding costs only. The first reparation mechanism is the DDD, it fixes the transportation unfeasibilities. It detects the early-arrival shippings and refine the subgraph to prevent unfeasible consolidations. The second reparation mechanism is an holding costs injection, to fix the storage defects. It spots the free holding arcs used in the subgraph solution, and update the real costs. The method iterates those mechanisms until no transportation/storage anomaly appears. Note that
the programs solved in this phase are linear relaxations only, to accelerate the process.

Then, the subgraph remaining early-arrival arcs are suppressed and all the holding costs are updated. We then solve the SCNDP mixed-integer program induced from the subgraph, and obtain a feasible solution.

4 Results and Discussion

We tested the heuristic on 2 set of instances - representing a total of 90 instances - and compared it with the solution of the full SCNDP model by Gurobi. The first set is referred as easy instances and the industrial solver found optimal solutions, within the time limit of 2 hours. The second set is referred as difficult instances and the industrial solver only gave suboptimal solutions. The following ratio compares the results:

\[
\text{ratio} = \frac{H_{sol} - \text{MILP}_{sol}}{\text{MILP}_{sol}} \times 100
\]

A negative ratio states the heuristic outperformed Gurobi, otherwise Gurobi found a better solution.

<table>
<thead>
<tr>
<th>N</th>
<th>Ratio(%)</th>
<th>N</th>
<th>Ratio(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.70</td>
<td>80</td>
<td>-0.056</td>
</tr>
<tr>
<td>20</td>
<td>2.19</td>
<td>100</td>
<td>-18.75</td>
</tr>
<tr>
<td>30</td>
<td>1.13</td>
<td>120</td>
<td>-28.17</td>
</tr>
</tbody>
</table>

Table 1: Performance on easy/difficult instances

To summarize, we agglomerated the instances by the number of nodes of the supply chain. For easy instances, the average ratio is positive as Gurobi finds optimal solutions, what the heuristic does not. However, the ratio values are close to 0, indicating the heuristic solutions are of good quality. The difficult instances table reveals our heuristic is able to provide better solution than Gurobi in a given computational time. We observe that this gap becomes larger as the instance complexity increases.

References


An integrated approach for inbound train split and container loading in an intermodal railway terminal

Bruno P. Bruck
Department of Computer Science and Operational Research,
Université de Montréal, Canada

Jean-François Cordeau
Department of Logistics and Operations Management,
HEC Montréal, Canada

Emma Frejinger
Department of Computer Science and Operational Research,
Université de Montréal, Canada
Email: jean-francois.cordeau@hec.ca

1 Introduction

Intermodal freight transportation may be considered as one of the stepping stones of globalization, as it allows for efficient intercontinental door-to-door transportation of goods through a multimodal chain of land and sea transportation services that often involves several different carriers. In a classical example of an intermodal chain, loaded containers leave the initial shipper location by truck and are directed either to a port or to an intermodal terminal, from where a train will transport them to a port. A ship then moves the containers to another port, from where they are transported to the destination by one or a combination of several means of transportation ([1]).

These intermodal terminals are special transshipment nodes that are responsible for consolidating traffic and dispatching containers on trains destined to other nodes of the network, so that these containers can eventually reach their final destination. Although several studies in the literature concentrate on classification and shunting yards, they should not be confused with intermodal terminals, as only the latter allow temporary storage, loading and off-loading of containers. For a comprehensive survey on the literature on intermodal transportation we refer the reader to the surveys of [1] and of [2].

While the international market mainly follows the ISO standard and uses 20, 40 and 45-foot containers, in North America there are also 53 and 48-foot containers, which are used for domestic traffic. Another complication of this market is that trains are usually double stacked and there are many types of railcars with different characteristics, e.g., number of wells (platforms), well length (40, 45, 48 and 53 feet long) and weight loading
limit. This great variety of containers and railcar types has a significant impact on the load planning, which concerns the assignment of containers to railcar slots. Performing a proper matching in the load planning is very important to ensure the best usage of the available capacity of railcars and also the fuel efficiency of the train, given that double stacked containers influence aerodynamic aspects.

The block planning is a tactical problem and is another critical issue for the design of an efficient and profitable rail transportation system. A block is defined as a group of railcars, with possibly different origins and destinations, that are moved as a single unit between terminals. Because of this the railcars of the same block do not need to be handled individually at intermediate terminals, reducing handling costs.

In this paper we focus on an operational problem faced by a North American railway in the context of railway transportation of intermodal containers. On a daily basis, terminal operators have to take decisions concerning several different and interconnected activities, such as: how inbound trains are split into sequences of railcars, on which tracks those railcars are parked for loading and off-loading operations or even for temporary parking, and how to design proper load and block plans. Because we focus on operational decisions, we assume that the block plan is partially given as the expected total length of each block and the demand of certain types of railcars are known. However, the choice of the individual cars that compose each block is optimized based on the available resources. We propose an integrated approach that incorporates all these decisions to achieve better results. In a first step a MIP model is used to create a set of configurations that specify how each inbound train is split and the assignment of railcars to the blocks. Then, a second MIP formulation decides which configurations to use and where to park the railcars, making sure that it is possible to bring the railcars to their assigned track and to pull them out when they have to depart.

2 Problem definition

We are given a sequence of inbound trains, a sequence of railcars arriving on these trains, a set of outbound blocks to be created, a set of outbound containers, and a set of tracks. Each inbound train can be split into a number of segments. A segment is composed by a set of railcars that occupy consecutive positions in the inbound train. Therefore, a segment can be defined by the first and last positions occupied by the railcars forming this segment. For example, a train comprising 100 railcars could be split into three segments: one from positions 1 to 35, one from 36 to 85 and one from 86 to 100. We assume that one can perform an a priori enumeration of the potential segments that can be created from any inbound train, and that the exact composition (in terms of railcars) and length of the segment is known with certainty. Each segment is further divided into a set of
sections (or sub-segments) that will ultimately be assigned to different outbound blocks.
For example, a segment with 35 railcars could be divided into a first section with 20 cars and a second one with 15 cars. The two sections will remain together until the departure of the outbound trains to which these sections are assigned.

The set of tracks can be partitioned into a set of storage and working tracks. While both types can be used to move railcars through the terminal and for temporary parking, containers can only be loaded on and off-loaded from railcars parked on working tracks. To model the fact that several segments can be parked on a single track at the same time, we discretize tracks into a number of track slots representing segments of a given length. For example, a 1000m track could be divided into 20 segments of 50m, thus a segment with length of 500m assigned to slot 1 would occupy the first 10 slots of this track. The sets of storage and working tracks can be further partitioned into a subset of single-ended tracks (where railcars enter and leave from just one end) and double-ended tracks (where railcars can enter and leave from both ends).

The order of outbound blocks in each outbound train is defined by the marshalling, which is a fixed plan given by the network administrator and can only be modified in exceptional situations after approval from the network side. For certain trains, changes might be needed to achieve a better slot utilization or even to ensure that it is possible to generate a solution in cases where there is high traffic at the terminal and it would be impossible to generate a solution that fully respects the marshalling. In these cases the order of the blocks could be readjusted by another terminal downstream.

The problem then consists in deciding on (i) how each inbound train is split, i.e., which segments and sections are created; (ii) which sections of each segment are assigned to each outbound block and (iii) in which track and track position each segment is parked.

3 Solution method

Our solution method is divided in two steps. The first one is the generation of a set of configurations for each inbound train. A configuration specifies how a given inbound train is split into segments, which sections compose each segment and the assignment of sections to outbound blocks. Given the combinatorial nature of this step, in practice, it is usually impractical to generate and consider all possible configurations for each inbound train. The objective then, is to generate a set of configurations as small as possible, containing configurations that somehow maximize the chances of being able to generate good feasible solutions in the second step. To solve the first step, we propose a MIP model that not only considers the requirements of each outbound block for certain types of railcars, but also takes into account feasibility constraints related to the marshalling and how each segment is split into a sequence of sections. Instead of converging to a single optimal solution, the
The aim of this model is to find as many feasible configurations as possible within a certain period of time.

The second step then takes as input a list of configurations for each inbound train and decides which configurations to use and where to place each segment in the terminal. We propose a second MIP formulation to solve this step, considering additional feasibility constraints related to how each segment is parked on the tracks, and how to pull out sections from the tracks in order to build the outbound trains. The objective function minimizes the distance of railcars to the container stacks, the time that railcars remain at the terminal, and penalties associated with problems in marshalling of the outbound trains, requirements of other terminals and the complexity of the movements required to pull sections out of the tracks.

4 Computational results

To test the algorithms we created a set of realistic instances based on data provided by the railway. We then performed a series of computational experiments with our algorithm to calibrate the parameters. Preliminary results show that we are able to generate feasible solutions for most instances in a reasonable amount of time. These tests also show that the efficiency of the second step is very sensitive to the number and quality of the configurations used, which confirm the importance of being able to generate small sets of good quality configurations in the first step.

Acknowledgments

We gratefully acknowledge the close collaboration with personnel from the Canadian National Railway Company (CN). This research was funded by the CN Chair in Optimization of Railway Operations at Université de Montréal and a Collaborative Research and Development Grant from the Natural Sciences and Engineering Research Council of Canada (CRD-477938-14).

References


A Fairness-driven Scheduling Model for Airport Slot Allocation: is there a cost for fairness?

Konstantinos N. Androutsopoulos
Department of Management Science & Technology
Athens University of Economics and Business, Athens, Greece

Michael A. Madas
Department of Applied Informatics
University of Macedonia, Thessaloniki, Greece

Vassileios Sitokonstantinou
Department of Management Science & Technology
Athens University of Economics and Business, Athens, Greece

1 Introduction

Traffic forecasts illustrate bleak future prospects in terms of capacity pressures, congestion and level of service challenges for the air transport system. Recently, there has been an increasing thrust of research [1] exploring scheduling methods and models aiming to allocate efficiently scarce airport resources, expressed in slots, among competing airlines at the strategic planning phase. The strategic airport slot allocation problem aims to optimize the allocation of specific coordination time intervals to the respective airlines’ slot requests within a given scheduling season. Slots signify time intervals on specific dates and times for which an airline is permitted to schedule a landing or take-off. Slot requests are expressed in series (5 or more slots requested for the same time and weekday) and further classified on the basis of their scheduling priority in historical, new entrant slot requests and all remaining requests. The herewith presented problem has been modelled in existing literature with different underlying criteria and objectives. The first generation of slot scheduling models focused on scheduling efficiency by introducing the “schedule delay” or displacement metric (e.g., [2], [3], and [4]) at single or airport network level. Schedule delay stands for the absolute value of the difference between requested and allocated slot times [2]. More recent research efforts enriched efficiency objectives with acceptability and fairness considerations ([5], [6] and [7]). Lately, fairness-informed modelling considerations of the slot scheduling problem have been also addressed in [8], where the schedule displacement imposed on each airline is forced to be proportional to the respective number of requested slots.

Borrowing fundamental principles from congestion-based pricing (e.g., [9] and [10]), our paper capitalizes on the conjecture that the intensity of congestion and delays impacts is strongly affected not only by the demand volume but also by the temporal distribution and peaking characteristics of demand. Hence, we propose a single-airport scheduling model optimizing scheduling efficiency, while simultaneously incorporating fairness considerations. The latter aims to ensure that each airline absorbs its "fair share" of congestion in the form of
schedule displacement exerted onto other users. Furthermore, we assess the "cost of fairness", that is, the impacts of fairness-driven scheduling on efficiency and level of service (LOS).

2 Proposed Model

Assume that a set of airlines $A$ place slot requests $R$ that span throughout the planning horizon $D$ (the set of calendar days). We denote by $T = \{0,1,\ldots,n-1\}$ the set of time intervals (with length equal to five minutes) during the active airport operating hours. Each request $r \in R$ involves a requested time interval $\tau_r \in T$, the type of movement (arrival vs. departure) and a set of calendar days $D_r (\subseteq D)$ over which the request is applicable. Hence, each request involves a set of requested movements on calendar days in $D_r$. We define binary parameter $\delta_{rd}, r \in R, d \in D$ that takes value 1 if $d \in D_r$ and 0 otherwise. The requests corresponding to arrivals are denoted by $R_{arr} (\subseteq R)$, while those referring to departures are denoted by $R_{dep} (\subseteq R)$. The set of movements corresponding to a slot request $r$ is denoted by $M_r$. The slot requests that are placed by airline $\alpha \in A$ and the corresponding requested movements are denoted by $R_{\alpha}$ and $M_{\alpha}$ respectively. Any pair of slot requests $(r_1, r_2), r_1 \in R_{arr}, r_2 \in R_{dep}$ that are linked (i.e., the corresponding movements are operated by the same aircraft or they are connected) are stored in set $P$. The time intervals allocated to any pair of linked slot requests $(r_1, r_2)$ should exceed a minimum time difference $s_{r_1r_2}$ (referred to as turnaround time). Moreover, the allocation of time intervals to slot requests should not violate the capacity constraints of the airport. It is assumed that no more than $q_{ctd}^c$ movement of type $c$ ($c \in C = \{0,1,2\}$, where 0 corresponds to arrivals, 1 to departures and 2 to both arrivals and departures) can be served by the airport within any time period $T_{ct}^c = [t, t + t_c)$ of length $t_c$ (e.g., 1h or 12 time intervals). The displacement $(f_{rt})$ of each request $r \in R$ is defined as the difference between the requested time interval $\tau_r$ and the time interval $t$ allocated to it, i.e., $f_{rt} = |\tau_r - t|$. The objective of the proposed slot allocation problem is to determine a feasible schedule for the slot requests that minimizes the total displacement while simultaneously promoting a fair distribution of the total displacement among participating airlines. The proposed formulation is based on the general resource-constrained scheduling problem [11]. The decision variables of the proposed formulations are $x_{rt} \in \{0,1\}, r \in R, t \in T$ which take value 1 if slot request $r$ (and hence any associated requested movement in $M_r$) is allocated interval $t$, and 0 otherwise. The proposed formulation is given by (1)-(7):

$$
\min \sum_{d \in D} \sum_{r \in R} \sum_{t \in T} x_{rt} \cdot f_{rt} \cdot \delta_{rd}
$$

s.t.

(1)

(2)

(3)

(4)

(5)

(6)

(7)
where $\pi_a$ is the proportion of displacement that should be allocated to airline $a \in A$, $\pi_{\text{min}} = \min\{\pi_a; a \in A, \pi_a \neq 0\}$, and $\varepsilon \in (0,1)$ is a parameter that allows minor discrepancy from $\pi_a$. Parameter $\mu_{rc} \in (0,1)$ takes value 1 if slot request $r$ involves movements of type $c$, and 0 otherwise. Objective function (1) expresses the aggregate displacement, that is, the total absolute difference between the requested and allocated time for all days. Constraint (2) indicates that every slot request must be allocated to exactly one time interval. Constraint (3) implies that the number of movements of type $c$ allocated to any time period of $t_c$ length cannot exceed the corresponding capacity value. Constraint (4) assures the turnaround time difference between any pair of linked movements. It is worth noting that constraints (2)-(4) are the same as in [2]. Constraints (5) and (6) imply that the proportion of total displacement allocated to any airline $a$ for which $\pi_a \neq 0$ should not deviate from $\pi_a$ by $\pm \varepsilon \%$. Constraint (7) ensures that for any airline with $\pi_a = 0$ (i.e., it does not contribute to congestion), its maximum possible share in displacement should not exceed the corresponding minimum value that is allocated to any airline with $\pi_a \neq 0$. The values for $\pi_a$ are computed as follows:

- We solve the mathematical model (1)-(4). The optimum displacement value that emerges denoted by $D_{\text{opt}}$.
- For each $a \in A$, we solve the mathematical model (1)-(4) by including the constraint:

$$x_{rt} = 1, \ r \in R_a$$

which implies that any slot request placed by airline $a$ is allocated its requested time $\tau_r$.

The emerging value of the objective function is denoted by $D_a$.

- The difference $B_a = D_a - D_{\text{opt}}$ expresses the additional displacement due to fully satisfying the slot requests of airline $a$.

The value of $\pi_a$ for $a \in A$ is computed by (9):

$$\pi_a = \frac{B_a}{\sum_{a \in A} B_a}$$

### 3 Computational Analysis and Conclusions

A series of experiments are performed in order to assess the impact of applying the proposed fairness measure to: i) the total displacement and ii) the level of service offered to the airlines. In this research work, the level of service is expressed in terms of a set of metrics founded on the assumption that each airline has a maximum tolerance on the displacement it receives per request denoted by $\xi_r$, $r \in R_a$. Hence, it is assumed that allocating a time interval $t \notin [\tau_r - \xi_r, \tau_r + \xi_r]$ to slot request $r$ of airline $a$ would be considered by the airline as a violated service [7]. The latter implies that the allocated slot violates “unacceptably” the original slot request above certain acceptability thresholds (or tolerance levels), denoted by $\xi_r$. If a request is allocated with a time interval within $[\tau_r - \xi_r, \tau_r + \xi_r]$, then the corresponding service of slot request is referred to as legitimate service on the grounds that it ensures an acceptable level of service within the tolerance limits of the airline's requests. Moreover, if a slot request $r$ is allocated with time interval $t > \tau_r + \xi_r$ or $t < \tau_r - \xi_r$, then the part of displacement that spans after $\tau_r + \xi_r$ (i.e., $t - (\tau_r + \xi_r)$) or before $\tau_r - \xi_r$ (i.e., $(\tau_r - \xi_r) - t$) is called excess displacement. The metrics used to express level of service are the following:

- **Percentage of requests with violated service**: this is the percentage of slot requests for which the allocated time interval is not in $[\tau_r - \xi_r, \tau_r + \xi_r]$. 

222
- **Average Legitimate Displacement**: this is the ratio of the sum of the displacement of the slot requests receiving regular service over the total number of regularly served requests.
- **Average Excess Displacement**: this is the ratio of the sum of the excess displacement over the number of requests experiencing violated service.

We calculate parameters $\pi_a$ for 56 airlines placing slot requests in a medium-sized European airport for the period between March 29 and May 31, 2009. Then, we solve the proposed model for various capacity levels and using a decreasing sequence of $\varepsilon$. For each run, we calculate the total displacement and the level of service metrics mentioned above. Preliminary results indicate that imposing fairness constraints results in an increase of the total aggregate displacement by 12% (for capacity level of 16 arrivals and 16 departures per hour), with 87.5% of this displacement being excess displacement.

**Acknowledgments**

The present research was partially supported by the Research Center of the Athens University of Economics and Business (AUEB-RC) through the projects EP-2479-01 & EP-2638-01.

**References**


Capacity-Oriented Marshaling and Shunting Yards Location Problem

Nikola Obrenović, Virginie Lurkin, Michel Bierlaire
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne
Email: nikola.obrenovic@epfl.ch

Vincent J. Baeriswyl, Jasmin Bigdon
SBB Cargo AG
Olten, Switzerland

1 Introduction

A railway network is commonly used to transfer commodities between origin and destination train stations. In order to reduce transport cost, a commodity is usually transferred from the origin station to a hub, where commodities from several origin stations are sorted and grouped by the common destination hub. At the destination hub, grouped commodities are separated and sent to their final destinations. Consequently, the number and locations of the hubs effect the operational costs to a great extent.

The hubs in railway networks are called marshaling and shunting yards, while the operations of sorting and regrouping the freight is commonly known as the freight shunting. The main differences between marshaling and shunting yards are the used technology and the available capacity in the terms how many wagons can be shunted over some period of time. In this abstract, we focus on the latter.

The goal of our work is to determine the optimal number and locations of marshaling and shunting yards in a railway network in order to reduce freight transport and shunting costs. For this purpose, we have defined an optimization problem whose solution gives the desired capacities of the potential shunting facilities in the network. The desired capacities consequently determine which facilities should be marshaling and shunting yards. The problem will be solved and tested over the railway network operated by the Swiss freight transportation company SBB Cargo.

The defined problem represents a combination of a multicommodity flow problem (MFP, [1]) and a hub location problem (HLP, [2]). Our problem models the capacity
of the hubs, instead of the usual approach of modeling arc capacities, and its solution simultaneously determines the hub locations and hub types, in terms of the maximum allowed capacity, and solves the MFP. On the other hand, in the available literature, we have not found any other MFP or FLP that has all mentioned properties combined.

There are several column generation and heuristic approaches for solution of a MFP [1]. However, our problem is aimed to be solved on a large network with a high amount of transported commodities, which the existing approaches would require a significant computational time for. Therefore, we apply the passenger assignment algorithm [4], adapted to commodities. This algorithm has shown to be very fast for the passenger assignment problem, and has the potential to considerably reduce the computational time for the assignment of the commodities, too. The assignment algorithm represents a recent contribution of the Transport and Mobility Laboratory at École Polytechnique Fédérale de Lausanne, and to the best of our knowledge, such approach for solving a MFP cannot be found elsewhere in the literature.

2 Optimization Problem

We have defined our problem as a mixed integer programming problem whose solution determines the required capacities of the potential marshaling and shunting yards, with respect to the given transport demand. In order to model the possibility that any railway station can also be a shunting or marshaling yard, each station is represented with a pair of incoming and outgoing nodes.

Between each pair of nodes representing the same station, we model three inner arcs that represent a regular station, shunting yard or marshaling yard. At any station, only one of these inner arcs can be used. These arcs have zero costs and their capacities depend on the type of the station. The model contains also the set of arcs that represent the direct links between the different stations.

We also include the set of commodities that need to be transported. Each commodity is defined by the origin and destination stations, number of wagons and weight.

The objective function represents the sum of transport costs for each commodity assigned to each arc, which are dependent on the commodity weight and arc length, and costs of freight shunting, dependent on the number of wagons. In addition to standard HLP and MFP constraints, we have included in our model the constraints for:

• modeling that each node can be either a regular station, shunting yard or marshaling yard, and limiting the node capacity according to its type,

• forcing that in each node, all commodities are processed either via the regular, shunting or marshaling inner arc, and
• limiting the number of shunting and marshaling yards in the railway network.

The solution of the problem is the flow on each arc. The flow on each used inner arc determines the required capacity of each node, i.e. station.

3 Case Study

The presented model will be applied to the freight railway network operated by the SBB Cargo company. The network contains at this moment 2 marshaling yards for the traffic within Switzerland, 3 marshaling yards on the borders, for the cross border traffic, 65 shunting yards and approximately 350 regular stations. Over this network, SBB Cargo transports over 65,000 commodities yearly between over 17,000 different pairs of origin-destination stations. The freight total weight is over 18M tons and it is transported with over 950,000 wagons in total. The operation as such is costly and the network might contain redundant or badly located marshaling and shunting yards. The overall interest of SBB Cargo is to minimize the operating cost of their network by redesigning the existing network of the inner marshaling and shunting yards.

3.1 Solution Algorithm

Many real networks, such as the SBB Cargo network, are too large for the presented optimization problem to be solved with an exact method. Therefore, we are currently developing a heuristic algorithm that both determines better locations for the shunting and marshaling yards, and routes the freight after the yard locations are established.

For determining yard locations, we use the variable neighborhood search (VNS) algorithm with the following operators:

1. transforming one of the marshaling yards into a shunting yard, i.e. reducing its maximum allowed capacity, or one of the shunting yards into a regular station. For this operation, we select the yard with the smallest ratio between its flow, calculated in the previous algorithm iteration, and its maximum capacity.

2. transforming one of the shunting yards into a marshaling yard, i.e. increasing its maximum allowed capacity, or one of the regular stations into a shunting yard. For this operation, we select the yard or station with the highest ratio between its flow, calculated in the previous algorithm iteration, and its maximum capacity.

3. relocating the marshaling or shunting yard to a connected station, if flow-maximum capacity ratio of the yard is low while the same ratio of the nearby station is high. The exact thresholds for the “high” and “low” ratios will be defined during the algorithm fine-tuning.
Inspired by [3], the VNS algorithm also keeps track on the success of the operators and favors the ones whose application has the higher probability of reaching a better solution.

For routing the freight after the yard locations are determined, we use the prioritized passenger assignment algorithm [4], applied to commodities. The commodity priorities are determined by the business rules of the company that operates the freight transport. The assignment algorithm also takes into account the capacities of the used marshaling and shunting yards, and train stations. This algorithm requires a low computational time, which makes it suitable for solving the real-life problems, or subproblems, as it is used in this case in particular.

We intend to verify the results of the developed heuristic by comparing them with the results obtained by solving the described optimization problem with CPLEX on a subset of network and demand data.

3.2 Anticipated Results

With the presented heuristic algorithm, we expect to design a network with lower freight transport and yard operation costs in comparison to the costs of the current network. The cost differences between the obtained, less expensive network and the current one will provide SBB Cargo incentives of the potential savings if some of the yards are established, closed or relocated. Also, the resulting cost obtained by applying only the assignment algorithm to a particular configuration of marshaling and shunting yards, can be used to assess the impact of a planned infrastructural change. By the time of the conference, we expect to have the heuristic algorithm implemented and the first results available.

References


Dynamic Electric Vehicle Routing with Mid-route Recharging and Uncertain Availability

Nicholas D. Kullman
Department of Computer Science
University of Francois Rabelais, Tours, France
Email: nicholas.kullman@etu.univ-tours.fr

Justin C. Goodson
John Cook School of Business,
St. Louis University, St. Louis, Missouri, USA

Jorge E. Mendoza
Department of Computer Science
University of Francois Rabelais, Tours, France

1 Introduction

As we shift towards sustainable transport, electric vehicles (EVs) are becoming more popular in supply chain distribution functions. However, EVs pose operational challenges to which their conventional petroleum-based counterparts are immune. For instance, EVs’ driving ranges are often only 25 percent that of conventional petroleum-based vehicles’ (CVs), charging infrastructure is still relatively sparse compared to the network of refueling stations for CVs, and the time required to charge an EV can range from 30 minutes to 12 hours depending on charging technology - orders of magnitude longer than the time needed to refuel a CV.

There are two general approaches to overcoming these operational challenges. The first is a simple approach in which routes are restricted to the vehicle’s autonomy. That is, the EV is routed back to the depot when its battery nears depletion so it may charge overnight in preparation for the subsequent day’s deliveries. In the second approach, the EV is allowed to perform mid-route recharging by taking advantage of charging infrastructure in the field.

In their 2016 study, Montoya, et al. showed that the second approach offers cost savings, because mid-route recharging allows for a decrease in the total distance traveled and an increase in the utilization of a single EV, thereby reducing the number of vehicles and drivers needed [1]. However, like the others that consider mid-route recharging (e.g., [2]),
the Montoya study makes the assumption that the charging stations (CSs) are always available to the EV when it arrives to charge. In reality, this is often not the case. Because charging station infrastructure is limited and EVs require significant time to charge, charging stations will often be unavailable when an EV arrives and the EV may be forced to queue. This discrepancy between modeling assumptions and reality has thus far prohibited logistics companies from implementing mid-route recharging, despite the suggested cost savings.

Our work aims to reduce this discrepancy by more realistically modeling both the uncertainty in availability and the queuing processes at public charging infrastructure. We model the EV Routing Problem with Mid-route Recharging and Uncertain Availability (E-VRP-MRUA) as a Markov decision process and implement a stochastic dynamic programming (SDP) solution for which we propose four routing policies. Further, using an information relaxation and information penalties, we establish a lower bound for the value of the optimal policy. Our results show that for a subset of our problem instances, our policies perform within approximately 5% of the optimal policy, providing encouragement for logistics companies to take advantage of the increases in capacity offered by mid-route recharging and thus extend the utility of EVs as delivery vehicles.

2 Problem description

The E-VRP-MRUA consists of a set of known customers $\mathcal{N}$ and charging stations $\mathcal{C}$ and a single EV. At time 0, the EV begins at the depot. It then traverses the complete graph on $\mathcal{N} \cup \mathcal{C}$. The vehicle must visit each customer $i \in \mathcal{N}$ and then return to the depot.

Between customer visits, the EV may elect to visit a CS $c \in \mathcal{C}$. The vehicle may charge if there are available charging terminals ("chargers"), or it may elect to join the queue if all chargers are in use. Let the number of chargers at CS $c$ be $\psi_c$. We assume that the $\psi_c$ chargers at the CS are identical, although charging technology may differ between charging stations. We further assume that the depot is always available for charging but all other charging station queue lengths are unknown prior to arrival.

We model waiting line dynamics at a CS $c$ as M/M/$\psi_c$ – a pooled first-come-first-served queue with a system capacity of $\ell_c \geq \psi_c$, where $\ell_c$ is chosen such that the system capacity is practically infinite. We consider a continuous-time Markov model on the state-space $\{0, 1, \ldots, \ell_c\}$ and assume that the the inter-arrival time of vehicles to $c$ and the service times of the chargers at $c$ are exponential random variables with known parameters $p_{c,x}$ and $p_{c,y}$, respectively. When a station is available, the vehicle may restore its charge to full capacity $Q$ or to an intermediate capacity.

The problem terminates when the EV has visited all customers and returns to the depot. The objective of the E-VRP-MRUA is to find the routing policy that minimizes
the total expected time for the EV to visit each customer in \( \mathcal{N} \), including travel time, charging time, and queuing time.

3 Solution methods

3.1 Heuristic policies

We have constructed four policies to solve the SDP: a myopic policy, a one-step rollout of the myopic policy, a static fixed-route policy, and a post-decision rollout of the fixed-route policy. For the fixed-route policy, we construct a route by computing the minimum-length Hamiltonian path from the current location to the depot that visits all remaining unvisited customers. If this route is not energy-feasible, we solve the fixed-route vehicle charging problem (FRVCP) using the labeling algorithm described in [3], which performs optimal insertion of charging stations and charging decisions.

3.2 Lower bounds

While we seek to produce policies that perform favorably relative to industry methods, gauging policy quality is hampered by the lack of a strong bound on the value of an optimal policy. Without an absolute performance benchmark, it is difficult to know if a policy’s performance is “good enough” for practice or if additional research is required to improve the routing scheme. To establish a bound on the value of an optimal routing policy, we combine information relaxation techniques [4] with mixed integer programming (MIP) methods to estimate the expected value of an optimal policy with perfect information, i.e., the performance achieved via a clairvoyant logistics planner.

We do this by granting the decision maker access to perfect information, thereby removing any uncertainty from the problem, and formulating the “perfect information dynamic program” (PIDP). We then formulate and solve a “perfect information MIP” (PIMIP), which we show to be equivalent to the PIDP.

To improve on the perfect-information bound, we develop computationally tractable information penalties, which punish the logistics planner for using information about the future to which they would not naturally have access. We impose these penalties via modifications to the PIMIP, resulting in the “penalized PIMIP” (PPIMIP).

4 Results

We simulated each policy under 50 realizations of uncertainty for 528 different instances. The instances vary in the number of customers and CSs (10-20 and 2-3, respectively), customer location method (random, clustered, and a hybrid), and average CS utilization (10%, 20%, ..., 80%).
Because solving the PIMIP and PPIMIP is computationally expensive, we have only been able to compute these bounds for a small subset of instances to date: those with 10 customers, 2 charging stations, and only 20%, 50%, and 80% average CS utilization. The results are shown in Figure 1. We find that the implementation of lookahead techniques improves upon the performance of the base policy alone in the case of both the myopic and fixed-route policies, with the rollout of the fixed-route policy being the strongest performing policy. For the subset of instances for which lower bounds were computed, we find that this policy performs within 5% of the optimal policy under high CS utilization and within 2% of the optimal policy under low-to-moderate CS utilization. Additionally, we find that the penalty only serves to improve the lower bound in the case of high CS utilization; for low-to-moderate utilization, the bound is unchanged.

Figure 1: *Left:* Performance of the heuristic policies constructed to solve the SDP. *Right:* Performance of the best policy (“Rollout + Fixed Route”) relative to the lower bounds established with information relaxations.

5 Conclusion

We introduce and solve the E-VRP-MRUA, which attempts to more realistically model the uncertainty in EVs’ access to charging stations. We also implement information relaxation techniques to obtain lower bounds for the optimal policy and impose information penalties to tighten the bounds; we believe this is a first in the field of vehicle routing. Using the information penalties, we find that for a subset of our problem instances, our dynamic policies perform within 5% of the optimal policy.
References


A Data-Driven Approach to the Vehicle Routing Problem with Time Windows under Uncertain Travel and Service Times

Szymon Albiński
School of Management, Technical University of Munich
Email: szymon.albinski@tum.de

Stefan Minner
School of Management, Technical University of Munich

1 Introduction
The trend to introduce highly customer-oriented services poses a great challenge for service providers. A prominent example is the concept of attended home delivery, where parcel companies try to reduce the time windows for guaranteed delivery. Besides the delivery to end-customers, the Vehicle Routing Problem with Time Windows (VRP-TW) finds also application in the planning of the supply of inner city stores or the planning and scheduling of on-site technician services. In the classical formulation, the travel times are assumed to be deterministic. This assumption does not hold in real-life cases since factors like congestion, road works or changing weather conditions make the travel times volatile. As a result, customers may be visited with delay. This can lead to additional costs due to additional rides or to customer dissatisfaction and future lost sales.

A solution to this problem is the stochastic VRP-TW (SVRP-TW), where the travel times are assumed to be uncertain. The literature on SVRP-TW can be divided into two main streams: chance constrained programming and recourse formulations [1, 2]. A third approach for solving a VRP-TW with uncertain travel times is proposed by Agra et al. [3]. They apply a robust optimization formulation and use different inequalities to find exact solutions. So far, all publications use parametric models to estimate the travel times. That is, the distributions of the stochastic travel times are estimated from historical data before they are used for optimization. Thus, wrong assumptions on the underlying distribution or inaccurate estimates of the corresponding parameters can incur large deviations in the optimization.

We propose a model where uncertainty is represented in a non-parametric way based on historical data. Our prescriptive analytics approach integrates the steps of prediction and optimization. Following the idea of Beutel and Minner [4], we do not parametrize the travel times but instead assume that they are distribution-free. This allows us to include a data-
driven regression of the travel times in the mixed-integer linear programming formulation of the deterministic VRP-TW. As a consequence, we can study the influence of several causal factors (e.g. time and type of day, weather, season, holiday) on the travel times. Service times can also be influenced by several factors, such as type of load, filling level or service facilities. Recently, Errico et al. proposed a chanced constrained program that uses stochastic travel times [5]. We show how our model can be extended in a straight-forward way to incorporate uncertain service times.

The VRP-TW is an NP-hard problem. Extending this with additional regression constraints increases the complexity of the problem. Therefore, we study efficient methods to compute exact solutions.

In this work we address the following research question: How can Big Data be used for solving the VRP-TW under uncertainty and how does this data-driven approach perform relative to the stochastic VRP-TW? Our three main contributions are: (1) Formulation of the data-driven Vehicle Routing Problem with Time Windows under uncertain travel or service times. (2) Development of efficient solution methods. (3) Evaluation of the model and comparison with stochastic approaches.

2 Model formulation

We consider a graph \( G = (N_0, E) \) with \( N_0 = N \cup \{0, n + 1\} \), where \( \{0, n + 1\} \) denotes the depot and \( N = \{1, \ldots, n\} \) the customers. We assume that the customers need a service, rather than having ordered goods. Thus, we study an uncapacitated VRP-TW. The interval \([e_i, l_i] \) defines the time window for customer \( i \in N \), \( s_i \) is the corresponding service time. \( t_{ijp} \) denotes the historical travel times from \( i \) to \( j \) observed in period \( p \in P = \{1, \ldots, \psi\} \). Since we simultaneously solve the linear regression and the routing problem, we use three groups of decision variables. The real-valued coefficients \( \beta_{f_{ij}} \) fit the \( \phi \) causal factors \( Y_{fp}, f \in F = \{1, \ldots, \phi\} \), best to \( t_{ijp} \). The binary decision variables \( x_{ijp} \) equal one if arc \((i, j)\) is traversed in period \( p \). The continuous variable \( b_{ip} \) defines the time point when the vehicle arrives at customer \( i \). \( a_{ip} \) defines the time point when the vehicle would arrive at customer \( i \) if the same tour is evaluated under the historical travel times \( t_{ijp} \). The travel cost from \( i \) to \( j \) is denoted by \( c_{ij} \). The fix cost for using a vehicle is \( c_0 \). The data-driven VRP-TW then reads as follows:

\[
\min \sum_{i,j \in N_0} \sum_{p \in P} c_{ij} x_{ijp} + c_0 \sum_{j \in N} \sum_{p \in P} x_{0jp} + \sum_{i \in N_0} \sum_{p \in P} (u_i w_{ip} + v_i z_{ip}) \\
\text{s.t.} \sum_{j \in N_0} x_{ijp} = 1 \quad \forall i \in N_0; p \in P \\
\sum_{j \in N_0} x_{jip} = 1 \quad \forall i \in N_0; p \in P
\]
\begin{equation}
   b_{ip} + s_i + \sum_{f \in F} \beta_{fij} Y_{fp} - b_{jp} \leq (1 - x_{ijp}) M \quad \forall i, j \in \mathbb{N}_0; p \in P
\end{equation}

\begin{equation}
   a_{ip} + s_i + t_{ijp} - a_{jp} \leq (1 - x_{ijp}) M \quad \forall i, j \in \mathbb{N}_0; p \in P
\end{equation}

\begin{equation}
   a_{ip}, b_{ip} \geq e_{ip} \quad \forall i \in \mathbb{N}_0; p \in P
\end{equation}

\begin{equation}
   w_{ip} \geq a_{ip} - l_{ip} \quad \forall i \in \mathbb{N}_0; p \in P
\end{equation}

\begin{equation}
   z_{ip} \geq b_{ip} - l_{ip} \quad \forall i \in \mathbb{N}_0; p \in P
\end{equation}

\begin{equation}
   w_{ip}, z_{ip} \geq 0, \beta_{fij} \in \mathbb{R}, x_{ijp} \in \{0, 1\} \quad \forall i, j \in \mathbb{N}_0; p \in P; f \in F
\end{equation}

In the objective function (1), we minimize the total traveling cost over all periods and
the fix cost per vehicle while penalizing violations of the upper limits of the time-windows.
In the two-index formulation, it is not possible to limit the number of used vehicles. To
minimize this number, we charge in the objective function a fix cost \( c_0 \) for every vehicle
that leaves the depot. In constraint (4), which states that the customers have to be visited
in a feasible order, we insert the linear regression of the travel times. We then evaluate
the routing decisions with the historical travel times by computing the arrival times \( a_{ip} \)
in constraint (5). Deviations in the planning (constraints (4)) and evaluation (constraints
(5)) are both penalized. For that purpose, we introduce auxiliary decision variables \( w_{ip} \)
and \( z_{ip} \) weighted by \( u_i \) and \( v_i \). These auxiliary variables are defined by soft-constraints
(7) and (8). Earlier arrivals are not explicitly penalized but instead excluded by the hard
constraints (6). Constraints (2) and (3) relate to the deterministic VRP-TW and ensure
a proper routing of the vehicles.

To consider uncertain service times, we have to replace \( s_i \) in constraints (4) by the
causal regression \( \sum_{f \in F} \gamma_{fij} Z_{fp} \), where \( Z_{fp} \) denotes the causal factors and \( \gamma_{fij} \in \mathbb{R} \) the
regression variables. All other constraints of the model remain unchanged.

### 3 Solution approach

To find an exact solution to the data-driven VRP-TW, we solve the problem on a small
subset \( P' \subseteq P \) of periods. The initial \( P' \) contains the outlying periods by means of travel
times. That is, they represent the extrema of the travel times. \( P' \) is determined by solving
a sphere packing problem over the historical data. Next, we check if the resulting \( \beta_{fij} \) (or
\( \gamma_{fij} \)) cause violations of the time-windows in periods \( P \setminus P' \). If this is the case, we add the
period with the highest penalty cost to \( P' \) and repeat the steps until no violation occurs.

The optimization of the routes and of the linear regression is linked by constraint
(4). A promising future research direction would be to exploit this special structure for
developing a column generation approach.

To obtain fair estimates of the coefficients \( \beta_{fij} \) and \( \gamma_{fij} \), the data-driven model needs to
be calibrated. This so-called training includes some pitfalls. Fitting errors occur when too
many or too few causal factors are considered whereas sample errors are a consequence of too small or to big training data. Evaluation errors follow from the use of a static training set instead rolling horizon data. Special attention should be paid to out-of-sample evaluations, the purpose of which is to verify the coefficients on data that were not used for training. To reduce these errors, we apply methods from machine learning to optimize the parameter selection. Furthermore, we do out-of-sample testing.

4 Conclusion
We present a new approach that shows how Big Data can be used for solving the Vehicle Routing Problem with Time Windows under uncertainty. We suggest a non-parametric, distribution-free model that combines causal forecasting of the travel and service times with the optimization of routing and scheduling. For this model, we propose a solution method based on the successive insertion of historical data from periods with violated time-windows. To improve the regression, we apply methods from machine learning. The model will be evaluated and compared to solutions of the stochastic VRP-TW. Further details of the solution method and numerical results will be presented at the conference.

References


The stochastic and time-dependent single vehicle field service personnel routing and scheduling problem

Edison Avraham and Tal Raviv
Department of Industrial Engineering
Tel Aviv University, Tel Aviv, Israel
Email: edisonav@post.tau.ac.il

1 Introduction
Field service personnel spend most of their working day on the road or at their customers. The common practice is to schedule assuming deterministic and time independent travel and service times. These simplifying assumptions lead to suboptimal solutions when implemented in real life. Recent advancement in mobile computing technology have allowed the collection of real data that provide a better understanding of the stochastic and time dependent nature of the travel times. Such understanding can be used in more accurate optimization models.

Routing and scheduling of field service personnel is typically subject to time window constraints. Two types of time windows, hard, as well as soft, have been studied. In the context of stochastic travel and service times, considering hard time windows seems inadequate, since such time windows generally leads to infeasible solutions.

In this study, we discuss a stochastic and time dependent version of the single vehicle problem where the vehicle leaves its depot, visits and serves some (but not necessarily all) customers and returns to the depot. Each customer bears a reward that the vehicle collects if he arrives. Late arrivals at customers are allowed and incur a penalty increasing in the extent of the lateness. This type of vehicle routing problems is referred to as Orienteering-type problems.

We formulated the orienteering version of this single vehicle problem as well as the TSP version of the problem. Due to the intricate nature of these problems even small instances cannot be solved effectively by a commercial solver. Therefore, we developed branch and bound (B&B) algorithms that can solve to optimality instances of up to 15 customers. Later, we devised Tabu search (TS) algorithms that can find high quality solutions for larger instances in a short time. Next, we conducted numerical experiments using real data of travel times. These experiments showed the added value of considering stochasticity and time dependency.

Few studies deal with time dependency and stochasticity simultaneously. Nahum and Hadas [1] study the stochastic time-dependent VRP (TD-S-VRP). They present a mathematical model for the problem that limits the probability of exceeding some given traveling time. The objective function is minimizing the average total travel time. The authors develop a solution heuristic based on the saving heuristic (Clarke and Wright [2]). Tas et al. [3] address a variant of the TD-S-VRP with soft time windows. The objective function is minimizing a weighted sum of the expected transportation and service (lateness and earliness) costs. The authors assume independent travel times that follow a gamma distribution and derive exact as well as approximated distributions of arrival times. They devise a TS and an ALNS algorithms for the problem. Verbeeck et al. [4], [5] study the deterministic
as well as the stochastic time-dependent orienteering problem with time windows. Ant colony algorithms are used to solve the problems.

In all the studies, it is assumed (explicitly or implicitly) that the travel times follow independent distributions. We believe that this simplifying assumption largely misrepresents the reality where the traffic conditions in close geographical locations is highly dependent. This study is the first to introduce a model and solution methods, both exact and heuristic, for routing and scheduling under time-dependent travel times and general stochastic travel and service times (without the independence assumption).

2 Problem definition
In this section, we formally define the stochastic and time dependent orienteering problem with soft time windows (S-TD-OR-STW). The stochasticity of the travel and service times is modelled by a set of scenarios that relate to a single working day rather than by explicit distributions. While this approach may result in sacrificing some accuracy, it has two important advantages: 1. It is relatively easy to create the input for the problem based on historical data; 2. Scenarios readily capture the inherent and complicated dependency between the travel times of journeys that are spatially or temporally close. Note that it is impractical to estimate the joint distributions of all these travel times, and thus many previous studies assumed independency. We believe that the independence assumption is naïve and may lead to plans that are too optimistic because there is a positive correlation between travel times in the same areas and times of the day.

The S-TD-OR-STW is defined as follows: There are N customers and a depot. Each customer has a time window for service beginning. The upper bounds of the time windows are soft while the lower bounds are hard. That is, arriving at a customer after the end of the customer's time window incurs a penalty, which is increasing in the extent of the lateness. In addition, each customer is associated with a reward gained from serving it. A single vehicle is available to serve the customers during a single working day. The vehicle departs from the depot not before a given time. There is a given set of scenarios, each defined a service time for each customer and a travel time between each pair of locations. The travel time between each pair of locations is time dependent. That is, the departure time determines it. The goal is to select a subset of the customers to serve and a sequence to visit them so as to maximize the total reward net of the expected penalty.

3 Methodology
The complex structure of the problem makes it very hard to formulate the problem as an integer program with reasonable number of variables and constraints. Therefore, we developed a customized B&B algorithm for the problem and an inexact solution approach based on the TS metaheuristic.

3.1 B&B algorithms
A solution is a sequence of served customers. The algorithm begins with a sequence that contains no customers and a solution value of 0. At each iteration of the process, new potential solutions are constructed by adding a single customer that has not been served yet to the sequence of served customers at the end of that sequence. The value of the solution following that insertion is calculated. If this insertion is profitable, an upper bound is calculated and the potential solution is inserted to the list. Note that an optimal solution contains only profitable customers since a sequence with non-profitable customers is always inferior to a similar sequence that excludes these customers while maintaining the order of the profitable customers.
The upper bound of a solution is equal to the sum of its current value and an upper bound on the profit that can be obtained from visiting some subset of the remaining customers. The latter is calculated as follows: Let \( i \) denote the customer at the end of the sequence and \( \text{Seq} \) denote the customers visited before \( i \). Let \( \pi_j \) be the sum of the net profit over all the scenarios that can be obtained from visiting customer \( j \in N \setminus \{\text{Seq} \cup \{i\}\} \) immediately after customer \( i \). Now, \( UB = \sum_{j \in N \setminus \{\text{Seq} \cup \{i\}\}} \max(0, \pi_j) \). The process ends when the list is empty.

### 3.2 TS algorithms

The initial solution of the algorithms is equal to the sequence of customers calculated by the EDD rule. We define the neighbourhood of a given solution as all the solutions that can be obtained by one of the following actions:

- Adding an unserved customer to the sequence
- Removing a served customer from the sequence
- Replacing a served customer with an unserved customer
- Moving a served customer along the sequence

Each obtained solution in the neighbourhood undergoes re-optimization that iteratively removes all non-profitable customers, starting from the earlier ones so as to create a sequence where all customers are profitable. The search procedure ends when a certain stopping criterion is reached.

For each of the above action we defined a reversing action which is Tabu for a certain number of iterations.

### 4 Numerical experiments

We have gathered data of real-time travel times from Google Map between 19 locations in central Israel, that is, one depot and 18 customers, at different times of the day along a period of 60 working days. The time granulation is 5 minutes, i.e. a single time unit represents a 5-minute period. This data constitutes 60 scenarios of time-dependent travel times. We generated 100 sets of service times for the customer in each realization. These sets are divided to five levels that differs in their variance but not in their expected values.

The length of the working day was set at 108 five minutes time periods (9 hours). Two types of time windows were considered. In Type 1, there are three non-overlapping time windows of three hours each, and the customers are equally divided between them. In Type 2, one-half of the customers can be served any time during the day, and the rest are assigned to time windows of two hours. That is, 200 problem instances that contain a depot and 18 customers were created.

Finally, we created 400 smaller test instances. 200 with 12 customers and 200 with 15 customers. Each smaller instance contains the depot and some randomly selected subset from the 18 customers. For these instances, the service times are increased while travel times remain unchanged. The penalty function of each customer is quadratic in the lateness and proportional to the reward.

Recall that each instance has 60 scenarios. We divided these scenarios into two groups. 40 scenarios comprised the training data. These scenarios are the input for the various solution methods we developed. 20 scenarios comprised the test data. These scenarios are used to evaluate the solutions previously found using new data.

Next, we summarize the results of our experiment are presented. For brevity, we present only the instances of 12 customers. Note that this is a typical number of customers that is served by field
service crew in a working day. Our test machine is a standard laptop (Intel i7-6500U, 2.5 GHz, with 8 GB RAM). The algorithms were coded in Python.

Summary of the results of our experiment are presented in Table 1. For each of the types of the time windows we present the processing time of the B&B algorithm and the results obtained by B&B and TS after 10 seconds. We present the average, median and maximum values of the 100 instances in each dataset.

Table 1: summary of the results

<table>
<thead>
<tr>
<th>Date set</th>
<th>B&amp;B CPU time (seconds)</th>
<th>B&amp;B Optimality Gap after 10 second</th>
<th>TS Optimality Gap after 10 second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1 TW</td>
<td>11.7</td>
<td>8</td>
<td>84</td>
</tr>
<tr>
<td>Type 2 TW</td>
<td>323.4</td>
<td>211.5</td>
<td>1108</td>
</tr>
</tbody>
</table>

Clearly, the type-2 time windows instances are harder to solve. The TS algorithm obtains median optimality gaps of less than 1% for the two types of time windows. It seems that the B&B algorithm is less suitable than the TS to cope with the type-2 time window instances when given a 10 seconds time limit.

Next, we wanted to check the actual contribution of solving the stochastic and time dependent problem rather than following the traditional approach and plan the route based on the nominal (average) travelling and service time. For this end, we solved a stochastic but time independent version of the problem (Stoc), a time dependent deterministic version (TD) and deterministic time independent version (Basic). We applied these solution as well as the exact solution obtained by our B&B algorithm (B&B) and the solution obtained after 10 seconds by our TS heuristic (TS) on the test data. The results are illustrated in Figure 1.

Clearly, the solution of S-TD-OR-STW problem yields better route and schedule than the one obtained by the solution of the basic problem, especially for the harder instances with the Type-2 time window. It seems that the consideration of the stochasticity in the problem is responsible for most of the improvement. Additionally, it is evident that solving the optimal solution of the training scenarios does result in over-fit with respect to test scenarios.

Figure 1: Average ratio between the solutions of the various versions of the problems

5 Conclusions

We introduced a single vehicle stochastic and time dependent routing problem with soft time windows and present an exact and a fast heuristic solution method for it. The later can be used as subroutine in algorithms that solve more elaborated fleet routing and scheduling problems. We have demonstrated the added value of considering the stochastic and time dependent nature of the problem over a solution that is based on the average scenario.
References


Strategic network design at the Polish Post

Maciek Nowak
Quinlan School of Business,
Loyola University, Chicago, USA
Email: mnowak4@luc.edu

Mike Hewitt
Quinlan School of Business,
Loyola University, Chicago, USA

Bogumił Kaminski
Collegium of Economic Analysis,
Warsaw School of Economics, Warsaw, Poland

Michał Pliszka
Polish Post
Warsaw, Poland

The Polish Post has changed dramatically in the last 25 years. Even after the fall of communism, government protection enabled a monopoly, with steady profits and little need to modernize. As would be expected under these circumstances, the logistics infrastructure was unnecessarily complex, rigid and expansive. The country was divided into regions, with little centralized decision making. There were no established metrics for efficiency, productivity or throughput as there was no need to find improvements in any of those areas. Naturally, customer service suffered, with extremely long lead times, little to no adoption of new technology, and no value added services.

Fast forward to today’s market, where there is considerable competition from the private sector, primarily in the form of parcel carriers such as DHL, UPS and FedEx, with rapidly increasing customer expectations for both timely delivery and information, and with a changing product, as the growth in parcel delivery parallels the decline in personal letter delivery. As competition has driven costs down and quality up, demand uncertainty has increased. The Polish Post now requires not only cost-efficient and high-quality processes, but also the flexibility to dynamically adjust supply to meet demand. An initiative to address these changes involves the strategic repositioning of resources within their network and the development of tactical plans for inter-hub transportation.

The Polish Post operates a traditional hub and spoke network, in which hundreds of postal outlets feed dozens of smaller terminals that connect to 15 larger hubs. As the largest employer in Poland, the Polish Post has thousands of drivers and warehouse personnel that execute the services delivered as a part of the network. This network connects the 16 provinces of Poland, with labor costs varying considerably from one province to another. A service plan coordinates the scheduling and movement of shipments through this network using the appropriately allocated resources. The development of such service plans has long been assisted by solving variations on the Service Network Design (SND)
problem [2], which can prescribe the choice of paths for shipments, their schedules (departure times), and the services and resources necessary to execute them, while achieving economic and service-quality targets. In this research, we build such a plan, extending previous work on the SND problem to include characteristics of the large scale Polish Post network, with a particular focus on heterogeneous resources.

The model that defines this problem is an extension of that found in [6]. That model assumes that each customer shipment has a known origin and destination, but that there is uncertainty regarding its volume. This uncertainty is incorporated into the decision-making process through the use of a two-stage, scenario-based model. While similar to traditional stochastic network design models [9, 3, 4], the first stage also includes strategic decisions regarding the number of resources acquired, the allocation of new resources, and the re-allocation of existing ones. Our model incorporates uncertainty, as well as other aspects of the problem, in the same fashion. However, the model in [6] assumes that capacity and speed are homogeneous across all resources. The Polish Post maintains a heterogeneous fleet of trucks, ranging in size from vans to semi-trucks. The significant majority of shipments, by number and volume, are moved by trucks that fall within four classifications. Table 1 provides the range of available capacities, as well as the cost per KM, for each class. These costs remain the same for every region of Poland. However, driver costs vary by province and drivers are further segmented depending on their certification to operate vehicles over 3.5 tons (Class I is the only that doesn’t require this certification).

Given these problem characteristics, we extend the definition of a service to choosing not only which transportation moves are executed (and when), but also their capacity and speed. For example, for the same pair of terminals, we model that the Polish Post may want to offer both a Class IV truck that takes two days and a Class I truck that only takes one day. As a service’s capacity can dictate the choice of transportation asset used, we also model the use and management of multiple types of transportation assets. Finally, we model the use and management of multiple classes of supporting resources that are needed for the movement of a transportation asset and that some transportation assets may require specific types of supporting resources (e.g. Class II, III and IV trucks require drivers certified to operate vehicles over 3.5 tons).

We extend the algorithm applied in [6] to solve this model, incorporating heterogeneous resources with varying capacity, speed, and supporting requirements, while still incorporating uncertain demand. As stochastic programs are difficult to solve exactly, this
algorithm utilizes a column-generation matheuristic. Making explicit use of the mathematical formulation in parts of the search, matheuristics have (computationally) proven to be an effective solution approach for hard combinatorial optimization problems. They have primarily been applied to deterministic problems [7, 10, 1], but with the increased availability of “big data” making possible the development of representative distributions for problem parameter values, more researchers are applying matheuristics to stochastic problems.

The proposed matheuristic takes the form of a neighborhood-based search scheme, where parts of the search space are fixed and the resulting restricted mixed-integer problem is solved exactly. This space-decomposition idea follows the successful contributions of [8], for vehicle routing, and [7] and [5] for LTL service network design. At an iteration of this matheuristic, we presume a known solution composed of paths and cycles used in that solution. A cycle models a sequence of possible movements during the schedule length for a resource assigned to a terminal, beginning and ending at that terminal, while a path constitutes a scheduled service or sequence of services from a shipment’s origin to its destination. Next, we determine the neighborhood to search, wherein a neighborhood includes both a set of paths that can be taken to service demand in each scenario, chosen from those known, and a set of cycles, chosen from those known. Then, to search that neighborhood we solve the model with an off-the-shelf optimization solver. We consider searching two different neighborhoods in the course of executing this algorithm. To create the first neighborhood, we first generate new cycles and paths before determining which cycles to include in the set of candidates. However, the neighborhood consists of all known paths. The second neighborhood does the opposite, including all known cycles, but only the paths for a limited set of scenarios.

This algorithm was tested in [6] and results indicate that it produces quality solutions for large scale problems. Using a range of problem instances with 144 nodes and 600 services, this matheuristic outperforms a column-generation algorithm that generates a broader selection of paths and cycles without the neighborhood search, with an average improvement in objective function of 5.44% and nearly 60% of instances with an objective gap greater than 3%. Also, the results indicate that the algorithm often uses nearly all the time allotted to find its best solution, suggesting it is thoroughly searching the solution space.

This research will present similar metrics for the modified version of the algorithm incorporating heterogeneous resources with varying capacity, speed, and supporting requirements. Using one month of demand data on the Polish Post network, the algorithm will be used to indicate the improvements that may be made in terms of resource allocation and service scheduling.
References


A Study on Travel Time Stochasticity in Service Network Design with Quality Targets

Giacomo Lanza
Department of Electrical, Electronic and Information Engineering,
University of Bologna, Bologna, Italy

Teodor Gabriel Crainic
CIRRELT and Department of Management and Technology,
Université du Québec à Montréal, Canada

Walter Rei
CIRRELT and Department of Management and Technology,
Université du Québec à Montréal, Canada

Nicoletta Ricciardi
CIRRELT and Department of Statistical Sciences
Sapienza University of Rome, Rome, Italy
Email: nicoletta.ricciardi@uniroma1.it

1 Problem Description

Freight consolidation is one of the many strategies that carriers apply to lower transportation costs and consequent service prices. In a consolidation-based transportation network, freight associated with multiple customers are combined and moved on common vehicles (e.g., railroads, less-than-truckload motor carriers, ocean shipping lines). The design of a consolidation-based service network is a complex planning process involving interrelated and interdependent decisions traditionally faced at tactical level by carriers and supported by service network design (SND) methodology. The scope of SND is to produce an operation plan that achieves the economic and quality targets of the carrier. The plan specifies the services to operate (physical route, stops, frequency, vehicle type, capacity, schedule) to meet the estimated transportation demand and how to move the freight (services used and terminals visited) through that service network [1]. Quality targets are usually related to the reliability of operations of the service network as measured by the regular on-time arrival of services at stops with respect to the published schedule, and the regular on-time arrival of freight at destination with respect to the agreed-upon time of delivery. The
plan is made for a certain time interval, called schedule length (e.g., a week), and applied repeatedly over a longer time period, called planning horizon (e.g., the next season) [1].

There is a significant body of literature on developing methodology to address deterministic SND models. Formulations considering uncertainty explicitly have been less proposed in the literature and only few contributions have dealt with stochastic times. The vast majority of the proposed model formulations, in fact, assume travel times as deterministic parameters, generally built on point forecasts based on historical data. Yet, travel times are influenced by a variety of random events, e.g., traffic congestion and heavy weather conditions, which may cause significant differences between estimated and observed travel times. Additional economic costs related to crews and resource utilization, as well as fines and loss of reputation for not respecting planned service arrival times and freight due dates may be caused by these fluctuations [2]. Consequently, a deterministic time assumption, that is, assuming perfect knowledge of future time realizations, does not represent an accurate and realistic approximation of actual travel times and may underestimate final costs.

The topic of our research is, thus, the study of a SND formulation in which travel times and the reliability (as previously defined) of the final service network are explicitly considered. Our objective is to enhance the understanding of the relations between the characteristics of a service network and its robustness, in terms of respect of the service schedules and delivery due dates, given business-as-usual fluctuations of travel times [3]. Main questions we explore in our work: What is gained by integrating information about the stochastic nature of travel times directly into the tactical planning methodology? Are different patterns, either in the service selection or in freight itineraries, suggested when such information is integrated into the model? Is the resulting transportation plan actually more robust with respect to travel time fluctuations? What characteristics are more important in producing such a robustness?

We proposed a stochastic travel-times SND formulation aimed at defining a cost-efficient and reliable service network ([4],[5]). An extensive experimental campaign was performed using a large set of random generated instances. In order to obtain results with the lowest bias possible, we focus on optimal solutions only, for both the deterministic and the stochastic formulations. For this reason, the experimentation was performed on small and moderate-sized problem instances, which a well-known commercial mixed-integer software was able to solve optimally in both cases. The analysis and comparison of the stochastic and deterministic solutions provided the means to identify characteristics that appear to hedge or, at least, reduce the bad effects of travel time uncertainty on the performance of a service network.
2 Problem Formulation

The model we proposed aims for a cost-efficient transportation plan (selection of services, their schedule and the routing of the demand) such that service-quality (arrival times at stops) and demand-quality (freight delivery due-dates) targets are respected, as much as possible, over time and despite business-as-usual travel time fluctuations.

The problem is formulated as a network design model on a time-space network. In such a structure, the representation of the physical network is replicated for a number of periods, which discretizes the chosen schedule length. Services are modelled through arcs linking different terminals at different time periods (moving arcs) and terminal activities through arcs linking the same terminals at different time periods (holding arcs). Time is assumed deterministic on holding arcs, while travel times between different terminals are considered stochastic. We further assume that the probability distributions for these parameters are independent (e.g., in interurban problem settings, this hypothesis appears reasonable). The appropriate definitions of the decision sequence and the related information revelation process is particularly challenging for this problem context, both the service selection and the routing decisions are part of the plan and thus determined a priori before any observation of the stochastic phenomenon. We propose a two-stage stochastic linear mixed-integer programming formulation [6]. Design and routing decisions make up the first stage, while the service and demand targets are accounted for in the second stage, through a set of penalties, once travel time realizations become known. Penalties are proportional to the difference between observed and scheduled arrival times for both services at each stop and freight at the final destinations.

First stage decisions take the form of a set of integer-valued variables representing service selection and a set of continuous commodity-specific freight-flow variables. The constraints of this stage are the usual ones: flow conservation, linking/capacity and variable domain restrictions. A simple recourse is applied in the second stage, where penalties are computed for given travel time realizations and a chosen design.

Stochasticity is described in terms of a set of scenarios, possible travel time realizations, and associated probabilities. The objective is to minimize the fixed service-selection and variable demand-routing costs, plus the expected penalty costs following the application of the chosen plan to the observed travel time realizations, over the scenario set.

3 The Experimental Setting

A large number of instances with different characteristics, in terms of level of variability, number of commodities, wideness of delivery time windows and penalty costs, were used to solve the stochastic and deterministic formulations. The random event was described by a Truncated Gamma probability distribution, which provides the specific characteristics
required: a lower bound (since a minimum time to cover the distance between two point always exists), a rapid increase to a maximum (the most usual or observed travel time realization), a slow decrease with a tail skewed to the right, and an upper bound (since infinite travel times are not observed in normal conditions). The stochastic parameter is replaced by a point forecast, namely the mode of the distribution, in the deterministic formulation. Two analyses were performed. The goal of the first one was to quantify the benefits of explicitly considering the stochastic nature of travel times. Stochastic and deterministic solutions were evaluated through a Monte Carlo simulation and their set-up (service activation plus routing costs) and full (set up cost plus penalties) costs were compared. The second analysis is a structural comparison of the stochastic and deterministic solutions, aiming at answering the questions stated previously before. Three recourse formulations, service related only, demand related only, both, were considered in both analyses.

4 The Odysseus presentation

We first recall the problem setting and the two-stage mixed-integer linear stochastic formulation with simple recourse proposed. Particular attention is then given to the experiments, the analyzes performed and the results and insights obtained. Future research paths are also discussed.

References


Separable Lagrangian decomposition for the Knapsack Relaxation of Multicommodity Network Design

Antonio Frangioni, Bernard Gendron, and Enrico Gorgone

1 Dipartimento di Informatica, Università di Pisa, Email: frangio@di.unipi.it
2 Department of Computer Science, Université de Montréal and CIRRELT
3 Dipartimento di Matematica e Informatica, Università di Cagliari

1 Introduction

The Multicommodity Network Design problem (MND) is a general network design problem with many applications [1]. The data of MND is a directed graph \( G = (N, A) \), where \( F_i \) and \( B_i \) respectively denote the set of outbound and inbound arcs of node \( i \in N \), and a set of commodities \( K \), with each \( k \in K \) characterized by has a deficit vector \( b^k = [b^k_i]_{i \in N} \). Each arc \((a_+, a_-) = a \in A\) can only be used if the corresponding fixed cost \( f_a > 0 \) is paid, in which case the mutual capacity \( u_a > 0 \) bounds the total amount of flow on \( a \), while individual capacities \( u^k_a \) bound the flow of commodity \( k \). The routing cost \( c^k_a \) has to be paid for each unit of commodity \( k \) moving through \( a \). Restrictions are usually imposed on the subset of arcs that can be constructed; because the specific form of these is largely immaterial to our development we will just define the set \( Y \subseteq \{0, 1\}^{|A|} \) of all feasible design vectors \( y = [y_a]_{a \in A} \). A formulation for the problem then is

\[
\begin{align*}
\min & \sum_{k \in K} \sum_{a \in A} c^k_a x^k_a + \sum_{a \in A} f_a y_a \\
\text{s.t.} & \sum_{a \in F_i} x^k_a - \sum_{a \in B_i} x^k_a = b^k_i & i \in N \ , \ k \in K \\
& \sum_{k \in K} x^k_a \leq u_a y_a & a \in A \\
& 0 \leq x^k_a \leq u^k_a y_a & a \in A \ , \ k \in K \\
y \in Y
\end{align*}
\]

Efficiently computing tight lower bounds on its optimal value is crucial for solution approaches. Lagrangian techniques can be used for this [2, 3], among which the knapsack relaxation (KR) one dualizes the flow conservation constraints (2). Let \( \lambda \) be the multipliers vector, then the corresponding relaxation can be tackled as follows: one first solves \(|A|\) independent continuous knapsack problems to find the optimal solutions \( x^*_a(\lambda), a \in A;\)
then, these are used to compute the updated cost coefficients $f_a(\lambda)$, $a \in A$ for the design variables, and one solves the design subproblem $\min \{ f(\lambda)y : y \in Y \}$. Hence, the Lagrangian function does not have a sum-structure for which the components are completely independent from each other. This property is very useful to the algorithms tasked with finding the optimal Lagrangian multipliers [3], despite the fact that some form of separability is there. Our contribution is a reformulation of the standard master problem of Dantzig-Wolfe decomposition which exposes this structure in such a way that it can be algorithmically exploited.

2 The Proposed Reformulation

To simplify the exposition we now cast MND into the more general class of problems that, when a set of linking constraints is removed, can be solved by applying two separate solution approaches in cascade, for which one solution depends on the other one. For an index set $N = \{1, \ldots, n\}$, a set $Y \subseteq \{0, 1\}^n$ is given, together with $n$ sets $X_i \subseteq \mathbb{R}^{n_i}$ and the corresponding partitioned vector(s) of variables $x = [x_i]_{i \in N}$. Each $x_i$ is a semi-continuous variable governed by the corresponding $y_i$, in the sense that $y_i = 0 \Rightarrow x_i = 0$, while $y_i = 0 \Rightarrow x_i \in X_i$. This is obtained by imposing constraints $A_i x_i \leq b_i y_i$ which define a compact polyhedron when $y_i = 1$. The sets $X_i$ may entail integrality restrictions, but since we are concerned with Lagrangian relaxation we actually work with the convex hull of $X_i$; the relevant object is then the set $\bar{X}_i$ of the extreme points of $X_i$, so that $X_i = \text{conv}(\bar{X}_i)$. To further ease the notation we will assume that $0 \in \bar{X}_i$, although the approach can be extended to the case where this does not hold. An abstract formulation of our problem is:

\begin{align*}
\max \quad & dy + \sum_{i \in N} c_i x_i \\
\text{s.t.} \quad & Dy + \sum_{i \in N} C_i x_i = b \\
& A_i x_i \leq b_i y_i \quad i \in N \tag{8} \\
& x_i \in X_i \quad i \in N \tag{9} \\
& y \in Y \tag{10}
\end{align*}

The $m$ constraints (7) are the linking ones, whose removal makes the problem (much) easier to solve: the Lagrangian relaxation w.r.t. (7)

\begin{equation}
\phi(\lambda) = \lambda b + \max \left\{ (d - \lambda D)y + \sum_{i \in N} (c_i - \lambda C_i) x_i : (8), (9), (10) \right\} \tag{11}
\end{equation}

can be computed with the two-stage approach already alluded to. Defining

\begin{equation}
\phi_i(\lambda) = \max \left\{ (c_i - \lambda C_i) x_i : x_i \in X_i \right\} \tag{12}
\end{equation}

one clearly has that the Lagrangian function is

\begin{equation}
\phi(\lambda) = \lambda b + \max \left\{ \sum_{i \in N} (d_i - \lambda D^i + \phi_i(\lambda)) y_i : y \in Y \right\} \tag{13}
\end{equation}
For any \( y^* \) optimal for (13), \((y^*, \tilde{x})\) is optimal for (11) with \( \tilde{x}_i = 0 \) if \( y_i^* = 0 \), and \( \tilde{x}_i = x_i^* \) if \( y_i^* = 1 \), with \( x_i^* \) any one of the optimal solutions of (12). However, this process hides some of the structure of the problem, i.e., the fact that computing \( \phi \) requires solving \( n + 1 \) separate—but not independent—subproblems. Function \( \phi \) does not have the sum structure for which disaggregated master problems can be used, which are well-known to substantially improve convergence speed [3]. There is a simple way to make \( \phi \) separable: also relax (8) with multipliers \( \mu = [\mu_i]_{i \in N} \geq 0 \), yielding

\[
\phi(\lambda, \mu) = \lambda b + \psi(\lambda, \mu) + \sum_{i \in N} \psi_i(\lambda, \mu_i) \quad \text{with} \quad (14)
\]

\[
\psi_i(\lambda, \mu_i) = \max \left\{ (c_i - \lambda C_i - \mu_i A_i) x_i : x_i \in X_i \right\} , \quad (15)
\]

\[
\psi(\lambda, \mu) = \max \left\{ \sum_{i \in N} (d_i - \lambda D_i - \mu_i b_i) y_i : y \in Y \right\} . \quad (16)
\]

Clearly, (15) and (16) are efficiently solvable if (12) and (13), and \( \phi(\lambda, \mu) \) is a sum-function.

The drawback is that (8) can be many, which may well negate the advantage of having a disaggregated master problem.

We now propose an alternative method that obtains the same result, i.e., a sum-function, while using fewer \( (n) \) extra Lagrangian multipliers. Our proposal hinges on a modification of the classical Dantzig-Wolfe reformulation of the problem corresponding to the Lagrangian approach described above. Let \( YX \) the set of extreme points \((\bar{y}, \bar{x} = [\bar{x}_i]_{i \in N})\) of (11), that can be obtained (for any given value of \( \lambda \)) with the procedure illustrated above. Then, the Dantzig-Wolfe relaxation of (6)–(10) is

\[
\max \sum_{(\bar{y}, \bar{x}) \in YX} \left( d\bar{y} + \sum_{i \in N} C_i \bar{x}_i \right) \theta(\bar{y}, \bar{x}) \quad (17)
\]

\[
\sum_{(\bar{y}, \bar{x}) \in YX} \left( D\bar{y} + \sum_{i \in N} C_i \bar{x}_i \right) \theta(\bar{y}, \bar{x}) = b \quad (18)
\]

\[
\sum_{(\bar{y}, \bar{x}) \in YX} \theta(\bar{y}, \bar{x}) = 1 , \quad \theta(\bar{y}, \bar{x}) \geq 0 \quad (\bar{y}, \bar{x}) \in YX \quad (19)
\]

For the decomposable version (14), we rather use the Dantzig-Wolfe relaxation corresponding to the “easy component” treatment [3], which just corresponds to the partial DW reformulation whereby the \( X_i \) are expressed as convex combination of their extreme points, but \( Y \) is not:

\[
\max dy + \sum_{i \in N} \sum_{\bar{x}_i \in X_i} (c_i \bar{x}_i) \theta_i \quad (20)
\]

\[
Dy + \sum_{i \in N} \sum_{\bar{x}_i \in X_i} (C_i \bar{x}_i) \theta_i = b \quad (21)
\]

\[
\sum_{\bar{x}_i \in X_i} (A_i \bar{x}_i) \theta_i \leq y_i \quad i \in N \quad (22)
\]

\[
y \in Y \quad (10)
\]

\[
\sum_{\bar{x}_i \in X_i} \theta_i \leq 1 , \quad \theta_i \geq 0 \quad \bar{x}_i \in \bar{X}_i \quad (23)
\]

We remark that constraint (23) would ordinarily be an equality, but since \( 0 \in \bar{X}_i \), the variable \( \theta_0 \) acts as a slack and we can consider the constraint an inequality (ideally removing 0 from it). What we propose here is a modification of (20)–(23) that directly exploits
the specific quasi-separable structure, i.e., the constraints (8) defining the semi-continuous variables. This is obtained by just replacing (22)–(23) with

\[ \sum_{x_i \in X_i} \theta x_i = y_i \quad i \in N . \] (24)

That is, the only required modification is that the many extra linking constraints (8)/(22), that one has to relax if separability has to be attained, are replaced with much fewer linking constraints (24): exactly one for each \( i \in N \). It is clear that the bound remains the same. It is also immediate to see that pricing correspond to the solution of two obvious modifications to (15)/(16); hence, the subproblems cost remains the same, while the Master Problem cost decreases w.r.t. that of the standard (20)–(23), while retaining the same advantage in convergence speed w.r.t. (17)–(19) due to separability. Actually, convergence speed can be expected to be higher, since there are (far) less multipliers.

In the MND case, for instance, the standard (20)–(23) would have \( |A| \times |K| \) multipliers \( \mu \), whereas our newly proposed version only have \( |A| \) of them; for problems with many commodities, the difference can be significant.

3 Conclusion

We have proposed a modification of the Dantzig-Wolfe reformulation of MND corresponding to the Knapsack Relaxation. Our reformulation provides solution algorithms with the useful separable structure at a much lesser cost than the standard approach. Our computational results will provide a measure of how large the advantage is in practice. Besides its interest in the specific application, the approach can clearly be extended to other classes of problems with quasi-separable structure.

References


Routing a hybrid fleet of conventional and electric vehicles: the case of a French utility

Jorge E. Mendoza
Laboratoire d’Informatique Fondamentale et Appliquée de Tours (EA 6300)
Université de Tours, Tours, France

Alejandro Montoya
Departamento de Ingeniería de Producción
Universidad EAFIT, Medellín, Colombia

Christelle Guéret
Laboratoire Angevin d recherche en Ingénierie de Systèmes (EA 7315)
Université d’Angers, Angers, France

Juan G. Villegas
Departamento de Ingeniería Industrial
Universidad de Antioquia, Medellín, Colombia
Email: jorge.mendoza@univ-tours.fr

1 Introduction

Motivated by stricter environmental regulations, government incentives, branding opportunities, and potential reductions in operational costs, companies all around the world are renewing their fleets with electric vehicles (EVs). One of the main challenges faced by companies in the transition to clean fleets is the lack of fleet management tools that are able to handle hybrid fleets of “conventional” and electric vehicles (hereafter CVs and EVs). In this talk we present a study carried in collaboration with French electricity giant EDF to try to address this challenge. More precisely, we describe the technician routing and scheduling problem with CVs and EVs (TRSP-CEV) faced by ENEDIS (an EDF subsidiary) and present a decomposition-based parallel matheuristic to solve it. We show results on real data provided by the company and discuss managerial insight into the impact of the fleet composition (percentage of EVs vs. CVs) in the feasibility and cost of the solutions.
2 Problem description

The TRSP-CEV can be defined on a directed and complete graph $G = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. The set of nodes is defined as $\mathcal{N} = \{0\} \cup \mathcal{C} \cup \mathcal{S}$, where node 0 represents the depot, $\mathcal{C}$ is a set of nodes representing the customers, and $\mathcal{S}$ is a set of nodes representing the charging stations (CSs) where the electric vehicles can recharge their batteries. Each customer $i \in \mathcal{C}$ has a request demanding a skill $k_i$ from a set $\mathcal{K}$ and having a service time $p_i$ and a time window $[ec_i, lc_i]$, where $ec_i$ and $lc_i$ are the earliest and latest possible service start times. For simplicity, hereafter we use the terms customer and request interchangeably. Each CS $c \in \mathcal{C}$ has a parking cost (in €/min) representing the cost of the charging time at that station. The set of technicians is denoted as $\mathcal{T}$. Each technician $t \in \mathcal{T}$ has: a fixed daily cost $c_t$; a subset of skills $\mathcal{K}_t \subseteq \mathcal{K}$; a shift $[es_{st}, ls_{st}]$, where $es_{st}$ is the technician’s earliest possible departure time from the depot and $ls_{st}$ is the technician’s latest return time to the depot; a lunch break that must start at $el_t$ and end at $ll_t$; and an energy consumption factor $cf_t$ associated to the technician’s driving profile (e.g., sportive, normal, eco). To cover their routes, the technicians drive vehicles belonging to a fixed fleet composed by different types of CVs and EVs. The set of vehicle types is defined as $\mathcal{V} = \mathcal{V}_c \cup \mathcal{V}_e$, where $\mathcal{V}_c$ is the set of CV types and $\mathcal{V}_e$ is the set of EV types. For each vehicle type $v \in \mathcal{V}$ there is a travel cost $tc_v$ (in €/km) and a fixed and limited number of vehicles $m_v$. Vehicles of type $v \in \mathcal{V}_e$ additionally have: a fixed cost $gc_v$ for recharging the battery (in €)$^1$, a battery capacity $Q_v$ (in kWh), a set $\mathcal{S}_v \subseteq \mathcal{S}$ of compatible CSs, and a discrete and non-linear charging function $f_{vs}$ describing the relation between the vehicle’s charging time and state of charge (SoC) at station $s \in \mathcal{S}_v$. Depending on the context we refer to the SoC as the amount of remaining energy (in kWh) or as the percentage of remaining battery capacity. The charging function is defined as $f_{vs} = \{a_b | b \in \{0, 1, \ldots, 100\}\}$ were $a_b$ is the time needed to take the SoC from 0 to $b$ percent of $Q$. Finally, set $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$ denotes the set of arcs connecting nodes in $\mathcal{N}$. Each arc $(i, j) \in \mathcal{A}$ has three associated nonnegative values: a travel time $tt_{ij}$, a distance $d_{ij}$, and a nominal energy consumption $e_{ijv}$ for each type of EV $v \in \mathcal{V}_e$.

In the TRSP-CEV the objective is to find a set of routes of minimum total cost. The latter is defined as the sum of i) travel costs, ii) fixed charging costs, iii) parking costs, and iv) technician utilization costs. The planned set of routes must satisfy the following constraints: each request is served exactly once within its time window by a technician with the required skill; the level of the battery when the EVs arrive at any vertex is nonnegative; the EVs only charge at compatible CSs; each technician works only during his or her shift; each technician takes the lunch break at the pre-defined times; the number of vehicles of type $v \in \mathcal{V}$ used in the plan is less or equal than $m_v$; and each route starts and ends at the depot.

$^1$This cost accounts for the long-term battery degradation cost
Algorithm 1: Parallel matheuristic: general structure

1: function PARALLELMATHEURISTIC(G, T, V)
2: \( \mathcal{TP} \leftarrow \text{groupTechnicians}(T) \)
3: \( \mathcal{TV} \leftarrow \text{buildAssignments}(\mathcal{TP}, V) \)
4: \( \Omega \leftarrow \emptyset \)
5: parallel for each \( tv \in \mathcal{TV} \)
6: \( \Omega_{tv} \leftarrow \text{GRASP}(tv, G) \)
7: \( \Omega \leftarrow \Omega \cup \Omega_{tv} \)
8: end for
9: \( \sigma \leftarrow \text{setCovering}(G, \Omega, V, \mathcal{TV}) \)
10: return \( \sigma \)
11: end function

3 Parallel matheuristic

Algorithm 1 describes the general structure of our parallel matheuristic (hereafter referred to as PMa). The algorithm starts by calling procedure \text{groupTechnicians}(T) – line 2. This procedure groups the technicians sharing the same characteristics (i.e., skills, fixed utilization cost, energy consumption factor, shift, and lunch break) and generates the set \( \mathcal{TP} \) of technician profiles. Then, the algorithm invokes procedure \text{buildAssignments}(\mathcal{TP}, V) – line 3. The latter builds the set \( \mathcal{TV} \) containing all possible technician profile-vehicle type assignments. Note that \( |\mathcal{TV}| = |\mathcal{TP}| \times |V_c| + |\mathcal{TP}| \times |V_e| \).

Then, the algorithm starts the parallel phase – lines 5 to 8. For each assignment \( tv \in \mathcal{TV} \) the algorithm solves, on a dedicated thread, a vehicle routing problem with time windows and lunch breaks (VRP-TWLB). Let \( p(tv) \in \mathcal{TP} \) and \( v(tv) \in V \) be the technician profile and the type of vehicle involved in assignment \( tv \). In the VRP-TWLB for assignment \( tv \) we assume that i) the fleet is unlimited and composed only of vehicles of type \( v(tv) \) and that we have an unlimited number of technicians with profile \( p(tv) \). If \( v(tv) \) is an EV, then the resulting problem is an electric VRP-TWLB. To solve the \( |\mathcal{TP}| \times |V_c| \) VRPs-TWLB and the \( |\mathcal{TP}| \times |V_e| \) eVRPs-TWLB our approach relies on a GRASP (line 6). The GRASP slightly varies depending on the type of problem being solved (VRP-TWLB or eVRP-TWLB). Figure 1 depicts the components embedded in the two versions. The GRASP returns a set \( \Omega_{tv} \) containing all the routes found in the local optima reached during the algorithm’s execution. The routes in \( \Omega_{tv} \) join the long term memory structure \( \Omega \) (line 7). After completing the parallel phase, the algorithm calls procedure \text{setCovering}(G, \Omega, V, \mathcal{TV}) – line 9 –, which solves an extended set covering formulation over \( \Omega \) to find a feasible TRSP-CEV solution. It is worth noting that it is only at this point that we take into account the constraints on the number of technicians and vehicles. Full details on the GRASP components, specially those used to solve the eVRPs-TWLB, and the parallel implementation will be discussed in the talk.
4 Computational experiments

We ran experiments on two sets of instances. The first set is made up of 24 “real-world” TRSP instances (10 small with proven optima + 14 large) built using data provided by ENEDIS. For the 10 small instances our PMa was able to find the optimal solutions. For the remaining 14 instances PMa reported average improvements of 6.5% with respect to the solutions delivered by the software currently used at ENEDIS. We adapted the instances to the TRSP-CEV by setting the portion of EVs in the fleet to \{20\%, 40\%, 60\%, 80\%, 100\%\}.

We ran experiments to evaluate the impact of the fleet composition on the total cost and feasibility of the solutions. We found that for urban settings, transitioning to a 100% electric fleet is feasible and may generate considerable savings in cost and emissions. On the other hand, to obtain feasible solutions for all rural instances, only 20\% of the fleet can be replaced by EVs. The second instance set consists of the 276 instances (108 small with proven optima + 168 large) proposed in [1] for the closely-related electric fleet size and mix vehicle routing problem with time windows and recharging stations. Although our method was not tailored for this problem, it was able to deliver competitive performances with respect to the state-of-the-art Adaptive Large Neighborhood Search (ALNS) proposed in [1]. On the small instances our PMa found 81/108 optimal solutions and reported better avg. gaps (0.32\% vs 0.55\%) and execution times (0.06min vs. 0.32min) than ALNS. On the large instances, our method unveiled 61 new best known solutions but reported larger avg. gaps than ALNS (2.20\% vs. 1.18\%) with comparable execution times.

References

Electric Vehicle Routing Problem with Time
Dependent Waiting Times at Recharging Stations

Merve Keskin
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey
CIRRELT, Montréal, Canada

Gilbert Laporte
CIRRELT and Department of Management Sciences, HEC, Montréal, Canada

Bülent Çatay
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey
Email: catay@sabanciuniv.edu

1 Introduction and Problem Description

The Electric Vehicle Routing Problem with Time Windows (EVRPTW) was introduced by [1] as an extension to the Green Vehicle Routing Problem of [2]. It is a variant of the Vehicle Routing Problem with Time Windows where a fleet of electric vehicles (EVs) is used instead of internal combustion engine vehicles. Since the EV has a limited driving range it may need to have its battery recharged at a station en route. In [1], it is assumed that the battery is fully recharged at any state of charge (SoC) and the duration is linearly proportional to the amount of energy transferred. The full recharge restriction was later relaxed in [3]. Furthermore, it is shown that the recharge duration is a concave function of the energy transferred and charging slows down when the energy level reaches 85% of the battery capacity [4-6].

The existing literature assumes that recharging at the station starts as soon as the EV arrives at the station. However, in practice the number of chargers in a station is limited and they may not be available at the time of arrival. So, the EV may have to queue before recharging and this waiting time may affect the routing decisions. The waiting time may vary depending on the location of the station and the time of the visit. Some variations in the waiting time are difficult to predict. For instance, if a charger is out of order, the EVs should queue in at another charger, if available, or go to a nearby station. On the other hand, some waiting time can be foreseen, for example when there are more vehicles on the roads during the rush hours, which translates in increased demand for recharging. In this study, we extend the EVRPTW by considering waiting times at the
recharging stations. We assume an M/M/1 queueing system and use expected waiting times to predict the queue lengths. We also use a nonlinear charging function and allow late arrivals at the customers and the depot with penalties. Moreover, we assume that the battery is operated between 10% and 90% of its capacity since it degrades faster beyond these limits [5]. Recently, [7] has addressed a similar environment where the stations have limited number of chargers (1, 2 or 3) and an EV may need to wait for service if the chargers are busy recharging other EVs in the fleet. In this problem, the use of the chargers depends on the routing and charging decisions made, whereas in our setting the queue lengths are independent from our fleet, such as is the case at public stations.

In this context, we split the planning horizon (usually a day) into a predetermined number of time intervals such as morning, afternoon, evening, and night, and the average queue lengths at the stations vary according to the arrival time. We assume different EV arrival rates depending on the time of the day, e.g. the stations are less busy in the morning since the EVs depart with a full battery whereas the energy on the battery is usually consumed in the afternoon or evening, hence the EVs need recharging to continue their routes. The routing decisions are then made according to these time-dependent waiting times at the recharging stations. The objective is to minimize a total cost function which includes the energy cost, cost of vehicles, driver wages, and penalties associated with time window violations at the customers and the depot.

2 Solution Methodology

We assume that each recharging station is equipped with a single charger. The arrivals are Poisson with mean $\lambda$, and the service times follow an exponential distribution with parameter $\mu$. We apply a first come first served queue discipline where the first in first out property holds, i.e. the vehicles leave the station in the order they have arrived.

Our solution approach is a matheuristic which integrates the Adaptive Large Neighborhood Search (ALNS) method with an exact method. ALNS is a metaheuristic framework that iteratively destroys a solution and then repairs it using a greedy heuristic [8]. It has been successfully applied for solving various VRP variants in the recent literature. It uses several operators to destroy and repair the solution which are selected adaptively based on their past performances in improving the solution. In this study, we use the customer removal and insertion operators introduced in [3]. The repaired solution is accepted according to a simulated annealing criterion which allows the acceptance of worse solutions with a certain probability. Furthermore, we optimize the charging station selection and recharge quantity decisions of the vehicles by solving a mixed integer program using a commercial solver every after a predetermined number of iterations $\Delta$. Basically, the best solution obtained in the last $\Delta$ iterations is further enhanced by optimizing charging de-
cisions while preserving the sequence of the customers. To solve this fixed-route problem fast, we formulate an effective mathematical model.

3 Experimental Results

To validate the performance of the proposed approach, we perform experiments using 36 small instances generated by [1]. We compare our results with those obtained with CPLEX. The service rate $\mu$ is calculated as explained in Section 1. We assume $\lambda = \mu/3$ for the least busy time interval and set the arrival rate $\lambda_t$ in time interval $t$ randomly by multiplying $\lambda$ with a positive scalar. All stations are assumed to have the same arrival rates for each time interval. Finally, the average waiting time at a station in time interval $t$ is calculated as $w_t = \lambda_t/\mu(\mu - \lambda_t)$. $\Delta$ is set to 200 and matheuristic terminates after 10,000 iterations. All experiments are conducted on an Intel Xeon E5 3.30 GHz processor and 32 GB of RAM. The mathematical model and the matheuristic are coded in Java and solved by CPLEX 12.6.2 with default settings.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Cost</th>
<th>Time</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>C101C10</td>
<td>3562.18</td>
<td>2233</td>
<td>3562.18</td>
<td>10</td>
</tr>
<tr>
<td>C104C10</td>
<td>2348.00</td>
<td>7200</td>
<td>2347.63</td>
<td>31</td>
</tr>
<tr>
<td>C202C10</td>
<td>3904.82</td>
<td>7200</td>
<td>3904.82</td>
<td>38</td>
</tr>
<tr>
<td>C205C10</td>
<td>3417.17</td>
<td>160</td>
<td>3417.17</td>
<td>25</td>
</tr>
<tr>
<td>R102C10</td>
<td>1611.83</td>
<td>7200</td>
<td>1608.37</td>
<td>12</td>
</tr>
<tr>
<td>R103C10</td>
<td>1271.70</td>
<td>7200</td>
<td>1271.70</td>
<td>11</td>
</tr>
<tr>
<td>R201C10</td>
<td>1289.27</td>
<td>6272</td>
<td>1289.27</td>
<td>48</td>
</tr>
<tr>
<td>R203C10</td>
<td>1319.45</td>
<td>7200</td>
<td>1319.45</td>
<td>55</td>
</tr>
<tr>
<td>RC102C10</td>
<td>2211.27</td>
<td>7200</td>
<td>2211.27</td>
<td>9</td>
</tr>
<tr>
<td>RC108C10</td>
<td>2073.69</td>
<td>7200</td>
<td>2073.69</td>
<td>9</td>
</tr>
<tr>
<td>RC201C10</td>
<td>1605.90</td>
<td>7200</td>
<td>1605.90</td>
<td>29</td>
</tr>
<tr>
<td>RC205C10</td>
<td>1534.29</td>
<td>7200</td>
<td>1534.29</td>
<td>26</td>
</tr>
</tbody>
</table>

The results are reported in Table 1. The computation times are given in seconds. We set the time limit for CPLEX to 7200 seconds. So, if the computation time of CPLEX is less than this limit, the solution is optimal; otherwise, it is the best upper bound found. The matheuristic solutions are the best of 10 runs whereas the computational times are average of 10 runs. Since both CPLEX and our method found the optimal solution in all 5-customer instances, we did not include them in the table. On average, the optimal solutions are attained in 206.8 and 10.7 seconds by CPLEX and the matheuristic, respectively. In 10-customer instances, the matheuristic provides better solutions in two instances which are highlighted in bold. In the remaining 10 instances, CPLEX and the matheuristic find the same solutions but CPLEX run times are significantly more and reach the time limit.
in most of the cases. In 15-customer instances the matheuristic achieves better solutions in 7 out of 12 instances. Furthermore, the run time is significantly better than CPLEX for all cases. Overall, we can claim that the proposed method outperforms CPLEX both in solution quality and computational time.

In conclusion, the proposed approach is effective in solving all small size instances. Our future work will concentrate on testing its performance on large instances under various scenarios of queue characteristics.

References


Modeling and solving the electric vehicle routing problem with nonlinear charging functions and capacitated charging stations

Aurélien Froger, Jorge E. Mendoza
Université François-Rabelais de Tours,
LI (EA 6300), ROOT (ERL CNRS 6305), Tours, France

Ola Jabali
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milano, Italy

Gilbert Laporte
Canada Research Chair in Distribution Management
HEC Montréal, Montréal, Canada H3T 2A7
Email: aurelien.froger@univ-tours.fr

1 Introduction

Electric vehicle routing problems (E-VRPs) are receiving growing attention from the operations research community. Electric vehicles differ substantially from internal combustion engine vehicles (ICEVs), the main difference lying in their limited autonomy, which can be recuperated at charging stations (CSs). These are much more scarce than conventional refueling stations for ICEVs, which means that EVs often may need to perform en route detours to reach a CS. The latter is specially true in the context of mid-haul or long-haul routing. Most of the research on E-VRPs implicitly assumes that the charging infrastructure is owned by the EV operator, which is plausible for large transportation companies. Modeling the charging functions is a focal point of E-VRPs. In practice the charging function of an EV is non-linear and the charging quantities are decision variables. However, most of the E-VRP literature relies on models that assume linear charging functions and/or full charging policies. In an effort to overcome these limitations, Montoya et al. [2] recently introduced the E-VRP with nonlinear charging function. In their problem the charging functions are more realistically approximated using piecewise linear functions and partial charging is allowed.

Another common assumption in the E-VRP literature is that the CSs are uncapaci-
tated, that is, they are able to simultaneously handle an unlimited number of EVs. In practice, each CS has a limited number of chargers. Needless to say, neglecting the CS capacity constraints may lead to poor decisions in practice. For instance we ran a feasibility test on the 120 BKS for the E-VRP-NL reported in Montoya et al. [2] while limiting the number of chargers per CS to 1, 2, 3, and 4. According to our results, nearly 50% of the these BKS become infeasible when only 1 charger is available. This figure drops to 11% and 2% for the cases with 2 and 3 chargers, respectively. On the other hand, when 4 chargers are available, all solutions remain feasible.

In this research we focus on the E-VRP-NL and we extend it to consider capacitated CSs. We call the resulting problem the E-VRP-NL with capacitated CSs (E-VRP-NL-C).

2 Problem statement

We define the E-VRP-NL-C as follows. An unlimited and homogeneous fleet of electric vehicles (EVs) need to serve a set of customers. At the start of the planning horizon, all EVs are located at a single depot which they leave fully charged. Traveling from a location to another location incurs a driving time and an energy consumption (the triangular inequality holds for both). Each CS has a charging technology (e.g., slow, moderate, fast) associated with a nonlinear charging process that is approximated with a piecewise linear function. Each CS has also a capacity, given by the number of available chargers. Feasible solutions to the E-VRP-NL-C satisfy the following conditions: 1) each customer is served exactly once by a single EV, 2) each route starts and ends at the depot, 3) each route satisfies a given maximum-duration limit, 4) each route is energy feasible (i.e., the state of charge of an EV upon arriving at a location or departing from it lies between 0 and the battery capacity), and 5) the number of EVs simultaneously charging at each CS does not exceeds the number of available chargers. The objective of the E-VRP-NL-C is to minimize the total time needed to serve all customers. This takes into account driving, service, and charging times. Due to the limited availability of CSs, it also includes the waiting times that may occur at CSs whenever an EV queues for a charger.

3 Mixed integer linear programming formulations

The E-VRP-NL-C is a combined routing (the EVs visiting customers) + scheduling (the charging operations) problem. In addition to a classical CS replication-based formulation (see [1],[3],[4]), we present a model that avoids replicating the charging stations nodes. To extend previous formulations of the E-VRP-NL to include the CS capacity constraints, we borrowed some ideas from the Resource Constrained Scheduling Problem (RCPSP) literature. There is, however, a major difference between these problems and our CS scheduling problem: in the latter i) the duration of each task (i.e., charging operation)
and ii) the number of tasks executed by each resource (i.e., charging station) are decision variables. To the best of our knowledge this case has never been addressed in the scheduling literature before. We propose two formulations of the capacity constraints, a flow-based one and an event-based one. As expected, solving our models using a commercial solver allows us to address only small size instances. Therefore, we developed a two-stage matheuristic for the problem.

4 A two-stage solution matheuristic

To tackle the E-VRP-NL-C we propose a route-first assemble-second matheuristic. In this two-stage method, we first build a diverse pool of routes, and then we assemble solutions by selecting a subset of routes from the pool.

In the routing stage of our method, we build a pool $\Omega$ of high-quality routes while relaxing the capacity constraints. To generate the pool of routes, we use a local search-based metaheuristic that combines components from the routing literature and components specifically designed to consider charging decisions. Specifically, the search uses classical operators focusing on sequencing decisions such as two-opt and relocate. Evaluating a move for the E-VRP-NL-C raises several challenges. Indeed, altering the sequence of customers in a route (i.e., removing or inserting one or more customers) can make the current charging decisions infeasible or suboptimal. Preliminary computational experiments showed that decoupling the charging decisions from the evaluation of sequencing moves can have undesirable effects. We investigated the impact of exactly evaluating each sequencing move on the efficiency of the method. We tested different strategies. First, we optimally evaluate each move by solving a constrained shortest path problem where the objective is to minimize the path duration and the state of charge of the EV acts as a resource constraint. For this purpose, we use a label-setting algorithm. To reduce the computational time, we examine a heuristic version of this algorithm. For a comparison, we also consider heuristic evaluation procedures including one that consists in disregarding the detour to CSs and the charging times. To improve the efficiency of the exploration of a neighborhood, we adopt two strategies. First, we use short-term and long-term memory structures to avoid evaluating twice the same routes or the same move. Second, we restrict the arcs that can be involved in a sequencing move.

In the assembly stage of our method, we select routes from the pool $\Omega$ to build a solution to the problem. Since route-first assemble-second approaches have been mostly applied to problems without route coupling constraints, the assembly phase traditionally consists in solving a set partitioning (SP) model over the pool of routes. Due to the CS capacity constraints, the assembly phase on the E-VRP-NL-C requires a more elaborate treatment. We therefore solve this stage using a Benders’ like decomposition method.
Specifically, we decompose this assembling problem into a route selection master problem and a CS capacity management sub-problem. The master problem consists in selecting a set of routes such that every customer is covered exactly by one route. Every selection of routes (output of the route selection problem) yields a set of charging operations; each operation being defined by a CS, a starting time, and a recharge amount. The sub-problem checks if the CS capacity constraints can be met. We investigate three different versions of the CS capacity management problem ranging from a simple check of the capacity constraints to the introduction of waiting times to the revision of the charging amounts in the selected routes. To efficiently solve the problem while exploiting this decomposition, we adopt the following approach implemented on top of a commercial solver. We solve the SP model related to the route selection problem using a branch-and-bound algorithm. At each integer node of the branch-and-bound tree, the corresponding solution is sent to the CS capacity management problem. We discard infeasible solutions or solutions for which the objective is underestimated (decisions in the sub-problem may impact the time-based objective of the E-VRP-NL-C) using cuts.

The results suggest that our method performs well on a set of instances adapted from the literature. Results show that using more complex strategies to solve bottleneck issues at CSs do not necessarily increase the computational burden but improve the quality of the solutions. Furthermore, the algorithm finds optimal solutions for several instances.

References


1 Introduction

The recent literature on routing problems is evolving to the study of more and more complex problems. This complexity stems from different sources, among which integration and flexibility are the most investigated. By integration, we mean to include broader parts of the decision systems, and not only the one focused on the pure stand-alone routing. Classical examples are the Inventory Routing Problem (IRP) (see [2, 3, 4]), location routing problems (see [7]) and two-echelon vehicle routing problems (see [5, 6]).

Flexibility is related to the possibility of relaxing some constraints in order to save costs. For example, in a distribution problem where customers requests have to be satisfied within a planning horizon, one may achieve cost savings if flexibility in the due dates is allowed, as shown in [1].

In this paper we study a routing problem where both integration and flexibility are considered. In particular, we study a problem coming from a real application where a supplier has to build a distribution plan to serve the customers through a two-layer distribution network. A single commodity is produced at a production plant, or stocked at the depot, and is distributed from there to a set of distribution centers (DCs). Then, the commodity is delivered to customers from the DCs. A planning horizon is considered...
which is discretized in periods, typically days. The supplier has the possibility to choose among the available DCs on a daily basis. In fact, we consider the DCs as the rented space in physical facilities shared with other companies and managed by a third party. Daily customers requests are known and dynamic. Moreover, each order has a due date, which represents the latest delivery date. Each order has to be entirely fulfilled in one delivery. A penalty is related to an unmet demand, i.e., to orders which are not satisfied by the delivery due date. Products are shipped from the depot to the selected DCs via full truckloads, and from DCs to customers via milk runs. The supplier has to take four simultaneous decisions: which DCs to use in each period, when to satisfy the orders of customers, from which of the selected DCs to ship to the customers, and how to create vehicle routes from the selected DCs to the customers.

We call this problem the Flexible Two-Echelon Location Routing Problem (F-2E-LRP), in which the objective is to minimize the total costs consisting of the sum of the shipping costs from the depot to the DCs, the delivery cost from the DCs to the customers, the renting cost of DCs, and the penalty cost for the unmet demand. The F-2E-LRP merges integration issues related to the decision of which DCs to rent, and flexibility issues coming from two sources:

- the possibility of selecting amongst the available DCs on a daily basis;
- the possibility of selecting the day when customer orders are satisfied, provided that either the due date is respected or a penalty is paid.

The name F-2E-LRP is related to the fact that the problem can be seen as a generalization of the Two-Echelon Location Routing Problem (2E-LRP) (see [5] for a survey on two-echelon routing problems) where multiple periods are considered.

2 Problem description

Let $T$ indicate the discretized planning horizon, typically days, of length $T$. Let $C$ represent the set of customers and $D$ the set of potential DCs, each with a single vehicle available for the distribution (if the DC is selected). Each customer $c \in C$ has a known demand $d^t_c$ for each period $t \in T$. Once the customer places an order, the demand could be fulfilled from any of the selected DCs within a due date $r$, which is fixed for all customers in all periods. Late orders are not lost but any demand fulfilled after the due date is subject to a penalty cost $p$ per period. Although the demand is known a priori, no demand can be satisfied in advance. Let $f^i$ be the daily fee for DC $i$. Each selected DC is rented for one day. If the same DC is selected for two or more consecutive days, it can hold inventory from one day to another up to a capacity $C_i$, $i \in D$. When a DC is not rented in a given
day, any remaining previous inventory is lost. We assume that the fee $f_i$ covers all the handling costs of products kept in the DCs, hence, no inventory holding cost is due.

All products are stored in DCs before being sent to the customers. Each DC possesses a vehicle with capacity $Q$. The vehicle may visit several customers per day in a single trip, starting and ending at the same DC. No partial shipment of an order is possible. Different orders from the same customer in different periods may be either bundled together or shipped separately from the same or different DCs, and/or in different periods, but one order from a customer in a period cannot be split in different periods.

Transportation costs are accounted as follows. Each shipment from the depot to DC $i$ costs $s_i$ and has a transportation capacity $W$. Vehicle routes from each DC to any of the customers incur a cost which is based on the distance traveled. A distance matrix $c_{ij}$ is known, $i, j \in \mathcal{C} \cup \mathcal{D}$, where $c_{ij}$ is the cost of traveling from location $i$ to location $j$. No transshipment between DCs is allowed, i.e., goods stored at a DC are distributed to customers only.

The objective of the F-2E-LRP is to minimize the total cost of distribution, including the DC rental fees, the transportation costs from the depot to the DCs and from DCs to the final customers as well as the late delivery penalty.

Solution approach and computational results

We propose a mathematical formulation the F-2E-LRP along with different classes of valid inequalities. The problem formulation is a MIP model with binary variables identifying, for each time period, which DCs are open, which customer requests are served and the arcs traversed by the vehicles. In addition, there are continuous variables related to the flow of goods to DCs and from DCs to customers. Constraints include inventory balance, routing constraints and capacity constraints. The resulting model combines the IRP structure with the choice related to which DCs are used in each period.

Valid inequalities are used to enhance the formulation by setting links between routing variables and visit variables and by limiting the set of customers which may be served in each time period on the basis of the aggregated demand.

All valid inequalities are in polynomial number so they are added to the problem formulation. The resulting formulation is solved through the commercial solver CPLEX 12.7.0 and IBM Concert Technology.

We run a large set of experiments on randomly generated instances to show the value of flexibility, both in terms of due dates, in terms of network design, and on their combined effect. Instances have up to 25 customers, 6 time periods and 3 DCs. We generated a total number of 1,350 instances.

The results highlight the cost saving advantages of both types of flexibility. In particu-
lar, concerning the flexibility of network design, we compared the case in which we impose the DCs selected in the first period to remain unchanged throughout the planning horizon (fixed network design) versus the case where the model re-evaluates the decision on which of the DCs should be rented in each period (flexible network design). The results show that, on average, we can save 62% of the total cost by allowing a flexible network design. Concerning the flexibility coming from due dates, we compare the cases with \( r = 0 \), \( r = 1 \) and \( r = 2 \). The results show that while serving the demand the next day rather than on the same day reduces the cost by to 22%, changing from next day delivery to delivery within two days leads to 10% additional savings. In addition, savings are larger for the flexible network design case with respect to fixed network design. Finally, we show that the combined effect of the two kinds of flexibility leads to a saving in total cost of up to almost 35%.

Concluding, the results show that the flexibility in the design of the service network provide a substantial economic benefit, i.e., the possibility of modifying the location of DCs from period to period leads to remarkable savings in the overall cost. This possibility may nowadays be provided by third-party logistics companies that offer storage spaces to their customers in warehousing facilities that are geographically spread over different areas and renting contracts and fees are flexible enough to allow for short periods of use.

References


Models and solution algorithms for Location-Arc Routing Problems

Elena Fernández
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya-BcnTech, Spain

Gilbert Laporte
Canada Research Chair in Distribution Management
HEC Montréal, Montreal, Canada

Jessica Rodríguez-Pereira
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya-BcnTech, Spain
Email: jessica.rodriguez@upc.edu

1 Introduction

We model and solve exactly several families of Location-Arc Routing Problems (LARPs). These problems extend the Multi-Depot Rural Postman Problem [1, 3] to the case where the depots are not fixed. We consider min-cost objectives aiming at minimizing the overall routing costs, as well as min-max objectives aiming at minimizing the makespan. While some of the models assume that there are no capacity limitations, we also study problems that include a cardinality constraint on the number of users that can be served from an open facility. Some of the models ignore facilities set-up costs but include a limitation on the maximum number of facilities to be located, whereas other models do not limit the number of open facilities but include facilities set-up costs in the objective function.

We present two types of formulations that use binary variables only. The first class uses three-index variables that link routes with open facilities. Polyhedral analysis indicates that the main families of constraints are facet defining. All models can be handled with this type of formulation. For min-cost models without capacity constraints the information of all the routes can be aggregated. This leads to two-index variables, associated with the edges traversed by the routes, which do not explicitly link them to the facilities from which the routes operate. This approach reduces the number of required variables at the expenses of presenting some additional difficulties. For the two-index formulation we present a new set of constraints guaranteeing that the routes return to their original depot.
2 Location-Arc Routing Problems

We consider LARPs defined on an undirected connected graph \( G = (V, E) \), where \( V \) is the vertex set, \( |V| = n \), and \( E \) is the edge set, with \( |E| = m \). The set \( D \subset V \) denotes a set of potential locations where facilities may be established. A given set \( R \subset E \) of edges must be traversed (served), which are referred to as required edges. The connected components induced by the required edges are referred to as required components and are denoted by \( C_k = (V_k, R_k), k \in K \). Hence, \( R = \bigcup_{k \in K} R_k \). Let also \( V_R = \bigcup_{k \in K} V_k \). We assume that no component has more than one candidate location, although it is possible that a component contains none, i.e. \( |V_k \cap D| \leq 1 \) for all \( k \in K \). There is a traversal cost \( c_e \geq 0 \) associated with each edge \( e \in E \), and a value \( f_d \geq 0 \), associated with each potential location \( d \in D \), which indicates the set-up cost of opening a facility at \( d \). Let \( p \) be an upper bound on the number of facilities to be located.

Feasible LARP solutions consist of a subset of open facilities \( D^* \subset D \), together with a set of non-empty routes, at least one for each selected facility, that serve all the required edges. We study the following problems:

**Definition 2.1**

- **The MC-\( p \)-LARP is to determine a feasible solution with at most \( p \) open facilities, i.e. \( |D^*| \leq p \), that minimizes the sum of the routing costs.**

- **The MM-\( p \)-LARP is to determine feasible solution with at most \( p \) open facilities, i.e. \( |D^*| \leq p \), that minimizes the makespan.**

- **The MC-LARP is to determine a feasible solution that minimizes the sum of the set-up costs of the selected facilities, plus the routing costs.**

We also consider capacitated versions of each of the above defined problems, where we assume that each required edge has a unit demand, and for each potential facility there is a constraint on the maximum demand that it can serve if it is opened. Since we consider unit demands, these capacitated versions reduce to cardinality constraints on the maximum number of required edges served by each facility.

We prove optimality conditions that indicate that no edge is traversed more than twice in an optimal solution. Moreover, the only edges that can be traversed twice in an optimal solution to the MC-\( p \)-LARP and the MM-\( p \)-LARP are the required edges plus the edges of \( T_C \), the minimum cost spanning tree on the graph induced by the components \( C_k, k \in K \). For the MC-LARP, any edge connecting two components can be used twice in an optimal solution. In each case, we denote by \( E^y \) the set of such edges.
All formulations use a set of binary decision variables $z_d$, $d \in D$, to identify the facilities that are opened, as well as two sets of binary variables, to represent the first and second traversal of edges in the routes. When three-index variables are used, for each $d \in D$, $x^d_e$ indicates whether or not edge $e \in E$ is traversed by a route from facility $d$, and $y^d_e$ whether or not edge $e \in E^y$ is traversed twice in the route from facility $d$. When two-index variables are used, the facility index is dropped, so we use variables $x_e$ and $y_e$ to respectively identify the first and second traversal of edges, without making explicit the facility associated with the route that traverses them. The main blocks of constraints impose the following conditions:

- connectivity of each route to its facility.
- parity (even degree) of every subset of vertices, via co-circuit inequalities [3].
- all required edges are served.
- no edge is traversed for a second time unless it has also been traversed for a first time.
- no edge is traversed by the route of a facility that has not been opened.

The formulations with two-index variables include, in addition, a set of constraints to guarantee that the routes return to their departing facility.

Since the families of connectivity and parity constraints are of exponential size in the number of vertices, exact and heuristic separation procedures have been developed for such families. These procedures have been integrated in exact branch-and-cut algorithms for each model and formulation.

Extensive computational experiments have been run on a series of benchmark instances adapted from well-known instances from the literature [1, 3, 2]. Table 1 summarizes the characteristics of the considered instances as well as the obtained results with the with three-index and two-index variables formulations for the MC-p-LARP, denoted 3IF and 2IF, respectively. These results clearly indicate the superiority of 2IF over 3IF. As can be seen, instances with up to 700 vertices could be solved to optimality with 2IF.

The results of computational experiments with three-index formulations for the models with a makesman objective and for the capacitated versions of the min-cost problems highlight the difficulty of this kind of problems. Despite their difficulty instances with up to 100 vertices were optimally solved both for the makesman objective and for the capacitated versions of the min-cost models.
Table 1: Summary of computational results for the MC-p-LARP

| # inst | |V| |E| |R| |3IF |2IF |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|        |       | %Opt | Gap(%) | CPU(s) | %Opt | Gap(%) | CPU(s) |
| D16    | 2     | 90–102| 143–159| 88–99 | 9/9  | 0 | 0.10 | 9/9  | 0 | 0.02 |
| D36    | 24    | 7–50  | 10–183 | 4–78  | 9/9  | 0 | 17.35 | 9/9  | 0 | 0.14 |
| D64    | 9     | 8–16  | 12–30  | 3–16  | 9/9  | 0 | 519.19 | 9/9  | 0 | 0.48 |
| D100   | 9     | 25–36 | 52–71  | 10–38 | 2/9  | 2.39 | 12361.85 | 9/9  | 0 | 12.41 |
| G16    | 9     | 40–63 | 92–120 | 27–75 | 9/9  | 0 | 0.26 | 9/9  | 0 | 0.03 |
| G36    | 9     | 76–100| 161–197| 50–121| 9/9  | 0 | 10.08 | 9/9  | 0 | 0.10 |
| G64    | 9     | 11–16 | 15–24  | 3–13  | 8/9  | 0.29 | 2031.50 | 9/9  | 0 | 0.33 |
| G100   | 9     | 22–36 | 34–60  | 11–35 | 2/9  | 24.96 | 12846.15 | 9/9  | 0 | 0.95 |
| R20    | 9     | 45–63 | 74–110 | 24–68 | 5/5  | 0 | 0.27 | 5/5  | 0 | 0.05 |
| R30    | 9     | 69–100| 121–180| 41–113| 5/5  | 0 | 1.77 | 5/5  | 0 | 0.08 |
| R40    | 5     | 13–15 | 24–72  | 4–7   | 5/5  | 0 | 44.01 | 5/5  | 0 | 0.30 |
| R50    | 5     | 15–23 | 28–99  | 7–11  | 5/5  | 0 | 31.32 | 5/5  | 0 | 0.21 |
| P      | 5     | 24–32 | 58–161 | 8–18  | 24/24| 0 | 11.19 | 24/24| 0 | 0.20 |
| ALB    | 5     | 23–39 | 82–169 | 13–20 | 2/2  | 0 | 4739.15 | 2/2  | 0 | 2.38 |
| ALB2   | 15    | 78–114| 133–172| 44–122| -    | - | - | 15/15| 0 | 17.23 |
| GRP    | 10    | 77–113| 138–171| 52–126| -    | - | - | 10/10| 0 | 59.83 |
| MAD    | 15    | 149–195| 274–318| 86–238| -    | - | - | 15/15| 0 | 251.52 |
| U500   | 7     | 298–493| 597–1403| 206–671| -    | - | - | 3/7  | 0.58 | 60832.93 |
| U700   | 8     | 452–744| 915–2089| 321–1003| -    | - | - | 2/8  | 18.04 | 70099.98 |

Acknowledgements

This research was partially supported by the Spanish Ministry of Economy and Competitiveness through grants EEBB-I-16-10670, BES-2013-063633, MTM2015-63779-R and MDM-2014-044 (MINECO/FEDER), and by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged.

References


Impact of Congestion Pricing Schemes on Costs and Emissions of Commercial Fleets in Urban Areas

Shu Zhang
School of Economics and Business Administration,
Chongqing University, Chongqing, China

Ann Melissa Campbell
Department of Management Sciences,
The University of Iowa, Iowa City, USA

Jan Fabian Ehmke
Management Science Group,
Otto-von-Guericke Universität Magdeburg, Magdeburg, Germany
Email: zhangshu@cqu.edu.cn

1 Introduction

Municipalities across the world have become concerned about the level of emissions in their city centers. As urbanization increases, they have become aware of the negative impacts of road-based transportation. Transportation causes traffic congestion, air pollution and can create health problems for the citizens of metropolitan areas. As a result, several cities have introduced tolling schemes to discourage vehicles from entering the inner city to reduce congestion and pollution, especially during peak hours. In this study, we are interested in the impact of congestion charges. Specifically, we are interested in how different kinds of congestion charge schemes perform from both the municipality’s as well as the commercial fleet operator’s perspective. While the role of the municipality is to ensure a reduction of emissions, operators of commercial fleets are mainly interested in keeping their costs as low as possible. Operators of commercial fleets have to consider congestion charges in their route planning, because they can increase the cost of routing significantly. This may lead to situations where certain types of congestion charge schemes can even lead to an unintended increase of emissions in the city center. Furthermore, some types of congestion charge schemes may lead to an increase of commercial traffic in the vicinity of the city center, counteracting the good intentions of the policy.
Since current congestion charge schemes vary in terms of fees, times, geographical area, etc., it is challenging to create vehicle routing instances and techniques to compare their effectiveness. Hence, little research has been done to examine the impact of congestion charge schemes on the emissions that occur in city centers systematically. The main exception is [1], who look at how to minimize total costs when routing vehicles to customers in a city with a fixed daily charge for entering the city center. However, they only consider one congestion charging type and have not analyze the problem from the policy perspective. In this study, we study a vehicle routing problem considering different tolling schemes for several city types and congestion zone sizes. We compare how they impact a company’s total costs, fuel usage (which drives emissions), and the routes of their vehicles. Our comprehensive computational experiments allow for a detailed analysis of what impact tolling schemes can have on the structure and cost of routes for logistics service providers. Furthermore, with our experiments, we can analyze whether the various types of tolling schemes work in the way municipalities are expecting it.

We first summarize the existing tolling schemes around the world. To date, only a few cities and towns are applying congestion charges to their urban roads. The list includes Singapore, London, Milan, Stockholm, Gothenburg (Sweden), Valletta (Malta), and Durham (England). The types of existing tolling schemes include per gantry fee (Singapore), per-day fee (London, Milan, and Durham), per-entry fee (Stockholm, Gothenburg), and per-minute fee (Valletta) [2][3]. The tolling schemes we investigate in this paper are based on these existing schemes.

We model the vehicle routing problem based on a standard single-depot vehicle routing problem, where each vehicle starts and ends its route at the same depot. All drivers begin their day at a pre-specified time and work for a maximum duration. We assume that each customer can be visited at any time during the day. We assume that the travel time between locations is known, but depends on the time the vehicle starts its travel, i.e., time-dependent travel times. We approximate the distance between two locations via the Manhattan distance. The speeds of vehicles are approximated by the average speed of traveling along the path and a time-dependent speed factor identified in [4]. We assume that the vehicles do not need to wait at customers. After arriving at a customer, each vehicle incurs the service time and then immediately begins to travel to the next customer in the sequence. The total costs consist of the labor cost, fuel cost, and the congestion charge. We consider four types of congestion charges in this study: daily fee, per entry fee, per minute fee, and per gantry fee. For per entry, per minute and per gantry fees, we consider a fixed version and a time-dependent version. That is, a higher fee will be collected if the vehicle is traveling during peak hours. For each congestion charge type, we consider high, medium, and low charge levels.
2 Solution Approach and Experimental Design

To solve the vehicle routing problem, we adapt and apply the LANCOST heuristic proposed in [1], which is based on a tabu search heuristic and is applicable to solve the least time vehicle routing problems with multiple vehicles and time-dependent travel times.

Figure 1: City and Congestion Zones

In the computational experiments, we experiment with three different city sizes (large, medium, and small cities). For each city size, we experiment with two different congestion zone sizes, one at a time. Figure 1 provides an example of a city and its two congestion zones. We assume that a vehicle can enter a congestion zone via one of the entry points located on the border of the zone. We consider instances with 100 customers. We investigate how different tolling schemes may impact the total costs, the distance traveled by the vehicles (in and outside the congestion zone), the travel time, total working time of the drivers, and the emissions. Also, we compare the impacts of the same tolling scheme on different geographies, i.e., different city and congestion zone sizes.

3 Results

As an example of the results, Table 1 presents the results for the medium sized city with a large congestion zone and high congestion charge fees.

Table 1: Results for Medium Sized City, Large Congestion Zone, High Charge

<table>
<thead>
<tr>
<th>fee</th>
<th>scheme</th>
<th>cost</th>
<th>dist in zone</th>
<th>time</th>
<th>drive in zone</th>
<th>fuel</th>
<th>fuel in zone</th>
<th>veh enter</th>
<th>times enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>524.65</td>
<td>168.24</td>
<td>18.71</td>
<td>8.65</td>
<td>100.56</td>
<td>41.44</td>
<td>4.38</td>
<td>4.54</td>
</tr>
<tr>
<td>18</td>
<td>daily</td>
<td>13.99%</td>
<td>0.03%</td>
<td>1.99%</td>
<td>-0.16%</td>
<td>0.77%</td>
<td>-0.11%</td>
<td>-10.00%</td>
<td>-9.63%</td>
</tr>
<tr>
<td>17</td>
<td>fixed entry</td>
<td>13.12%</td>
<td>-0.78%</td>
<td>1.80%</td>
<td>-1.00%</td>
<td>0.76%</td>
<td>-0.94%</td>
<td>-11.00%</td>
<td>-13.93%</td>
</tr>
<tr>
<td>17</td>
<td>t-d entry</td>
<td>13.16%</td>
<td>-1.46%</td>
<td>0.31%</td>
<td>-1.62%</td>
<td>0.25%</td>
<td>-1.58%</td>
<td>-10.50%</td>
<td>-13.43%</td>
</tr>
<tr>
<td>0.06</td>
<td>fixed min</td>
<td>14.30%</td>
<td>-3.42%</td>
<td>-0.13%</td>
<td>-3.42%</td>
<td>0.14%</td>
<td>-3.42%</td>
<td>0.07%</td>
<td>0.33%</td>
</tr>
<tr>
<td>0.06</td>
<td>t-d min</td>
<td>14.55%</td>
<td>-4.21%</td>
<td>-0.09%</td>
<td>-4.27%</td>
<td>0.21%</td>
<td>-4.25%</td>
<td>-1.40%</td>
<td>-3.67%</td>
</tr>
<tr>
<td>0.42</td>
<td>gantry</td>
<td>12.70%</td>
<td>-7.63%</td>
<td>0.27%</td>
<td>-7.70%</td>
<td>0.88%</td>
<td>-7.68%</td>
<td>-0.63%</td>
<td>-1.77%</td>
</tr>
</tbody>
</table>

The first two columns reflect the fee being charged (fee) for entering the congestion zone and the charging scheme (scheme), respectively. The third through the last columns present the key metrics that we have measured. In the first row, we report the values found with no congestion fees collected, while in subsequent rows, we report the percentage change in the listed metrics relative to the metrics obtained with no congestion fees charged. This helps us understand how different congestion charge schemes change the
solution, especially their financial and environmental impact. From the results, we can see that the introduction of congestion tolls can help to reduce the emissions within the congestion charge zone. Also, having fewer vehicles entering zone is not necessarily better than having more entering zone, because the former may lead to traveling for longer distance in the zone. For instance, with the fixed per minute fee, there are 0.07% more vehicles entering the zone, but the emissions in the zone is reduced by 3.42%, while with the per day charge, the number of vehicles entering zone is reduced by 10%, but there is only 0.11% reduction in emissions in zone.

Overall, the observations illustrated by our computational experiments are summarized as below. First, the congestion pricing schemes can shift behaviors of commercial fleets. For instance, with a congestion charge, the vehicles may travel less in the congestion zone than with no charge. Surprisingly, the daily congestion charge does not always shift behaviors in the desired way, i.e., with a daily fee, the vehicles may even travel more in the congestion zone. Second, larger congestion fees usually have more impact than smaller fees and the gantry charging schemes are surprisingly effective for most city and congestion zone sizes. Third, having fewer vehicles or entrances to the zone can translate to less efficient use of the vehicles in the zone when compared with letting more vehicles enter the zone for short periods. Fourth, the congestion charging schemes investigated in this paper usually have more impact on the key metrics (i.e., distance in zone, drive in zone and fuel in zone) for larger city and congestion zones than for smaller sizes. When the city and congestion zones are small, smaller distances are travelled, then less impact will be felt. This implies that the daily entry fees are a particularly bad choice for small cities. Further, the gantry based charging schemes are useful for all city sizes with relatively large congestion zones. Last, the departure times of the vehicles will impact the key metrics of fleet behaviors and the number of vehicles required overall and to enter the congestion zone.

References


Carbon emission effects of consolidating shipments
- taking topological effects and temporal constraints into consideration

Marcel Turkensteen
Department of Economics, Aarhus University, Denmark. Email: matu@econ.au.dk

Geir Hasle
Mathematics and Cybernetics, SINTEF, Oslo, Norway. Email: geir.hasle@sintef.no

1 Introduction
We consider the emissions savings that can be achieved by combining pickups and deliveries on the same vehicle. In contrast with the Pollution-Routing literature, we assume that transport providers will minimize monetary costs.

We compare a non-consolidated set-up where deliveries and pickups are performed on separate vehicles and a consolidated set-up, where outbound and inbound items can be mixed freely on the vehicle, as long as capacity and temporal constraints are adhered to. A previous paper [9] presents the effect consolidation on carbon emission savings under several simplifying assumptions, namely the usage of Euclidean distances and the absence of temporal constraints. One of the striking results is that, although emission savings are often between 10% and 40%, there are situations where consolidation leads to a carbon emission increase. In this study, we investigate the effects of a richer model that accommodates road distances, tour duration constraints, and time windows.

Consolidated tours tend to be longer, but with more and on average shorter legs, than non-consolidated tours. This may change to some degree in case of temporal constraints and road distances. For example, time windows and tour duration constraints may limit the distance savings from combining deliveries and pickups, but they also reduce the load on the vehicle, meaning that the overall effect on emissions is not clear. In this paper, we conduct an extensive analysis on the impact of these factors based on computational experiments.

2 Literature and hypotheses
The problems of visiting a given set of locations from a central depot with possibly multiple vehicles are known as Vehicle Routing Problem (VRPs). Emission computations are included in the VRP variant called the Pollution-Routing Problem [1]. The standard approach in the
literature for computing emissions for vehicles with different loads is the so-called CMEM [1], which requires many parameters and a realistic speed profile and its validity may be questioned. Instead, we base our results on empirical findings that, on a fully loaded vehicle, between 10% (for small truck in urban traffic) and 50% (for large, heavy trucks on highways) are due to the load on the vehicle. We consider the extremes of 10% and 50% in our experiments.

Studies on the impact of factors such as geography and temporal constraints on vehicle routing are mainly from the field of continuous approximation, on locations that are uniformly distributed in an area. Boscoe et al. [2] find a high correlation between Euclidean and road distances in the US but warn that features such as bridges, islands, and irregular shorelines may lead to clearly longer distances. Nevertheless, Cooper [4] finds a strong correlation between Euclidean distances and vehicle routing costs in the British Midlands.

It is found in [8] that a tour duration limit has in principle a similar effect as a capacity constraint, assuming that the density of locations does not change. However, the non-consolidated tours covering a certain area in [9] tend to be longest, so tour duration limits would affect them most. Regarding time windows, Daganzo [5] studies the impact of having different intervals that serve as time windows for different demand points and finds that the total distance is a square root function of m. If constraints on the capacity make that closely located pickups and delivery locations with similar time windows cannot be visited together, this could have the same effect as increasing the number of intervals m in the consolidated set-up.

Based on the literature review, we expect that 1) changing from Euclidean distances to road distances will have little impact, and 2) the addition of temporal constraints will lead to reduce the distance emission savings from consolidation.

3 Models and experimental design

We mimic the decisions of a transport provider that has to serve a given set of delivery and pickup customers from a central depot with a homogeneous fleet of vehicles. The provider minimizes the number of vehicles as the primary objective, and the total distance driven as the secondary objective.

For the non-consolidated set-up we generate two instances of the Vehicle Routing Problem with Time Windows (VRPTW): one for the deliveries and one for the pickups. For the consolidated set-up, we generate an instance of the Vehicle Routing Problem with Pickups and Deliveries and Time Windows (VRPPDTW). We use a commercial solver ([3], [6]) to find
approximate solutions, where the distance and the average load for each tour determine the carbon emissions.

Our benchmark consists of the Li & Lim VRPPDTW instances [7] with approximately 100 pickup and delivery pairs. We modify the instances such that all deliveries originate from a single depot and all pickups are destined for the same depot. We investigate three road topologies: Denmark (DK), the Dutch province of South Holland (SH), and Illinois (IL), taken from Mileage Charts. We distribute the delivery and pickup quantities of selected Li & Lim instances over randomly selected locations in these three areas to make the instances comparable to the Euclidean case. The tour duration limits are currently set to three levels of tightness. The time-windows include those in the regular Li & Lim instances, but we also include time windows that are loose and that only apply to delivery locations.

4 Experimental investigation
Our preliminary computational experiments cover in total 120 instances.

In comparison with the corresponding Euclidean instances, the usage of road distances appears to reduce the emission and distance savings from consolidation to up to 10% for the DK topology, possibly hauls are on average shorter in case of consolidation. We plan further investigations into this phenomenon. In comparison to instances without temporal constraints, the emission savings typically decrease with the tightness of the tour duration constraint, as expected. However, in several cases, we obtain the surprising result that the addition of a tour duration constraint increases emission savings from consolidation. In those cases, more tours are needed to satisfy the duration constraints, which has a small effect on the total distance but leads to a lower load than in the non-consolidated set-up. The addition of time windows has unpredictable effects because the minimization of vehicles has the highest priority. When there are few tours in case of consolidation, these tours may be long in order to satisfy time windows. We have observed carbon emissions increases in such cases up to 8%, in particular when pickups tend to precede deliveries. In general, vehicles are less well utilized in the presence of temporal constraints, meaning that the impact of the load plays a smaller role than in [9].

5 Conclusions
We extend a previously investigated model with real-life aspects to more accurately measure the carbon emission effect of consolidating pickups and deliveries, namely real road distances and temporal constraints. Typically, there is a clear impact: when using road distances,
emission savings appear to be lower than for Euclidean instances, whereas temporal constraints reduce the emission savings of consolidation. However, we have seen several cases where the resulting lower load factor resulting from such constraints can increase savings. Detailed results, including additional experiments, will be presented during the conference.

References


---

\( ^1 \) The Li & Lim instances can be retrieved on [https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/](https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/)

\( ^2 \) The MileageCharts Database is available on [http://www.mileage-charts.com/](http://www.mileage-charts.com/)
1 Introduction

SYNCHRO-NET (http://www.synchronet.eu/) is a Horizon2020 European research project that aims to overcome the stress due to increasing transportation distances, higher complexity, and vulnerability of modern supply chains. The SYNCHRO-NET approach allows to plan and optimize effectively a complex supply chain through smart coordination mechanisms that emphasize synchro-modality (i.e., inter-modal transportation in which operators can switch between modes in real-time for higher efficiency) and slow steaming practices (i.e., operating cargo ships at significantly less than their design speed, reducing fuel costs and greenhouse gas emissions). An integrated optimization and simulation system (incorporating strategic and real-time logistics optimization, smart steaming ship simulation and monitoring, risk/benefits analysis, and stakeholder impact assessment [1]) has been developed. Within this platform, a so-called Strategic Optimization Toolset provides logistics operators with a decision support to plan routing and scheduling freight movements on truck, rail, and ship. The optimization is guided by several (possibly conflicting) attributes such as trip length, duration, emissions, and monetary costs [2]. However, given the high uncertainty of realistic transportation scenarios, robust solutions need to consider other risk-related aspects such as safety, flexibility, time and costs reliability. To this purpose, a Monte-Carlo Rollout risk analysis approach has been developed and integrated into the optimization toolset.

2 Risk Analysis for synchro-modal freight transportation

Within the SYCNHRO-NET platform, alternative transportation plans are provided by the basic optimization routines to logistics operators. To offer the possibility of wiser and more robust planning decisions, different Key Risk Indicators (KRIs), i.e., different aspects of the potential risks coming with each plan, are evaluated by the Risk Analysis Module (RAM) using data analysis techniques. However, the developed risk analysis follows a general approach to overcome the complexity of real synchro-modal freight transportation scenarios, in the hope it could be suitable also out of the specific project.

2.1 Risk Analysis Module interactions

As shown in Fig.1, the RAM is composed of two main components, the Risk Analysis Simulator (RAS) and the Risk Profiler (RP). The RAS receives a list of alternative optimized plans from the Optimization Engine,
according to the user preferences and constraints, and returns a list of calculated KRIIs through a Monte-Carlo Rollout simulation. To run the simulations, the RP generates random disturbances (i.e., unforeseen events that affect the regular schedule or the transportation path) for each link of a plan, based on available historical data. To this aim, existing knowledge on ports’ and links’ congestion is of fundamental importance for the robustness of the analysis. These data are collected during past routes’ executions.

![Risk Analysis Module structure and interactions](image1)

**2.2 Monte-Carlo Rollout simulation approach**

The Risk Analysis Module uses a simulation-based algorithm called Monte-Carlo Rollout [3]. A schematic overview of this procedure is shown in Fig. 2.

![Schematic overview of the Monte-Carlo Rollout approach](image2)

In this approach, a so-called random player generates path disturbances and delays on each input alternative (i.e., each different transportation plan), then a so-called decision maker checks if each plan is still feasible or not after being afflicted by each disturbance. If the analyzed plan results to be not feasible anymore for a certain disturbance (e.g., if the cargo is not ready in a specific port to be boarded for the next transportation link due to time delay in the previous one), the RAS sends new requests to the Optimization Engine to adapt the plan to the new unexpected conditions (e.g., new time constraints may be added to manage the delay). For each alternative, this simulation procedure iteratively continues (by recording the resulting cost and time for adaption) until the final desired destination is reached.
The generation of the random disturbances is an important aspect to consider, in particular when dealing with systems for which the quantity and the precision of available historical data is not known a priori. We have distinguished two main cases:

- Case 1: when there is not enough historical data, different initial distribution functions (chosen according to experience) are used to generate random numbers for disturbances. To simulate a path disturbance (such as works in progress along a road), we use the Bernoulli distribution to generate a random number between 0 and 1. If that number is less than 0.05, it means that the path will be actually changed and a new plan, using a different path, will be considered for adaption. For simulating a time disturbance, instead, we use different distribution functions for the three different transportation modes considered. These distributions are centered on an expected value for the possible delay. For example, for truck movements, we use a Normal distribution centered around the value 20 (i.e., we have higher tendency to generate the number 20) since it is quite consolidated that the average time delay is 20 minutes every 200 minutes of trip;

- Case 2: when historical data are enough for a robust simulation, we sample random disturbances and their values from the most relevant historical data. To identify the most relevant data, we use some machine-learning feature selection techniques to select its most discriminative property. For instance, concerning the transportation of goods through Irish sea, the season seems the most discriminative property (since in Winter there are probably larger and more frequent time delays than in Summer).

2.3 Key Risk Indicators

Four main KRIs are calculated for each plan through the above approach:

- **Flexibility**: is the average time spent for adaption. All the time deriving from the waiting of new suitable movements, in the case of misses, is summed up and averaged over all simulations;

- **Safety**: we first calculate how many times the accident occurred on a route, then the rate of occurrence is achieved by dividing by the total execution times. The final safety value is calculated by using 100 to subtract the happening rate;

- **Time deviation**: is the average time deviation with respect to the expected one. It sums up all the time deviations occurred in the risk simulations, and then achieves the averaged number by dividing by the total number of the risk simulations;

- **Cost deviation**: is the difference between the planned cost and the executed cost in the simulation. The executed cost is the total cost of all the executed links and the cost due to disturbances.

Finally, a general risk rating is calculated as a weighted function of the above four KRIs. This is useful to rank the alternative plans with respect to a common scale. Note that, during the plan optimization phase, the user can specify the value of those weights according to the importance he gives to the different KRIs.
3 Results and concluding remarks

The described RAM has been implemented directly into the SYNCHRO-NET Strategic Optimization Toolset by means of specific open-source Java libraries for statistical analysis. The KRIs resulting from a simulation-based risk evaluation of a transportation plan are shown along with other KPIs (cost, time, and emission) and plotted in a bar chart alike the example in Fig. 3. For each KRI, a bar indicates the average value (normalized in the range [0,100]) from the simulation runs. Additionally, a tooltip shows the standard deviation, the minimum and maximum value of the indicator over the runs. Hence, the decision-maker obtains an easy-to-use tool to compare alternative plans not only based on traditional KPIs but also considering possible consequences of real-time disturbances and adaptive actions. Moreover, the module enables to select the best transportation plan according to the personal risk aversion or affinity.

The experiences show that the developed risk analysis gives additional values in supporting robust routing and scheduling decisions for synchro-modal freight transportation. However, up today, KRIs are calculated only \textit{a posteriori} with respect to the plan optimization. A possible future research could be about integrating risk attributes directly in the optimization procedures.

Acknowledgements: Funding for this work was provided by the SYNCHRO-NET project, H2020-EU.3.4. – Societal Challenges – Smart, Green and Integrated Transport, ref. 636354.

References


A reactive decision support system for freight intermodal transportation:
a Revenue Management perspective

Yunfei Wang¹, Ioana Bilegan¹, Teodor Gabriel Crainic², David Duvivier¹
¹ LAMIH UMR CNRS 8201, Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, France
² CIRRELT and School of Management, Université du Québec à Montréal, Canada
Email: Ioana.Bilegan@univ-valenciennes.fr

1 Introduction

Broadly used in practice at the operational level for passenger transportation, Revenue Management (RM) is rather new to the freight transportation and quite few contributions exist in the literature (e.g., [1], [2]). The use of principles and concepts of RM in the models developed for freight transportation is sustained by the general understanding that, in a deregulated market, customers have different behaviors, different requirements and different degrees of willingness to pay. In this context, an important challenge is the way customer types, demand uncertainty and variability, etc. are integrated in the decision-making process in a consistent manner. At the operational level, RM techniques had already proven their increasing interest. Our research focuses on the translation of these aspects towards tactical planning and, moreover, towards the development of a Reactive Decision Support System (R-DSS). A simulation framework has been implemented to study the impacts of such techniques in service network design models, in order to improve their effectiveness in the context of a Revenue Management based decision support system dealing with both operational and tactical decisions. At the tactical level, the goal of the R-DSS is to identify service plans (itinerary, schedule & capacity) that maximize revenue, while satisfying in the best manner the forecasted demands, coming from regular, as well as spot customers. At the operational level, once the service plan is fixed, the decision is to accept (or to refuse) transportation demands, and to accordingly identify the optimal routing for the accepted ones, based on the urgency of the request, on the associated fare class, as well as on the corresponding customer category.

The overall objective of the R-DSS is to manage activities and resources in a reactive manner, i.e. by taking into account the latest data, information and knowledge available
in the system. In this perspective, updates of demand forecasts are regularly performed, at
the tactical level and at the operational one. Two mathematical models were developed in
our previous studies, the first to solve the service network design problem, at the tactical
level ([1]), and the second dedicated to the network capacity allocation problem at the
operational level ([2]). We have shown it is promising to consider RM concepts in the
decision-making process. Compared to traditional (no RM) approaches, higher expected
revenues and better resource utilization have been reported, independently, at the tactical
or at the operational level. We propose here a reactive decision support methodology for
freight intermodal transportation, meant to perform resource and revenue management at
two levels, based on the two RM models interacting during the decision-making process.
The R-DSS is designed so that communication between the two levels takes place in
both directions: from tactical to the operational level, when tactical plans are applied at
the operational level; from the operational to the tactical level, when data collection is
performed during operations to feed back the tactical model with aggregated data (e.g.,
customers’ behavior, demand characteristics, ...).

The case-study we focus on is barge transportation. Barge transportation is an impor-
tant research topic that started to draw increasing scientific attention in the recent decade.
Considered as sustainable, environment-friendly and economical, barge transportation has
been identified as a competitive alternative for freight transportation, complementing the
traditional road and rail modes. However, contributions related to barge transportation,
especially in the context of intermodal transportation, are still scarce (e.g., [2, 3, 4]).

2 A simulation framework to validate the R-DSS

According to the business relationship between a carrier and its customers, different cat-
egories of customers may have different behaviors. When modeling the decision-making
process we assume different categories of customers, with different types of requirements.
We assume carriers are able to make different decisions, based on their own experience and
according to the customer-relationships involved. Note that in our research, in addition to
the regular customers, with whom a carrier signs long-term contracts, spot customers, with
whom the carrier has no contractual commitments, are also assumed to make up the total
transportation demand. For each category of customers (regular or spot), transportation
demands with different fares are also considered. Each demand has a specific delivery
requirement (i.e., fast or slow delivery) and thus corresponds to a specific fare class.

The characteristics of the simulation framework used to validate the R-DSS proposed
are briefly described in the following. Given the information about the physical network,
potential services and forecasted demands, at tactical level, the proposed RM model ([1])
and a more traditional one, which does not include revenue management considerations,
are, in turn, applied to make optimal decisions about the service plan (itinerary, schedule & capacity). The service plans obtained are used as input for the operational level, where the decision is to accept or reject transportation requests. For the accepted ones, a predictive optimal routing is calculated, based on updated information about potential future demands expected on the network [2]. For the two decision levels, different granularity of demand forecasts is required. The transportation requests are assumed to arrive sequentially to the booking system. Within the simulation framework, at the operational level, they are generated following an iterative procedure that makes use of some probability distribution functions. In order to guarantee the consistency between the two levels of decision, the same type of probability distribution functions are used when generating demands at the tactical level, based on aggregated demand information.

3 Experimental plan

As described previously, at the tactical level, both RM model and a traditional model are applied, in turn, to obtain two different solutions for the service network design problem. Different groups of experiments are conducted to assess the performances of the proposed R-DSS from different perspectives. We first design a group of experiments to analyse the results when different types of behavior of the regular customers are assumed and tested. We then study, in a second group of experiments, the results obtained when the proportion of regular and spot customers varies. We test three situations: regular customers predominate, spot customers predominate, and regular and spot customer are, in equivalent proportion, present in the carrier’s market. A third group of experiments is meant to analyse results in terms of profit, resource utilization, etc., when assuming different degrees of accuracy for the forecasts of demand at the tactical level. Three situations are considered in this group: accurate, overestimated, and underestimated demand. Throughout the experiments, a set of Performance Indicators (PI), as proposed in [5], are selected and studied to offer better insights into the performances of the proposed R-DSS. For both levels, the total number of containers requested to be transported and the total number of accepted containers help in understanding how well the decisions and the activity planning, in terms of open services (tactical) and routing of demand (operational), allow the carrier to offer reactive answers to its customers. Another indicator, equally important for the carriers, used as a PI at both levels in our R-DSS, is the capacity usage of open services, which gives direct insights about the effective usage of the fleet of vehicles. At the tactical level, the number of open services for each type of vehicle helps in understanding the structural properties of the service network design solution obtained. At the operational level, economic PIs like revenue and net profit are used to have a direct assessment of the profitability of the R-DSS methodology proposed.
4 Conclusions

According to the initial numerical results, the proposed R-DSS generates more robust decisions, in terms of net profit and resource utilization, compared to more traditional decision support methods (with no RM consideration). The introduction of RM concepts at tactical level improves the interaction between the two levels of decision making for the intermodal barge transportation system studied. It is observed that the behavior of the regular customers, as well as their proportion on the market, with respect to the spot customers, have important effects on the profits of a carrier. We are going to discuss how a carrier could take advantage of these observations and under what circumstances the use of RM based decision support methodologies is best suited to plan transportation activities over the medium- and the short- terms.

Acknowledgments

Funding for this research is partially provided by the ELSAT 2020 project. The ELSAT 2020 project is co-financed by the European Union with the European Regional Development Fund, the French State, and the Hauts de France Region Council.

References


Branch-Cut-and-Price for the VRP with Time Windows and Convex Node Costs

Qie He
Department of Industrial and Systems Engineering, University of Minnesota

Stefan Irnich
Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Email: irnich@uni-mainz.de

Yongjia Song
Department of Statistical Sciences and Operations Research, Virginia Commonwealth University

1 The VRPTW and Convex Node Costs

We study an extension of the vehicle routing problem with time windows (VRPTW) [2] in which the objective is to minimize the sum of routing and customer inconvenience costs. The latter inconvenience costs are defined by general convex functions, one for each customer, that express the customer’s preference for a specific service start time. We call this problem the VRP with time windows and convex node costs (VRPTW-CNC).

For a more formal definition of the VRPTW-CNC, we rely on the following definition of the VRPTW: Let \( N = \{1, 2, \ldots, n\} \) be the set of customers. Each customer \( i \in N \) has a given demand \( q_i \) and a service time window \([e_i, \ell_i]\). A fleet of \( K \) homogeneous vehicles, each with capacity \( Q \), is stationed at the depot (vertices 0 and \( n+1 \)). For simplicity, we assume \( q_0 = q_{n+1} = 0 \) and that depot time windows \([e_0, \ell_0]\) and \([e_{n+1}, \ell_{n+1}]\) are given. The VRPTW is defined on the directed graph \( G = (V, A) \) with vertex set \( V = N \cup \{0, n+1\} \) and arc set \( A \). Each arc \((i, j) \in A\) has an associated travel time \( T_{ij} \) and routing cost \( c_{ij} \).

A feasible route \( P = (r_0, r_1, \ldots, r_m) \) in the VRPTW is an elementary 0-(n + 1)-path in \( G \) of length \( m > 1 \) that is feasible with respect to time-window and capacity constraints. The route \( P \) is time-window feasible if there exist schedule times \( t_0, t_1, \ldots, t_m \in \mathbb{R} \) with

\[
\begin{align*}
  t_i &\in [e_{r_i}, \ell_{r_i}] \quad \text{for all } i \in \{0, \ldots, m\} \\
  t_{i+1} - T_{r_i, r_{i+1}} &\leq t_i \quad \text{for all } i \in \{1, \ldots, m\}.
\end{align*}
\]

The route \( P \) respects the vehicle capacity if \( \sum_{i=0}^{m} q_{r_i} \leq Q \). It has routing costs \( c_P = \sum_{i=1}^{m} c_{r_{i-1}, r_i} \). The VRPTW asks for a set of up to \( K \) feasible routes that visit each
customer exactly once and minimize the sum of the routing costs.

The extension introduced in the VRPTW-CNC is that now each vertex \( i \in V \) has a convex inconvenience cost function \( f_i(t_i) \). The cost of a route \( P = (r_0, r_1, \ldots, r_m) \) becomes the sum of the routing costs \( c_P \) and the minimum inconvenience costs resulting from the solution of the following \((m + 1)\)-dimensional optimization problem:

\[
f_P = \min \sum_{i=0}^{m} f_{r_i}(t_i) \quad \text{subject to } (1).
\]

We refer to (2) as the service scheduling problem. Hence, the VRPTW-CNC is a VRPTW in which the routing costs \( c_P \) of a route \( P \) are replaced by \( c_P + f_P \), where the component \( f_P \) reflects the cost of customers’ inconveniences remaining after optimizing the schedule of route \( P \) in this respect. The VRPTW-CNC is a three-level optimization problem with interdependent levels for clustering, routing, and schedule optimization.

2 Contributions and Main Results

We propose the first effective branch-cut-and-price (BCP, [1]) algorithm for the VRPTW-CNC that is able to solve to optimality instances with 100 customers. We implemented several standard techniques such as \( ng \)-route relaxation and cutting-plane algorithms with robust (2-path cuts, capacity cuts) as well as non-robust cuts (subset-row inequalities).

The novelty of our algorithm lies in column-generation mechanism. We show that dynamic-programming labeling can be used to simultaneously solve the two lower levels of routing and schedule optimization, while the clustering in the first level is standard. As for the VRPTW, clustering relies on an extensive, path-based set-partitioning formulation of the VRPTW-CNC.

2.1 Solution of the Column-Generation Subproblem

Recall that the solution of the column-generation subproblem in the VRPTW can be established by dynamic-programming labeling algorithms [4]. Labeling algorithms iteratively extend partial paths in a vertex-by-vertex fashion until they lead to negative reduced-cost routes. Labels store a set of attributes characterizing a partial path so that feasibility checks and dominance between paths can be decided efficiently. For the VRPTW, the standard attributes describe the accumulated reduced cost, demand, (earliest service start) time, and visited vertices for a partial path.

In essence, the new subproblem for the VRPTW-CNC can be solved by replacing the two attributes for cost and time by a function \( f_P(t) \) that models the tradeoff between cost and service start time \( t \). For example, consider the following partial path \( P = (0, 1, 2, 3, 4, 5) \) with time windows \([c_i, \ell_i]\), travel times \( T_{i-1,i} \) and inconvenience functions \( f_i(t) \):
For this partial path $P$, the associated time-cost tradeoff curve is depicted in Figure 1. It is convex and defined piecewise. Using results from [3], our new characterization of the tradeoff curve is based on a careful analysis of possible block structures, i.e., the question for which consecutive vertices $i-1$ and $i$ the service scheduling problem has a binding constraint (1b). We derive forward and backward resource extension functions to enable bidirectional labeling, and non-trivial rules for domination between sets of partial paths.

### 2.2 Variation of the Inconvenience Cost

In one type of computational experiments, we analyze the impact that the inconvenience cost functions have. For each customer $i \in N$, we define $f_i(t) = \rho \cdot (t - e_i - \ell_i)^2$, where $\rho$ is a cost factor that we vary in the following. With this setup, the inconvenience functions are zero in the middle of their time windows.

Figure 2 shows how costs (inconvenience, routing, and overall cost) of optimal solutions vary for the 50-customer instance $R102.50$ of Solomon’s benchmark depending on the factor $\rho$. Note that an optimal solution to this instance as a pure VRPTW, i.e., for $\rho = 0$, has routing costs of 909.0 and employs 11 vehicles. Starting from $\rho = 10^{-5}$ we increased $\rho$ with a factor of 1.05 so that we reach $\rho = 1.0$ after 236 iterations. Each of these 236 instances is now solved twice as a VRPTW-CNC: In left part of Figure 2, the fleet size is not limited (as in Solomon’s benchmark). With increasing $\rho$, the inconvenience cost varies non-monotonically between 0 and approximately 102. In contrast, the routing and overall costs increase monotonically. The routing cost function is a staircase function while the overall cost changes continuously.
We also display in this left part of Figure 2 the change of the used vehicles: Starting from 11 for $\rho = 10^{-5}$ the final solution for $\rho = 1.0$ requires 23 vehicles so that the average route length is nearly two customers per route. It means that a high value of $\rho$ lets the solutions tend to only visit customers at their preferred service start time. A high $\rho$-value does not so much increase the inconvenience cost because high inconvenience costs can be circumvented by using more vehicles. Indeed, starting from $\rho \approx 0.36$, the solution does not change any more because the 23 routes can ensure that every customer $i$ is served at the preferred time $(e_i + \ell_i)/2$ so that routing and overall costs become identical (=1389.9).

In right part of Figure 2, we do not allow to increase the required fleet, i.e., we restrict the number of vehicles to 11. Now the inconvenience and overall cost increase very fast. For $\rho \approx 0.0039$, inconvenience costs exceed the routing cost. Larger values of $\rho$ hardly make sense.

References


An Exact Algorithm for the Vehicle Routing
Problem with Time Windows and
Flexible Delivery Locations

Alexander Döge, Markus Frey, Rainer Kolisch
TUM School of Management
Technical University of Munich, Munich, Germany
Email: alexander.doege@tum.de

Guy Desaulniers
Department of Mathematics and Industrial Engineering
École Polytechnique and GERAD, Montréal, Canada

Vehicle routing is well-studied in the operations research and management science literature. It has theoretical as well as practical relevance to scientific communities and industries, such as logistics and healthcare. In the classic vehicle routing problem (VRP), vehicles traverse a network with the objective to, e.g. minimize routing costs. Each destination in the network corresponds to exactly one customer, and each customer is visited exactly once. Various extensions exist for the VRP, e.g. associating customers with time windows leads to the VRP with time windows (VRPTW). In this paper, we present another extension of the VRP with substantial enhancement of the demand side: the VRPTW and flexible delivery locations (VRPTW-FL).

The VRPTW-FL is a VRPTW with a heterogeneous fleet in which each customer is served in one out of a set of potential service locations, each of which has a certain capacity. In addition, precedence relations exist between certain customers, i.e. serving one customer must be finished before another customer is served. Both the location capacity and the precedence constraints make this a very challenging problem. The VRPTW-FL arises in the context of hospital-wide therapist scheduling, where the location capacity is the capacity of the treatment rooms and precedence relations exist for treatments of the same patient within one day. The properties of the VRPTW-FL can be found in other industrial contexts as well. The location capacity exists, e.g. in electric vehicle routing where the number of charging stations is limited at a given location. Location flexibility also applies to parcel delivery where postal services can deliver to private households or parcel boxes. However, this problem differs from our topic since the time the parcels occupy the parcel boxes is stochastic and may exceed the planning horizon of one day whereas service times
in the VRPTW-FL cannot exceed one day.

The VRP and its extensions have been studied extensively in the literature. However, there are only two routing problems in which serving customers is possible in multiple locations: the vehicle routing-allocation problem introduced by [1], and the VRP with roaming delivery locations introduced by [2]. To the best of our knowledge, we are the first to consider capacities for service locations. The main contribution of this paper is the development of the first exact branch-price-and-cut algorithm for the VRPTW-FL. In particular, we introduce customized branching strategies and valid inequalities for this problem.

1 Branch-price-and-cut algorithm

To solve the VRPTW-FL, we propose a branch-price-and-cut algorithm, i.e. column generation is applied to compute lower bounds, cutting planes are generated to tighten these bounds, and if needed branching decisions are imposed to derive integer solutions. We first relax the location capacity and precedence constraints in the master problem, and later we will consider these constraints in the branching. Our column generation’s master problem, thus is identical to and the pricing problems similar to their equivalents for the VRPTW with a heterogeneous fleet. The pricing problems differs as the underlying graph is much bigger since for each customer one node exists per possible service location. Theoretically, the pricing problems correspond to elementary shortest path problems with resource constraints, which we solve using a bidirectional label setting algorithm with decremental state space relaxation. To quickly generate columns with negative reduced costs, we also use two types of heuristics: a heuristic dynamic program employing aggressive label dominance and eliminating unpromising arcs, as well as a tabu search (see also [3]).

Once column generation terminates, we branch for integrality if needed, and once integrality is reached, the location capacity and the precedence constraints are checked for violations. If a violation is detected, we branch on the start time windows of the nodes in the pricing problem to forbid this specific violation.

Note that adding the location capacity and precedence constraints to the master problem is possible, but this would shift the algorithm’s complexity from branching to pricing. The planning horizon would then have to be discretized for the location capacity constraints, leading to many more nodes in the pricing problems, and the duals of the precedence constraints would further complicate the pricing problems. The latter could be tackled by incorporating linear node costs within the labeling algorithm as described by [4]. Efficiently dealing with the expanded number of nodes from the location capacity is an interesting research question.
2 Branching strategies

We use two types of branching strategies: (1) branching on the arcs of the pricing problem to reach integrality, and (2) branching on time windows to enforce the precedence and the location capacity constraints.

2.1 Arc branching

Branching on job-location arcs: Branching on arcs works as follows: Select an arc such that the sum of the flows on this arc in the optimal solution is between 0 and 1. Then derive two branches: (a) forbid this arc in the solution, and (b) enforce this arc in the solution. Note that for the VRPTW-FL, a node in the graph of the pricing problem corresponds to a customer-location combination and not a single customer.

Branching on aggregated job arcs: Since multiple nodes might exist for the same customer, the graph may contain many arcs between any pair of customers, so branching on one arc at a time could be inefficient. Therefore, we first branch on aggregated customer arcs, i.e. all arcs connecting a customer $i$ to a customer $j$ regardless of locations. Branching ensures that either no flow exists on all of these arcs, or the sum of the flows on all of these arcs must be 1. Note that if all aggregated flows are integral, customers could still be served in multiple locations, and branching on job-location arcs would be needed.

2.2 Time window branching

Once integrality is reached, the precedence and the location capacity constraints are checked for violation. If violated, branching on the start time windows of the nodes in the pricing problem ensures that this violation cannot occur in any subsequent branch.

Branching on time windows to ensure job precedences: The precedence constraint between two customers $i$ and $j$ is violated if $T_i + s_i + t_{l_i,l_j} > T_j$ holds, where $T_i$ is the start time of $i$, $s_i$ is the service time for $i$, and $t_{l_i,l_j}$ is the travel time between locations $l_i$ and $l_j$, where $i$ and $j$ are served. Let $a_i$ be the earliest and $b_i$ the latest start time for customer $i$, then the time windows $TW_i$ for the child branches can be defined as follows:

- Branch 1: Forbid route of predecessor: $TW_{\text{pred}} = [a_{\text{pred}}, \lceil T_{\text{pred}} - 1 \rceil]$
- Branch 2: Forbid route of successor: $TW_{\text{pred}} = \lfloor T_{\text{pred}} \rfloor$, $b_{\text{pred}} \wedge TW_{\text{succ}} = \lfloor T_{\text{succ}} + 1 \rfloor$, $b_{\text{succ}}$

Note that if further reducing the time window of a node is impossible, this node, not the customer, is no longer feasible and is thus deleted from the graph.

Branching on time windows to ensure location capacities: If location capacities are violated in an integer feasible solution, time windows are adjusted such that the service either ends before the violation, or starts after the violation. Let $\tau$ be any time point within the time interval of a capacity violation, and let $Q_l$ be the capacity of location $l$, then for
the first \(|Q_l| + 1\) customers \(i\) violating the capacity in \(l\), the time windows are adjusted as follows:

- Branch 1: Service ends before violation: \(TW_i = [a_i, \lceil \tau - s_i - 1 \rceil]\)
- Branch 2: Service starts after violation: \(TW_i = \lceil \lfloor \tau \rfloor \rceil, b_i\)

If a time window adjustment is impossible, the corresponding node is deleted.

### 2.3 Valid inequalities

We use valid inequalities to tighten the lower bound of the column generation. Due to the start time windows of the customers, subsets of customers might exist which would automatically violate the location capacity when served in the same location. For these subsets of customers \(I_c\), we can impose the following constraints:

\[
\sum_{i \in I_c} \beta_{i,l} \leq Q_l \quad \forall \ l \in \mathcal{L}, c \in \mathcal{C}_l,
\]

where set \(\mathcal{C}_l\) contains all possible violations for location \(l\), \(Q_l\) is the capacity of location \(l\), and \(\beta_{i,l}\) indicates whether a customer \(i\) is serviced in location \(l\). Note that more of these cuts will be found, if time windows are small in comparison to the service duration. By branching on time windows, the length of the start time windows shrinks, increasing the chance of finding more violated cuts at a later stage of the branch-and-bound procedure.

### 3 Computational study

The first implementation of our algorithm shows promising results. By the time of the conference, we will have a thorough computational study in which we will examine location flexibility on a variety of performance metrics.

### References


An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and its Variants

Ali Mehsin Alyasiry, Michael Forbes, Michael Bulmer
School of Mathematics and Physics, Faculty of Science
The University of Queensland, St Lucia, Brisbane, Australia
Email: a.alyasiry@uq.edu.au, m.forbes@uq.edu.au, m.bulmer@uq.edu.au

1 Introduction

In the Pickup and Delivery Problem with Time Windows (PDPTW) vehicles with limited capacity must be routed to serve given requests each of which consists of a pickup location (origin) and corresponding delivery location (destination). For each request The origin must precede the destination and both must be in the same route. Any route must respect vehicle capacity and allowed time windows at each location, as well as constraints which apply to specific problem variants.

In this work, we propose a novel exact method by introducing a new methodology and formulation for solving the PDPTW. We propose a branch-and-cut approach based on fragments - a series of pickup and delivery requests starting and ending with an empty vehicle. Using fragments, we formulate a relaxed network flow model with side constraint and use lazy constraints to cut off any illegal solutions generated while solving the resultant integer program. This method is easy to implement and can be extended in a straightforward way to solve many variants of the PDPTW for problems where it is possible to generate all fragments.

Compared to approaches that consider entire routes, we reduce the number of variables by generating fragments of requests. We make multiple copies of each fragment by dividing the time windows of the starting nodes of each fragment into fixed width intervals and round down the arrival time at the other end to the closest time interval. We also do the same for every possible arc (empty vehicle arc) connecting delivery to pickup nodes which may be used to connect the fragments. We construct a new relaxed network flow model to chain the generated fragments.

We solve the resultant Integer Program using a general-purpose integer programming (IP) solver. Using a solver ‘Callback’ function, we check that every chain of fragments is a valid vehicle route whenever we find a candidate improved integer feasible solution. If
not, lazy constraints are used to cut off the combination of variables corresponding to the smallest portion of the chain that is illegal, and to eliminate any cycles that may occur.

We have tested our method to solve two variants of the PDPTW named the PDPTW with last-in-first-out (LIFO) Loading (PDPTWL) and the PDPTW with Multiple Stacks (PDPTWMS). Computational results confirm that our method significantly outperforms the current state-of-the-art algorithms for solving PDPTWL and PDPTWMS.

2 Methodology

2.1 Fragments

Generating all admissible routes that satisfy the PDPTW constraints, even for modest size problems, leads to a huge number of routes. Branch-and-price approaches attempt to overcome this problem by considering all possible routes implicitly, rather than explicitly. Nonetheless, solving such problems normally requires sophisticated procedures. As an alternative to branch-and-price approaches, we reduce the number of variables by generating fragments.

We define a fragment to be part of legal vehicle route such that the vehicle starts empty at a pickup node and ends empty at a delivery node, but it is never empty at any intermediate node. Fragments are similar to mini-clusters mentioned in the literature (see, e.g., [1, 2, 3, 4]). However, the purpose and the role of fragments here are different from those of mini-clusters. By definition fragments respect all capacity, time window, pairing and precedence constraints, as well as any other loading constraint such as those associated with the PDPTWL and PDPTWMS. An example of a vehicle route with 2 fragments is illustrated in Figure 1.

2.2 Time window discretization

We discretized the time window of all nodes except for the depot. Resulting from the discretization is a relaxed network that consists of several types of “timed” arcs: “start” arcs that are used to connect the depot to a timed pickup node, “end” arcs that connect a timed delivery node to the depot, “waiting” arcs that represent a vehicle waiting at each timed node, and “empty” arcs that are used to connect a timed delivery node to a timed pickup node. Time discretization also creates multiple versions of each fragment. A timed fragment starts from a timed version of the first pickup location and ends at a timed version of the last delivery location. Timed arcs and fragments incur the same cost as the original versions, and waiting arcs incur 0 cost.

We round down the arrival time of the timed fragments and the empty timed arcs to the closest time interval. Due to the cumulative effects of rounding down arrival times, not all paths through the relaxed network correspond to legal vehicle routes. Illegal paths
An empty timed arc or a timed fragment which starts at a later time will dominate its peer if both arrive at the same timed node, and if both have the same sequence of pickup and delivery nodes. Dominated arcs are removed from the relaxed network. An example of a simple vehicle route is illustrated in Figure 1.

3 Branch-and-Cut Algorithm

Our method can be summarized in the following algorithm steps:

1: Define the problem in a directed graph (network $H$).
2: Tighten the time windows of each node as much as possible.
3: Reduce $H$ size by eliminating inadmissible arcs (the resulting network $G$).
4: Generate fragments.
5: Construct the discretized network of timed fragments, timed empty arcs and timed waiting arcs (the resulting network $G'$).
6: Formulate the integer programming model.
7: Solve the linear relaxation problem with objective of minimizing $Z_v$ (the number of vehicles).
8: Set $v_{lb} = v_{ub} = \lceil Z_v \rceil$ (the upper and the lower bound of the number of vehicles).
9: loop
10: Solve the IP model with objective of minimizing the total routing cost using IP-
    Callback function to check integer solutions.
11: if solution is feasible then
12:   break loop
13: else
14:   Set $v_{ub} = v_{lb} = v_{lb} + 1$
15: end if
16: end loop

The IPCallback function is used to examine the solution for each vehicle route, if any
route is illegal in $G$ then the lazy constraints is added to cut off the shortest illegal chain
of fragments.

4 Computational Results

To the best of our knowledge, the current state-of-the-art methods for exactly solving the
PDPTWL and the PDPTWMS in the literature were introduced by [5] and [6] respectively.
Cherkesly et al. [5] used 60 instances (10 of each group) to test their algorithms for the
PDPTWL and Cherkesly et al. [6] used 1,914 instances (319 of each group) to test their
algorithms for the PDPTWMS.

Tables 1 and 2 show summary of the computational results and their columns represent
the following: instances by group (Group), the number of instances in each group that
are solved to optimality (Solved), the average CPU time in seconds (Sec.) of each group
on roughly comparable computers, considering only instances solved by both methods,
including the improvements made later by [7].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Sec.</td>
<td>Solved</td>
<td>Sec.</td>
</tr>
<tr>
<td>AA</td>
<td>10</td>
<td>144.6</td>
<td>10</td>
<td>1.3</td>
</tr>
<tr>
<td>BB</td>
<td>10</td>
<td>197.6</td>
<td>10</td>
<td>2.2</td>
</tr>
<tr>
<td>CC</td>
<td>6</td>
<td>892.1</td>
<td>7</td>
<td>189.0</td>
</tr>
<tr>
<td>DD</td>
<td>5</td>
<td>476.4</td>
<td>8</td>
<td>39.0</td>
</tr>
<tr>
<td>AA*</td>
<td>8</td>
<td>319.3</td>
<td>10</td>
<td>1.3</td>
</tr>
<tr>
<td>BB*</td>
<td>7</td>
<td>461.1</td>
<td>10</td>
<td>2.0</td>
</tr>
<tr>
<td>Total</td>
<td>46</td>
<td>55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


A branch-price-and-cut algorithm for the TWAVRP – addressing symmetry in route synchronization

Kevin Dalmeijer
Department of Econometrics, Erasmus School of Economics, Rotterdam, The Netherlands
Email: dalmeijer@ese.eur.nl

Guy Desaulniers
Department of Mathematics and Industrial Engineering, Polytechnique Montréal, Montréal, Canada

February 16, 2018

1 Introduction

We consider the Time Window Assignment Vehicle Routing Problem (TWAVRP), the problem of assigning time windows to clients in a distribution network before the demand of the clients is revealed. It is assumed, however, that a list of possible demand scenarios and corresponding probabilities is given. The TWAVRP asks for an assignment of time windows to the clients and, for each scenario, a set of feasible routes that adhere to the assigned time windows and the scenario specific demand, such that the expected cost of distribution is minimized. That is, each client is assigned a single time window, which should be respected, regardless of which scenario occurs. Each assigned time window, or endogenous time window, is of fixed width. Furthermore, the endogenous time windows must be assigned within given exogenous time windows.

The TWAVRP was first introduced by Spliet and Gabor [1], and was inspired by distribution networks of retail chains. In this setting, the clients are retail stores that are supplied from a central depot and the exogenous time windows represent the opening hours of the stores. In retail, stores are often assigned a time window that does not change for a longer period of time (e.g., one year). The TWAVRP was introduced to assign time windows in this setting, taking the uncertainty of the future demand into account.

In [1], a branch-price-and-cut algorithm is presented to solve TWAVRP instances with up to 25 clients and three demand scenarios, within one hour of computation time. In [2], a
branch-and-cut algorithm for the TWAVRP is introduced, which makes use of a new class of valid inequalities; the precedence inequalities. This algorithm solves the benchmark instances introduced by [1] on average 193.9 times faster than the original algorithm, and allows for instances with up to 35 clients and three demand scenarios to be solved.

The TWAVRP can be seen as a vehicle routing problem with operation synchronization constraints [3]. That is, the routes in the different scenarios have to be synchronized, such that the visits to a client (the operations) in different scenarios are at approximately the same time.

It is well known that symmetry has a negative impact on branch-and-bound algorithms, as it can lead to multiple branches corresponding to symmetries of the same solution. In this paper, we investigate symmetries due to the synchronization of the routes, and we show that symmetry has a big impact on both the algorithms by [1] and [2].

We then present a branch-price-and-cut algorithm that avoids this symmetry by branching on (undirected) edges instead of branching on (directed) arcs. For partial solutions with integral edges, we present an algorithm that either proves or disproves that the partial solution is synchronizable. Valid inequalities are introduced to cut off the current partial solution in case it is not synchronizable.

Surprisingly, we find from computational experiments that branch-price-and-cut can outperform branch-and-cut, if symmetry in route synchronization is properly addressed. This is in contrast with [2], in which branch-and-cut outperforms branch-price-and-cut.

2 Symmetry in route synchronization

By analyzing the TWAVRP instances introduced by [1], it becomes clear that the route synchronization aspect of the TWAVRP often leads to unwanted symmetries. E.g., we observe that many instances have multiple optimal solutions. These solutions typically make use of the same edges, but with some of the routes performed in the reverse order. We call a partial solution in which the edges (but not necessarily the arcs) are integral, to be synchronizable if there exists a feasible solution to the TWAVRP which uses only these edges. That is, we can assign directions to the paths and we can assign the corresponding visiting times, such that the resulting routes are synchronized.

2.1 Verifying synchronizability

For our branch-price-and-cut algorithm, it is important to be able to distinguish synchronizable and non-synchronizable partial solutions. For this purpose, we introduce a mixed integer program (MIP), in which the integer variables correspond to the directions of the paths in the partial solution. More precise, we introduce a variable $z_p \in \{-1, 1\}$ for all paths $p$. If path $p$ is traversed such that the client with the lowest index is visited first,
then $z_p = 1$, else $z_p = -1$. A partial solution is synchronizable if and only if the MIP has a feasible solution. The MIP can be solved by a standard MIP solver, but we also present a depth-first search algorithm to solve it.

In experiments we observe that the above algorithm can be slow, especially when the partial solution is non-synchronizable. To speed up the algorithm, we present a polynomial time pre-processing step, based on the precedence inequalities in [2]. In our case, the precedence inequalities imply that for some combinations of paths $p$ and $p'$, we can deduce that $z_p z_{p'} = 1$. For other combinations, we can deduce that $z_p z_{p'} = -1$. This allows for the construction of a conflict graph, in which each vertex corresponds to a path, and each arc indicates the relationship between $p$ and $p'$.

We show that finding conflicts in the conflict graph is equivalent to coloring a bipartite graph, and can thus be performed in polynomial time. If no conflict can be found, we show that the number of integer variables in the aforementioned MIP can be reduced to the number of components of the conflict graph, which is typically significantly less than the number of paths.

3 Branch-price-and-cut algorithm

Our branch-price-and-cut algorithm is based on the set-partitioning formulation presented in [1], strengthened with capacity cuts [4], subset-row inequalities [5] and precedence inequalities [2]. The pricing problem is an elementary shortest path problem with capacity constraints and time windows constraints, and additionally, linear node costs. To solve the pricing problem, we use a simplified version of the method proposed in [1].

The branch-price-and-cut algorithm is implemented in GENCOL. We branch on the number of vehicles and on fractional edges. When a partial solution is encountered with integral edges, we use the algorithm as described in Section 2.1 to verify synchronizability of the partial solution. If the partial solution is synchronizable, we have found a feasible solution to the TWAVRP, and the current node can be pruned. If we find that the partial solution is not synchronizable, we add a synchronization cut: a valid inequality that cuts off the non-synchronizable partial solution.

4 Results and conclusion

We test our algorithm on the benchmark instances introduced in [1], with a time limit of one hour. Our preliminary results suggest the following. If we branch on arcs, and we thus ignore the symmetry in route synchronization, our branch-price-and-cut algorithm outperforms the algorithm in [1] and performs comparable to the algorithm in [2]. In total, 27 out of the 90 instances could not be solved to optimality within one hour of computation time.
If we address symmetry using the strategy presented above, without the pre-processing step discussed in Section 2.1, we see that performance improves: 18 previously unsolved instances can now be solved to optimality, and the average run time decreases from 1304 seconds to 512 seconds per instance.

Finally, if we also apply the pre-processing step, another five previously unsolved instances can be solved to optimality and the average run time decreases further to 301 seconds per instance.

We have demonstrated that symmetry in route synchronization is highly relevant for the TWAVRP, and properly addressing symmetry speeds up solution methods. Furthermore, the proposed polynomial time pre-processing step is shown to be effective, and allows for reducing the solution time even further.

From our computational experiments we conclude that, in contrast to earlier research, branch-price-and-cut can outperform branch-and-cut for the TWAVRP. Previously, instances with up to 35 clients and three demand scenarios could be solved to optimality within one hour of computation time. With the presented branch-price-and-cut algorithm, instances with up to 50 clients and three demand scenarios can now be solved.

In this paper, we focus on symmetry in route synchronization, and we apply our ideas to the TWAVRP. In the future, it may prove interesting to apply the same ideas to other vehicle routing problems involving synchronization.

References


1 Introduction

In recent years, several researchers have looked at various models for using unmanned aerial vehicles, or drones, in different non-military applications domains, including parcel delivery [1], traffic planning and management [2], traffic monitoring [3], agriculture surveillance and crop spraying, monitoring fires, sports and entertainment event coverage [4].

In this paper, we consider the healthcare sector as an application area for the use of drones. In this sector, drones could potentially create a significant improvement in terms of cost and time required for providing essential medical supplies such as defibrillators, blood samples, and vaccines to communities in rural areas and in underdeveloped regions [5, 6, 7, 8, 9]. In particular, we study the problem of sending perishable supplies to demand points that are not well connected to the rest of the network, for example, communities which are accessible only via a single road or only via poor quality roads. We look at two different scenarios. In the first scenario, we assume that demand points are close enough to a drone launch station where the supply is also stored. In the second scenario, we assume that the distance from the demand points to the nearest supply point is greater than the flight range of a drone. Consequently, a two-stage delivery system is employed where traditional transportation is employed to deliver to a drone launch station which is sufficiently close to the demand points to address the last-mile deliveries issues. In this context, we defined two problems, namely P1 and P2, to optimally locate, assign and
schedule drones to serve all the demand points. Both the problems are formulated as integrated location-scheduling optimization problems, as explained in the next section.

2 Problem Description

Let \( D = \{1, 2, \ldots, m\} \) be a set of demand points which we assume is known in advance. Each demand point has a request for a package which needs to be delivered within a given due date \( d_k, \forall k \in D \), and it cannot be delivered before a given release date \( r_k, k \in D \). Since the package contains perishable items, the sooner it is delivered the better. Hence, we assume each demand point has a utility function \( e_k(\cdot) \) which is a convex decreasing function of its delivery time. We consider that a drone can deliver one package at a time and that it starts its trip from a launch platform which can operate only one drone. That is, we assume a drone is launched from the platform, makes a delivery and then returns to the platform for a change of battery and for picking up the next package to be delivered \([10, 11]\). Let \( S = \{1, 2, \ldots, n\} \) be a set of potential locations for drone launch platforms.

We would like to answer the following question:

**Problem P1:** What is the optimal location of \( p \) launch platforms so that all the demand points are served and the total utility is maximized?

The mathematical formulation of the problem is given in Figure 1. We denote by \( d_{jk}, \forall j \in S, \forall k \in D \), the flying time needed by a drone launched from platform \( j \) to reach the demand point \( k \) and to come back to the platform. The set of demand points which are within the maximum flying time of a drone operating from platform \( j \) is denoted by \( D(j) \). The decision variables of the model are as follows: \( x_j \in \{0, 1\}, \forall j \in S \) is equal to 1 if the launch platform location \( j \) is selected, 0 otherwise; \( u_{jk}, \forall j \in S, \forall k \in D \) is equal to 1 if demand point \( k \) is served by the drone in platform location \( j \); \( t_{jk} \geq 0, \forall j \in S, \forall k \in D \) is the delivery time at demand point \( k \) which is served by drone at platform \( j \); \( z_{jhk} \in \{0, 1\}, \forall h, k \in D \) is equal to 1 if demand points \( h \) and \( k \) are both served by the drone at platform \( j \), and \( h \) precedes \( k \), and 0 otherwise.

When a tandem strategy which combines land transportation with last mile drone delivery needs to be optimized, problem P1 can be extended to account for such an integrated strategy. Let \( W = \{1, 2, \ldots, k\} \) be a set of potential locations for warehouses (e.g., storage areas, hospitals, clinics) where packages are stored. We would like to answer the following questions:

**Problem P2a:** What is the optimal location of \( q \) warehouses and of \( p \) launch platforms so that all the demand points are served within the due date?

**Problem P2b:** What is the optimal location of \( q \) warehouses and of \( p \) launch platforms so that all the demand points are served and their utility maximized?

The mathematical formulation of problems P2a and P2b is not shown due to the limited.
\[
\begin{align*}
\text{max} & \sum_{j \in S, k \in D} e_k(t_{jk}) & \rightarrow \text{Total utility is maximized.} \\
\text{s.t.} & & \\
\text{Location-allocation Constraints} & & \\
\sum_{j \in S} x_j & = p & \rightarrow \text{Exactly } p \text{ platforms are selected} \\
\sum_{j \in S} u_{jk} & = 1 & \forall k \in D \rightarrow \text{Each demand point is served} \\
\sum_{k \in D} u_{jk} & \leq Mx_j & \forall j \in S \rightarrow \text{A platform can be associated} \\
& & \text{with a demand point only if it is selected} \\
\text{Scheduling Constraints} & & \\
t_{jk} & \leq dku_{jk} & \forall j \in S, \forall k \in D \rightarrow \text{A demand point is served within} \\
t_{jk} & \geq r_ku_{jk} & \forall j \in S, \forall k \in D \text{ its duedate and release date} \\
t_{jk} - t_{jh} - d_{jh} + M(1 - z_{jkh}) & \geq 0 & \forall j \in S, \forall k > h \in D \rightarrow \text{Precedence constraints among} \\
t_{jh} - t_{jk} - d_{jk} + M(1 - z_{jkh}) & \geq 0 & \forall j \in S, \forall k > h \in D \text{ demand points served by the} \\
z_{jkh} & \leq 0.5(u_{jh} + u_{jk}) & \forall j \in S, \forall k, h \in D \rightarrow \text{same platform} \\
z_{jkh} & \geq u_{jh} + u_{jk} - 1 & \forall j \in S, \forall k > h \in D \\
z_{jkh} & \leq 1 & \forall j \in S, \forall k, h \in D \\
\text{Decision Variables} & & \\
x_j & \in \{0, 1\} & \forall j \in S \rightarrow \text{Platform selection variable} \\
t_{jk} & \geq 0 & \forall k \in D, \forall j \in S \rightarrow \text{Delivery time at demand point} \\
z_{jkh} & \in \{0, 1\} & \forall j \in S, \forall h, k \in D \rightarrow \text{Joint service variable} \\
u_{jk} & \in \{0, 1\} & \forall j \in S, \forall k \in D \rightarrow \text{Platform-Demand} \\
& & \text{point association variable}
\end{align*}
\]

Figure 1: Mathematical formulation for Problem P1.

length of this paper. It is, however, a modification of the mathematical formulation of problem P1 where the following additional decisions are accounted for: (i) which warehouses to select among the available ones, and (ii) which launch platforms are associated with each warehouse.

### 3 Summary of Results

In this paper we study the computational complexity of problems P1 and P2. Complexity of some special cases is also analyzed. Heuristic solutions of the problems are presented and compared with the solution obtained by solving exactly the mathematical formulations employing standard solvers.
References


Ship and Drone Routing Problems

Stefan Poikonen
Department of Mathematics,  
University of Maryland, College Park, USA  
Email: spoikone@math.umd.edu

Bruce Golden
R.H. Smith School of Business,  
University of Maryland, College Park, USA

1 Background to Ship and Drone Routing Problems

1.1 Motivation
In the aftermath of major natural disasters, there are frequently large scale humanitarian relief efforts conducted by the naval forces of various countries. The Indian Ocean tsunami in 2004, Hurricane Katrina along the Gulf Coast of the United States in 2005, the 7.0 magnitude earthquake in Haiti in 2010, and the major earthquake and tsunami off the coast of Japan in 2011 all produced responses from the navy of the United States, the United Kingdom, and others. Both India and China have signalled their desires to take on a larger role in humanitarian responses to disasters in the Pacific and Indian Ocean basins.

Responses to these major natural disasters may be burdened by many factors complicating the distribution of supplies to affected regions. Ports, roads, and other infrastructure along the coasts may be severely damaged. Therefore, we introduce the Ship and Drone Routing Problem (SDRP) and variants. These problems consider the ability of an unmanned aerial vehicle (“drone”) to launch from a ship to deliver emergency supplies. The increase in drone capabilities in recent years and the ability of a drone to overfly difficulties on the ground makes a ship and drone combination potentially very effective in distributing critical supplies. There may be other applications including search-and-rescue, military operations, and regularly scheduled delivery of goods to island locations.

1.2 Literature Review
The closest analogue in the literature are problems related to truck-and-drone tandems for delivery of packages. The truck may directly deliver packages or deploy a drone to deliver packages. A few papers that consider this model include Murray and Chu [2], Agatz et
al. [1], Wang et al. [4], and Poikonen et al. [3] However, unlike these problems, our proposed model will allow the larger vehicle to operate in a continuous space, rather than on a defined network, represented by a graph.

2 The Ship and Drone Routing Problem: Main Version

Let $T$ be a set of target locations. Each $t \in T$ must be visited by a drone. The drone is deployed from a ship, satisfies the demand of a single target location, then returns to the ship for refueling/recharging before being deployed to the next target. We assume both ship and drone are free to move according to the Euclidean metric. The drone is capable of carrying 1 unit of supplies, and we assume each $t \in T$ only demands one unit of supplies. The drone is assumed to have finite range $R$ and travels at $\alpha > 1$ times the speed of the ship. (In the case that $\alpha \leq 1$, the optimal solution is the same as the solution to the traveling salesman problem.) There is a start location given $(t_0)$, where both ship and drone begin. The objective is to visit each $t \in T$ and return ship and drone back to the start location in the least possible amount of time.

3 Solution Methodology

This problem combines two forms of optimization. Firstly, we must determine the sequence of target visits, which is a form of combinatorial optimization. Secondly, for each target $t \in T$, we must decide upon a launch and landing point ($\text{launchPoint}(t)$ and $\text{landingPoint}(t)$). The launch point is the location where the drone deploys from the ship; the landing point is the location where the drone returns to the ship. This decision is set in continuous space.

Suppose we are provided a sequence of visit locations, $S = [t_1, t_2, ..., t_n]$. We may find the optimal set of launch and landing locations by solving a second order cone program.

\[
\begin{align*}
\text{minimize} & \quad \sum_{k=1}^{n+1} (cTime(k) + sTime(k)) \\
\text{subject to:} & \\
\text{For } k=1 \text{ to } n & \\
& \|\text{launchPoint}(t_{k+1}) - \text{landPoint}(t_k)\| \leq cTime(t_k) \\
& \|\text{launchPoint}(t_k) - \text{landPoint}(t_k)\| \leq sTime(t_k) \\
& \|t_k - \text{launchPoint}(t_k)\| \leq \text{outboundDroneDistance}(t_k) \\
& \|t_k - \text{landPoint}(t_k)\| \leq \text{inboundDroneDistance}(t_k) \\
& (\text{outboundDroneDistance}(t_k) + \text{inboundDroneDistance}(t_k))/\alpha \leq sTime(t_k)
\end{align*}
\]

End For

$\text{launchPoint}(t_0) = \text{landPoint}(t_0) = \text{launchPoint}(t_{n+1}) = \text{landPoint}(t_{n+1}) = t_0$

We may now find a solution for a fixed sequence $S$. However, we are still posed with
the question of finding the optimal choice of $S$. We construct a branch-and-bound tree where every node of the branch-and-bound tree corresponds with a particular sequence of visit locations. The root node begins with an arbitrary small sequence $[t_0, t_1, t_0]$, where $t_0$ is used to represent the starting/final location. The lower bound of a node is given by applying the above second order cone program to the sequence associated with the node. The children of a parent node insert one additional target location into the sequence. For example, the children of root node have sequences $[t_0, t_1, t_2, t_0]$ and $[t_0, t_2, t_1, t_0]$. By only inserting additional target locations, the objective value of a child node most always be at least as large as the parent node’s objective value. Thus, we have a valid lower bound. An upper bound for a node is $\infty$, unless the sequence attached to the node contains each $t_i \in T$, thus representing a feasible solution, that visits all target locations. In such a case, the upper bound and lower bound at that node are equal.

The above method is capable of finding the optimal solution in minutes for less than 30 targets. The presentation will contain more detailed tables of results. One sample solution for $|T| = 40$ targets, $\alpha = 2$, and $R = 5$ is given in Figure 1.

![Figure 1: A sample solution to the Ship and Drone Routing Problem. Black lines indicate the path of the ship. Blue asterisks represent the location of the targets. Red and blue lines represent the outbound and inbound flight paths of the drone, respectively.](image)

4 The Ship and Drone Routing Problem: Generalizations and Variants

The solution method described allows some flexibility. The constraints and objective function of the second order cone program may be modified to reflect slightly different problem assumptions. We can, for example, define a visit of target $t_i$ as passing within a specified radius $RR_i > 0$ of a target. This may have applications to signal collection, search-and-
rescue, or other domains where line-of-sight is important. This may be accomplished by modifying the second order cone program solved at each node of the branch-and-bound tree. The rest of the solution method remains unchanged, however. Alternatively, we could modify the objective to minimize a linear combination of travel distance of the ship and flight time of the drone. In general, we are free to modify the second order cone program constraints and objective, so long as we maintain the structure of a second order cone program.

Other variants of this problem require more substantial modifications. We propose an optimal solution method to a variant of this problem that allows a drone to visit multiple targets consecutively before returning to the ship. This is done by modifying our branching procedure, and assigning an order-dependent partition to each node of the branch-and-bound tree. We also consider the case where dry land or geopolitical boundaries do not allow the ship to operate according to the Euclidean metric. We modify our algorithm to produce feasible solutions avoiding these regions. However, avoiding these regions creates non-convexity within the problem, so our solution method to this variant is only heuristic, rather than exact.

5 Conclusions

We propose a solution methodology to ship-and-drone routing problems, where both vehicles operate in continuous space, rather than on a network metric. This method offers some flexibility and is shown to optimally solve instances of dozens of nodes.

References


E-commerce is expected to grow 11.4% annually. One of the biggest trends in this growing e-commerce market is same-day delivery (SDD). The same-day delivery segment is expected to outpace general e-commerce growth with an annual growth rate of 40% [1].

While a source of growth, SDD leads to significant cost challenges for service providers. To overcome the cost challenges of SDD, companies have begun to incorporate drones into their SDD operations. Drones have the advantage that they enable fast and direct delivery from the depot to a customer regardless of the traffic conditions. However, they can transport only one item per trip and require recharging or a battery swap afterward. Thus, drones may not be able to entirely replace conventional delivery vehicles, particularly when volumes are high.

The problem of routing road-based delivery vehicles (hereafter referred to as “vehicle”) and drones routing problem is a routing problem with a heterogeneous fleet, albeit one focused on the dynamic same-day delivery routing problem. We call the problem the same-day delivery routing problem with heterogeneous fleets (SDDPHF). During a shift, a fleet of vehicles and a fleet of drones deliver goods from a depot to customers. These customers request orders during the shift and are unknown before the time of their order. For each ordering customer, the provider must decide whether or not the order can be served on the same day and whether a vehicle or a drone performs the delivery. If SDD is offered, either a drone is loaded and sent to the customer or a vehicle picks up the order at the depot and delivers it within the delivery deadline. The objective is to maximize the expected number of customers served with SDD. During a shift, a fleet of vehicles and a fleet of drones deliver goods from a depot to customers. These customers request orders during the shift and are unknown before the time of their order. For each ordering customer, the provider must decide whether or not the order can be served on the same
day and whether a vehicle or a drone performs the delivery. If SDD is offered, either a drone is loaded and sent to the customer or a vehicle picks up the order at the depot and delivers it within the delivery deadline. The objective is to maximize the expected number of customers served with SDD.

In previous research, two of the authors explore whether and how a combination of vehicles and drones may reduce the required delivery costs and increase the number of customers served in SDD operations. Ulmer and Thomas [2] present a method to determine whether to use a drone or a vehicle for delivery. The question of drone or vehicle needs to account for the impact of the immediate decision on the ability to meet as yet unknown future requests. Given the scale of both the state and decision spaces, [2] seek to identify a good heuristic decision making policy. Their proposed method takes advantage of the intuition that, generally, vehicles may be suitable in downtown areas close to the depot and with high customer density while using drones may be beneficial for more distant suburban areas with widely dispersed customers. Based on this idea, to facilitate decision making, [2] introduce a parametric policy function approximation (PFA). In a PFA, one usually seeks to determine the best values for a parameterized policy. PFAs are appealing because they are easy to understand and is highly runtime-efficient. In this case, the parameter is a threshold of travel distance from the depot that splits the service area into two zones. Customers in the zone within the threshold are preferably served by vehicles and customers in the zone further distant than this threshold are preferable served by drones.

To determine the best threshold, [2] iterate through a set of potential values and evaluate its performance, choosing the best one. The approach is effective and computationally efficient, but it is limited in its ability to scale. As an alternative, in this research, we borrow from recent advances in reinforcement learning and particularly the success of DeepMind in training a computer to beat the best human players in the world at the game AlphaGo. The most recent version of DeepMind’s AlphaGo relies only on techniques from reinforcement learning. That is, the system learns by playing games of Go against itself. Specifically, AlphaGo Zero, as described in [3], and its earlier versions use a variant of policy gradient descent to train a neural network to output what action to take given a particular state of the game board. In this research, we demonstrate how these concepts can be applied to learn high-dimensional, state-dependent policies for the SDDPHF.

1 Preliminaries

Neural nets can be described by three layers. The first layer is an input layer, the second the hidden layer, and the third an output layer. The input layer is described by nodes, each of which represents a feature selected from the current state of the system. In the description of [2], there is a single feature, the distance from the depot. The values from the input layer are then combined using an affine transformation and sent to the nodes of the hidden layer. The hidden layer nodes then transform the linear combination of the input layer values. In
modern neural net implementations, there is often more than one hidden node layer. The most common transformation is done using a sigmoid function \( \frac{1}{1-e^{-x}} \). The output of the hidden layer nodes is then again combined, usually through an affine transformation, and sent to the output nodes. The output nodes are the value which users use to make decisions. In our case, this value will be the probability with which we should choose a given action.

With single feature, the distance to the depot of a new request, we can model a simple feed forward neural net with a single input node, a single hidden layer and single hidden-layer node, and a single output node. Figure 1 presents such a neural net graphically. In this case, the input \( x \) is the distance from the depot of a new request. This value is transformed using \( w_0 \) and \( w_1 \) to \( w_0 + w_1x \). This sum is then transformed using a sigmoid in the hidden layer. As we have only one hidden node, we do not transform the output and the output of the hidden layer, \( y \), is the same as that of the output node. The output value \( y \) can be interpreted as a probability with which we should choose a vehicle given the given distance from the depot of a new customer.

Using this simple neural net, we can match existing results. As an example, we consider an instance in which there are 500 expected orders whose locations are normally distributed around a central depot, three vehicles, and 10 drones. Travel speeds, service and loading times, and drone recharging times are the same as given in Ulmer and Thomas [2]. In this case, as shown in Figure 2a, the best threshold found by the iterative method is 13. Training the neural net using the standard policy gradient method [4] results in the sigmoid shown in Figure 2b. The resulting neural net has parameters \( w_0 \) and \( w_1 \) of 18.6667 and 73.1956, respectively. The threshold is 12.71, which corresponds to the point of the sigmoid at which the associated probability crosses 0.5. Thus, like the original method, the neural net suggests switching from vehicles to drones when the new customer request is farther than 13 units of distance from the depot.
2 Next Steps

While the proposed neural net can match the results of [2], Figure 2c shows that it requires 130,000 iterations to converge, four times as many iterations to converge as was needed in the iterative method. However, our goal is to increase the number of features that we consider. In that case, the computational effort of the iterative method increases combinatorially. Neural nets can handle additional features simply by adding input and hidden layer nodes. There is some increase in computational effort because of the need to update the parameters of the additional hidden layer nodes, but this computational cost grows linearly in the hidden layer nodes. The next step in our research will be to develop a method for using neural nets to learn the state-dependent policies for the SDDPHF.

At each decision epoch, a state of the MDP for the SDDPHF includes the requesting customer (and thus its distance from the depot), the next time that each drone is available given the requests currently assigned to drones, and the times at which each of the vehicles will return to the depot given its current route. Accounting for this information in a PFA allows us to consider a much richer set of policies in contrast to the existing work that uses only a single summary feature of the state information, the distance from the depot of a new request, and thus can only explore the threshold-type policies discussed earlier.

At the same time, we can also increase the number of decisions that we are making. Currently, the existing policy determines only whether or not we use a vehicle or drone. It accepts any customer that can be feasibly served. However, there are some customers whose requests might consume significant resources. It might be better to reject those requests for same-day delivery and offer to serve them tomorrow at a discount. By increasing the number of nodes in the output layer, we can account for this additional decision. Again, adding one more output node has minimal impact on computation.

Finally, there are also opportunities to improve the convergence of the neural net. In the work in the previous section, we use standard policy gradient methods. This method updates the parameters using only the gradient of the policy. Research has shown that the convergence of the method can be greatly improved by also incorporating an estimate of the value of the policy. These methods are known as actor-critic methods [5]. Given our success in matching the results of previous research with neural nets, we believe that we have identified a method to overcome the limitations of existing work and will present the results of this effort at the conference.

References


Same-Day Delivery with Drone Resupply

Martin Savelsbergh, Iman Dayarian
School of Industrial & Systems Engineering,
Georgia Institute of Technology, Atlanta, USA

John-Paul Clarke
School of Aerospace Engineering,
Georgia Institute of Technology, Atlanta, USA
Corresponding author: martin.savelsbergh@isye.gatech.edu

1 Introduction

Large retailers and courier delivery companies are already considering home delivery services partially or fully performed by drones. In the last few years, several papers on models and algorithms to support the operational use of drones have appeared (Murray and Chu, 2015; Agatz et al., 2016; Campbell et al., 2017). Two settings have been investigated: (1) drones deliver a single package on an out-and-back trip from a fulfillment center and supplement the delivery capacity provided by a regular fleet of delivery vehicles, and (2) drones deliver a single package on an out-and-back trip launched from a delivery vehicle, where the delivery vehicle carries one or more drones, also supplementing the delivery capacity provided by a regular fleet of delivery vehicles. The prevailing assumption in existing concepts for drone-based package delivery is that each package will be delivered by a single drone, and that package recipients will be adjacent to a location where the drone can land or in a location where the package may be lowered to them via a rope or similar mechanism. Drone operations will be particularly challenging in extreme urban environments where the vast majority of the windows in tall buildings are inoperable; the wind conditions at potential landing zones, on the roof-top of building or at ground-level near and between tall buildings, are outside the operating envelope of the small- to medium-sized drones that have been proposed to-date for package delivery; the so-called “urban canyons” between tall buildings create conditions for multiple conflicting surveillance signals (e.g. GPS multipath) and the loss of communication due to line-of-sight blockages; and the high population density increases the risk of fatalities if a drone should fail. Therefore, we are considering a different use of drones, namely drones that resupply delivery vehicles. We envision a delivery system in which delivery vehicles make deliveries in certain areas and are regularly resupplied by a drone from a fulfillment center.
located at the periphery of the city. Resupply can take place anywhere as long as the delivery vehicle is stationary (the drone will land on the roof of the delivery vehicle). The envisioned delivery system has the advantage that (1) it is less costly, and (2) the area in which same-day delivery service can be offered can be expanded, because drones can travel faster. We introduce the vehicle routing problem with drone resupply to represent this form of drone-assisted delivery and focus our algorithmic efforts on the special case with a single drone and a single delivery vehicle. The goal is to deliver as many orders as possible respecting a promised service time guarantee and doing so at minimum cost, where the cost is a function of the distance traveled by the drones and the distance traveled by the delivery vehicles.

Our contributions can be summarized as follows: We introduce an innovative home delivery system concept, especially designed to support a highly dynamic environment with tight service guarantees, based on the use of a drone to resupply a delivery vehicle. The insights obtained from an extensive computational study indicate that the proposed home delivery system is effective under a wide range of system parameters.

2 The vehicle routing problem with drone resupply

We assume that online shoppers dynamically place orders throughout a time horizon $T$ and these orders are subsequently dispatched from a fulfillment center and delivered to the online shoppers at their homes. We consider two sets of locations: first, the set of potential customer locations of online buyers, and second, the set of locations where a drone can resupply a delivery vehicle, where the distance between each location pair is given. The order placement rate at each customer location is given. We consider a common service time guarantee $S$, i.e., orders have to be delivered at latest $S$ time units after being placed. The speed of the drone and the delivery vehicle are $v^D$ and $v^K$, respectively, where $v^D \geq v^K$. The capacity of a delivery vehicle is assumed to be infinite, while the capacity of a drone is limited and is given by $P^D$, representing the maximum number of packages that can be carried. Finally, we assume that the transfer of packages from a drone to a delivery vehicle takes a fixed amount of time $\tau_0$.

The route of the delivery vehicle receiving the orders is re-optimized such that the service time guarantee of the as-yet undelivered orders already on the delivery vehicle as well as the orders just transferred to the delivery vehicle are met. Since drone landing and take-off are only possible when the vehicle is stationary, two package transfer options are conceivable: (1) Stationary transfer in which the vehicle remains at the meeting location until the transfer of packages has been completed, and (2) In-motion transfer in which the vehicle immediately departs for its next destination after a drone lands on its roof, and package transfer takes place while the vehicle is in motion: the drone takes off for the
fulfillment center at the first stop after the transfer of packages has been completed.

Dispatching a delivery vehicle involves decisions regarding the dispatch time of the vehicle, the subset of orders to load on the vehicle, and the sequence in which to deliver these orders. Dispatching a drone involves decisions regarding the dispatch time of the drone, the subset of orders to load on the drone, the meeting location of the drone and the delivery vehicle, and the sequence in which to deliver the orders on the vehicle after resupply has taken place (i.e., remaining and transferred orders).

3 Solution approach

We consider two drone resupply strategies:

**Restricted-resupply**: In this strategy, a drone dispatch from the fulfillment center is planned in such a way that a meeting with the vehicle takes place at the end of the active delivery route, once all on-board orders are delivered. The vehicle is first dispatched shortly after the start of the planning period, carrying a subset of orders placed before the vehicle’s dispatch time. It will return to the fulfillment center only at the end of the planning period. In between, the vehicle may be resupplied multiple times by the drone.

Vehicle routing decisions are made at the first vehicle dispatch from the fulfillment center, and every time the vehicle is resupplied with new packages by the drone. In both cases, decisions regarding the subset of active orders to be loaded on the vehicle as well as the routing decisions are made concurrently. Since no late delivery is allowed, the largest subset of orders which are guaranteed to be delivered on time are loaded on the vehicle. Since no waiting time is allowed for the drone at a meeting location, the drone dispatch time at the fulfillment center is scheduled so that its arrival at the meeting location is guaranteed to be greater than or equal to the vehicle’s arrival time. The drone can be dispatched only when it is at the fulfillment center. After delivering all on-board orders and receiving instructions from central dispatch, the vehicle moves to a chosen meeting location. The chosen meeting location will depend on the location of the last order on the vehicle’s active route as well as on the set of active orders at the fulfillment center. The goal is to choose the meeting location which allows the largest number of orders to be delivered on time.

**Flexible-resupply**: In the flexible-resupply strategy the meeting between a drone and a vehicle does not necessarily take place after delivering the last order on the vehicle. Instead, the vehicle may be instructed to perform a detour at some point along its current route towards a meeting location where it can meet with a drone and receive new orders to deliver. A meeting before the end of vehicle’s current route may result in rerouting the vehicle to deliver the set of remaining on-board orders as well as the resupplied active orders by the drone, altogether constituting the new set of on-board orders. Similar
to the restricted-resupply strategy, decisions are made upon the drone’s return to the fulfillment center. At a decision time, the set of admissible drone resupply options are identified. An admissible drone resupply option is the one can still be reached by the vehicle along its route without violating due time of on-board orders. For each admissible drone resupply options, a route is generated that delivers the largest subset of active orders at the fulfillment center and all remaining orders on-board of the delivery vehicle. The meeting location and time resulting in the maximum number of deliveries is selected.

A rolling horizon based routing and scheduling meta-heuristic is designed and implemented for the solution of instances of the vehicle routing problem with drone resupply.

4 Computational Results

The results from a computational study show, as expected, that the flexible-resupply strategy performs better than the restricted-resupply strategy, and that in-motion package transfer allows the on-time delivery of a larger number of orders. The results also show that when the service guarantee is aggressive, the drone capacity has a relatively small impact on the number of packages delivered. However, when the service guarantee is less aggressive, drone capacity does matter.

Most importantly, the results show that there are significant benefits to employing drone resupply strategies. The reasons for this are: (1) The use of drone resupply allows more efficient use of the delivery vehicle, which results in a significant increase in the percentage of orders served. (2) The use of drone resupply allows for expansion of the service area.

References


Optimizing the vessel fleet size and mix to perform maintenance at offshore wind farms

Kamilla Bolstad, Manu Joshi, Magnus Stålhane
Department of Industrial Economics and Technology Management
Norwegian University of Science and Technology, Trondheim, Norway
Email: magnus.staalhane@ntnu.no

Lars-Magnus Hvattum
Faculty of Logistics,
Molde University College, Molde, Norway

1 Introduction

The world energy demand is increasing, and the EU has set ambitious targets for energy consumption from renewable sources by 2020 [1]. As a green and renewable energy source, power generated from offshore wind plays an important part in reaching these targets. However, at present offshore wind projects are considered to be around 50% more expensive than onshore wind projects. Due to the high expenses, offshore wind is still not profitable on its own and depends on governmental subsidies. Operations and maintenance (O&M) costs account for a substantial amount of the expenses, and is estimated to be around 20 - 25% of the total lifetime cost of an offshore wind farm [2].

The main cost components of maintenance operations are related to the costs of acquiring and operating a vessel fleet to conduct maintenance, and the loss of revenue incurred when turbines are shut down due to failures or maintenance execution (downtime cost). In general, studies show that the cost of acquiring and operating a vessel fleet accounts for up to 73% of O&M costs [3], while the downtime cost accounts for up to 66% of O&M costs [4]. The high maintenance cost of offshore wind is to a large extent caused by the rough weather conditions offshore, which makes the turbines more exposed to breakdowns and more difficult to access. The operational capabilities of the vessels used to conduct maintenance is therefore important, as the vessels ability to handle harsh weather conditions directly influence the downtime cost of the wind farm. In order to make offshore wind power profitable and viable without governmental subsidies, finding the vessel fleet that minimize the O&M costs is hence of crucial importance.

When composing a vessel fleet for conducting maintenance, different strategic decisions
need to be made. First, a choice of which vessel concepts to acquire must be made. The characteristics of the different vessel concepts in the fleet affect the accessibility and travel time to the wind farm, and the ability to perform maintenance tasks. Second, a choice regarding the method and timing of acquisition needs to be considered. Different charter contracts have different costs, and the charter rates of vessels can vary significantly from year to year, and between vessel concepts. Finally, the decision of where to locate the maintenance vessels also needs to be considered. Locating vessels offshore in close proximity to the wind farms can reduce travel times and increase farm accessibility significantly. However, offshore stations are expensive to install and maintain. To complicate further, the strategic fleet size and mix decisions place restrictions on the tactical decisions of how to deploy the fleet to conduct maintenance. In order to evaluate the quality of fleet size and mix decisions, the expected downtime cost of the wind farm for the given fleet must be calculated.

Both the strategic fleet size and mix decisions and the tactical deployment decisions are subject to a wide range of uncertainties. At the strategic level, examples of such uncertainties are: long-term trends in electricity prices, the level of future governmental subsidies, the introduction of new vessel concepts in the market, and whether or not new turbines will be added to a wind farm in the future. At the tactical level, uncertainty in the weather conditions and the demand for maintenance influence the decision of how to utilize the fleet.

Offshore wind is a relatively young industry, and the number of publications studying the fleet size and mix problem for this industry is limited. Previous work on the fleet size and mix problem for offshore wind farms either handled the tactical uncertainty in a very simplified manner [6], or disregarded uncertainty at the strategic level [7], [8].

2 Problem Structure and solution method

We present a new approach to the strategic fleet size and mix problem in offshore wind, that considers uncertainty at both the strategic and the tactical planning level. This is challenging due to combination of different time scales: while the strategic decisions of fleet size and mix have time horizons of many years, the tactical deployment decisions are made on a daily or weekly basis. In addition, decisions on both planning levels are subject to uncertainty.

We have modelled the problem based on the dual-level scenario tree presented in [9]. The general structure of the scenario tree used is depicted in Figure 1, where the strategic nodes represent points in time where strategic decisions are made. The strategic nodes are modelled with a set of embedded tactical scenarios for each season. The seasons have been introduced in order to capture seasonal differences in aspects like weather conditions,
chart rates and electricity prices. Each tactical scenario contains a set of periods which
denotes the time interval between two consecutive time-discretisation points. Tactical
decisions are made in each period.

Figure 1: Problem structure of the DLPOW, inspired by [9].

The strategic decisions are how many vessels of each type to long-term charter, where to
locate the vessels, and how long the chartering periods should be. For each season related
to a strategic node, a decision on the number of vessels to short-term charter in and out
in each season must be made. These decisions are made with the same information as in
the strategic nodes, and can hence be seen as strategic decisions. In the tactical scenarios,
a recourse of fleet size and mix the decision finds the expected downtime cost of deploying
the fleet. It is assumed that the strategic uncertainty realized between the strategic nodes
are independent of the tactical uncertainty realized in the tactical scenarios.

In order to solve real-life instances of the dual level fleet size and mix problem within a
reasonable amount time, a metaheuristic has been developed using a Greedy Randomized
Adaptive Search Procedure (GRASP) to construct a set of strategic decisions, while an
embedded greedy tactical heuristic is used to evaluate the cost of the given fleet in each
node. To the authors knowledge, this is the first application of GRASP on a dual-level
stochastic model, and possibly the second application on a stochastic model in general [5].

3 Computational results

Extensive testing has been conducted of the GRASP heuristic. The performance of the
GRASP has been evaluated by comparing solution time and quality to the equivalent
values obtained from a standard optimization solver. Test results show that the GRASP
consistently provides good solutions to the problem, and compared to the solver, the
GRASP finds a better fleet size and mix solution for 24 of the 29 tested instances. In
addition, the GRASP manages to solve significantly larger instances than the optimization solver, within a considerable shorter amount of time.

References


Combined Fleet Deployment and Inventory Management with Flexibility in Port Choice

Bo Dong, Marielle Christiansen, Kjetil Fagerholt
Department of Industrial Economics and Technology Management,
Norwegian University of Science and Technology, Norway
Email: bo.dong@ntnu.no

Tolga Bektaş
Southampton Business School,
University of Southampton, UK

1 Introduction

In this article we study a fleet deployment problem with the treatment of inventory in the roll-on-roll-off (Ro-Ro) segment of liner shipping. Liner shipping is one of the three major transportation modes in maritime transportation and resembles a bus service in land transportation where the ships services a given set voyages along trades according to a published schedule. The Ro-Ro segment of liner shipping transports cars, trucks and other types of cargoes that can be placed on trolleys for loading and unloading from a ship without the help of external machinery like cranes.

The main tactical level decisions of a liner shipping problem is that of fleet deployment, which assigns fleet of ships to a number of given voyages that the company must service to minimize the costs over a planning horizon of typically a few months up to a year. Inventory management at ports decides the quantities to transport to/from the other ports so that the port inventory levels are kept within given limits. There are a few examples in the literature for maritime inventory routing problems, though mostly in the industrial and tramp shipping (see for example [1]).

The purpose of this work is to present a mathematical model for the combined Maritime Fleet Deployment and Inventory Management Problem (MFDIMP). There exist some references to the fleet deployment combined with inventory management in Ro-Ro shipping, such as [2, 3]. However, in these study, an unnecessary restriction (even though it is based on current practice) of visiting all ports on every voyage is imposed. Therefore, in this paper, a new version of this problem is formulated where the inherent planning flexibility of port choice is utilized. Some ports along the trade are allowed to be skipped
for certain voyages if there is no need to visit. The inclusion of this planning flexibility might improve efficiency and result to substantial financial savings, but would also complicate the problem further. The pre-generation of all possible configurations is applied to each trade by choosing different ordered sets of ports on that trade. Then different rules of generating a subset of configurations will be discussed in order to solve the problem more efficiently. Computational results will be presented to compare the different configurations generation rules and to quantify the benefits obtained by the port choice flexibility.

2 Problem Description

A Ro-Ro shipping company operates a fleet of heterogeneous ships with different capacities, speeds and bunker consumption profiles. All ships have their own initial positions and they might have to wait a certain amount of time before they can start to service new voyages.

A trade consisting of a number of ports in a given sequence is a geographical network used to transport cargo. Without loss of generality, we assume that all ports in a trade are indexed in an increasing order in which they appear in the sequence geographically. For example, Fig.1 shows two trades, namely Trade 1 and Trade 2, as well as their associated ports (loading ports are given by shaded circles and unloading ports are given by non-shaded circles). The shipping company operates on a number of trades. Due to contractual requirements and demand, each trade must be serviced regularly, e.g. on a weekly basis. After a ship has completed one voyage on a trade, it may serve another voyage on the same trade or another one or end its service. The ship often needs to reposition itself without load to be ready for the next voyage.

![Figure 1: Two trades with the possible configurations.](image)

Some ports can be allowed skipped during a particular voyage depending on demand. A configuration of a trade is defined as an ordered set of ports that are chosen to be visited. Fig.1 also shows all the possible configurations for the two trades. In particular, Trade 1 only has one configuration \{1,2\}, whereas Trade 2 has three configuration \{3,5\}, \{4,5\}, \{3,4,5\}. This means that we assume that it is not allowed to visit the ports in a sequence that deviates from the ports’ natural ordering along the trade as it would increase the sailing distance, e.g. configuration \{4,3,5\} is not allowed.
Fig. 2 shows an example of routes for two ships, one with no port choice flexibility [3], and the other with port choice flexibility. In the top half of Fig. 2, both two ships sail all ports on a trade for all voyages. In the bottom half of Fig. 2, for example, ship 1 sails Voyage 1 on Trade 1 with configuration \{1,2\}, Voyage 1 on Trade 2 with configuration \{4,5\}, Voyage 3 on Trade 1 with configuration \{1,2\}. The configuration nodes chosen for a voyage is indicated by the filled-in color.

We assume deterministic sailing times for all ships between ports and that the time spent at each port is given regardless of the quantities loaded/unloaded. Thus, each voyage has an estimated duration depending on which configuration is chosen. This duration includes the sailing time between all visited ports and the time spent in these ports. Even though each trade should be serviced according to a given frequency, there is some flexibility when each voyage must start, given by an associated time window. A time window associated with each voyage, defined by an earliest and latest start time, defines within what time interval the service in the first port of the voyage must start.

Servicing a voyage incurs costs such as port and fuel consumption costs, depending on the ship type and the configuration chosen.

A number of products are transported along the trades. Each product has given pro-
duction rates at its associated loading ports and given consumption rates at its associated unloading ports, which is assumed constant during the planning horizon. Each ship is able to carry a number of different products at the same time. Similarly, each port has a given storage capacity. Cargoes are rarely transshipped in Ro-Ro shipping, so transshipment is disregarded.

The objective is to choose a service configuration for each voyage, assign the ships in the fleet to the voyages, and to decide the quantities of each product to be loaded/unloaded at each port along each voyage, so that total transportation costs are minimized. Furthermore, we must make sure that all planned voyages are serviced within their time windows, the aggregate inventory limits of all products at each port are kept within their maximum and minimum limits, and that there is not backlogging of demand for any product in any of the ports.

3 Preliminary Results and Conclusion

The problem is originally formulated as a nonlinear mixed-integer linear program which is then linearized. For small-sized instances, all possible configurations for all trades are pre-generated, which is then input into the model. For larger-sized instances, different rules of generating a subset of configurations that are likely to result in good quality results will be presented. Using a commercial optimizer Xpress to solve the model, computational results will be presented to compare the different configurations generation rules and to quantify the benefits obtained by the port choice flexibility.

References


1 Introduction and Problem Definition

We study an Industrial and Tramp Ship Routing and Scheduling Problem (ITSRSP) where the speed on each sailing leg (route segment) is a decision variable, and fuel consumption is a convex function of speed and payload [4]. Routing models often assume a constant speed, resulting in several limitations. For example, different speeds can significantly change the travel cost and time-window feasibility of a solution. Therefore, without speed as a decision variable, it is not possible to efficiently explore the trade-off between routing decisions and fuel consumption. This is more evident in ship routing scenarios, as shipping operators may engage in slow steaming practices to reduce fuel costs [5]. The influence of payload on fuel consumption extends previous work [3], and results in leg-dependent fuel consumption functions, as the load on board of the ship changes at every port visit. We refer to the studied problem as the ITSRSP with Speed Optimization (ITSRSPSO). It can be viewed as an extension of the Pickup and Delivery Problem with Time Windows with a heterogeneous fleet, ship-cargo compatibility constraints, different ship starting points and starting times, service flexibility with penalties, and speed optimization.

We now define the ITSRSPSO. A tramp or industrial shipping operator has a fleet of $m$ ships. Let $G = (N, A)$ be a complete directed graph where $N$ is the union of a
set of pickup nodes \( P = \{1, \ldots, n\} \), delivery nodes \( D = \{n + 1, \ldots, 2n\} \), and starting points \( \{0_1, \ldots, 0_m\} \). There are \( n \) cargoes available. Each cargo \( i \in \{1, \ldots, n\} \) consists of transporting a load of size \( q_i \) from a pickup \( i \in P \) to a corresponding delivery location \( n + i \in D \). Every node \( i \in P \cup D \) is associated with a time window of allowable visit times \([a_i, b_i]\). A ship \( k \in \{1, \ldots, m\} \) is initially located at \( 0_k \), has capacity \( Q_k \), and is first available at time \( s_{0k} \). It can traverse an arc \((i, j) \in A \) at any speed \( v \in [v_{k_{\text{min}}}, v_{k_{\text{max}}}] \) with a cost \( f_{ij}^k(v, u) = \frac{1}{24} \mu_k v^2 e_{ij}^k (0.8 + 0.2u) \), where \( \mu_k \) is a ship-specific constant, \( e_{ij}^k \) is the arc length, and \( u \) is the proportion of maximum load on board the ship before arriving node \( j \).

For every combination of ship \( k \) and node \( i \in P \cup D \), there is an associated service (port) cost \( s_{ik}^c \) and duration \( s_{ik}^d \) for visiting \( i \). There might be incompatibilities between ships and cargoes. A penalty \( p_i \) is paid if a cargo \( i \) is not transported by the fleet. The objective of the ITSRSPSO is to minimize total travel cost plus the associated penalties in the case where some cargoes are not serviced.

2 Solution Methodology

To solve the ITSRSPSO, we propose a Hybrid Genetic Search (HGS), a non-trivial extension of the Unified Hybrid Genetic Search of [6]. The proposed HGS includes a set-partitioning-based large neighborhood and uses problem-tailored crossover and local search operators to cover multiple ITSRSPSO attributes which were not included in the original framework. Being a hybrid metaheuristic, the HGS combines the exploration capabilities of genetic algorithms with efficient local search improvement procedures. The search space of the HGS considers penalized infeasible solutions to enhance the search towards high-quality feasible solutions, penalizing the maximum load violation, time-window infeasibility (using “returns in time” for constant-time evaluations of infeasible routes [7]), and number of incompatible cargoes. Several components of the HGS ensure a balance between solution quality and population diversity, and the local search procedure also employs granular search techniques to reduce the size of the neighborhoods.

The speed of each leg is jointly optimized on every local search move evaluation. If a neighbor solution is time-window infeasible, the move is evaluated in \( \mathcal{O}(1) \) using preprocessed information and route concatenation techniques. For this scenario, travel cost is obtained from the fuel consumption functions assuming maximum ship speed and full load. Otherwise, if a neighbor solution is time-window feasible, the optimal speed for each leg is found by solving a Resource Allocation Problem with Nested Constraints (RAP-NC) using the \( \mathcal{O}(n \log m \log \frac{nB}{\epsilon}) \) Monotonic Decomposition Algorithm (MDA) of [8]. Ships have a minimum and maximum cruising speed. Therefore, they may wait on an early arrival due to a combination of speed bounds and time windows. This leads to fuel consumption functions that are not strictly-convex (if \( v < v_{k_{\text{min}}} \), \( f_{ij}^k(v, u) = f_{ij}^k(v_{k_{\text{min}}}, u) \)) in the RAP-NC.
formulation. Fortunately, the MDA does not require strict convexity and therefore is able to account for waiting times.

3 Computational Experiments and Conclusions

We implemented the HGS in C++ and conducted our experiments on an i7-3960X CPU. We compared a version of HGS without speed optimization (HGS-W) with several Adaptive Large Neighborhood Search (ALNS) variants from the literature on a set of 240 ITSRSP benchmark instances based on real-life scenarios [1, 2]. Then, we adapted these instances to evaluate the impact of joint speed optimization. We compared our HGS with joint speed optimization (HGS-J) with a version of HGS that assumes a fixed speed and optimizes speed once at the end (HGS-O). Comparison with previous literature is shown in Table 1 and Table 2, while speed optimization results are shown in Table 2. Table 1 provides summary results over a subset of the instances evaluated by [1], and the results in Table 2 are grouped according to problem topology (short sea, deep sea) and cargo type (mixed load, full load). Average values for gap (in percentage) to previously best known solution and CPU time (in minutes) are respectively given by “Gap” and “T”.

<table>
<thead>
<tr>
<th>ALNS-1</th>
<th>ALNS-2</th>
<th>ALNS-3</th>
<th>ALNS-4</th>
<th>ALNS-5</th>
<th>ALNS-6</th>
<th>HGS-W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gap</td>
<td>1.11</td>
<td>0.81</td>
<td>0.98</td>
<td>0.84</td>
<td>0.79</td>
<td>1.14</td>
</tr>
<tr>
<td>T</td>
<td>0.68</td>
<td>0.75</td>
<td>0.76</td>
<td>0.78</td>
<td>0.68</td>
<td>0.64</td>
</tr>
</tbody>
</table>

Table 2: Comparison with [2] results and evaluation of the impact of joint speed optimization.

<table>
<thead>
<tr>
<th>Group</th>
<th>ALNS</th>
<th>HGS-W</th>
<th>HGS-O</th>
<th>HGS-J</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gap</td>
<td>T</td>
<td>Gap</td>
<td>T</td>
</tr>
<tr>
<td>SS_MUN</td>
<td>0.87</td>
<td>2.71</td>
<td>-0.48</td>
<td>1.75</td>
</tr>
<tr>
<td>SS_FUN</td>
<td>0.23</td>
<td>1.93</td>
<td>-0.19</td>
<td>0.51</td>
</tr>
<tr>
<td>DS_MUN</td>
<td>1.13</td>
<td>2.77</td>
<td>-0.65</td>
<td>2.26</td>
</tr>
<tr>
<td>DS_FUN</td>
<td>0.29</td>
<td>1.93</td>
<td>-0.12</td>
<td>0.55</td>
</tr>
<tr>
<td>Avg</td>
<td>0.63</td>
<td>2.33</td>
<td>-0.36</td>
<td>1.27</td>
</tr>
</tbody>
</table>

The HGS-W significantly outperforms all ALNS variants from previous literature, becoming the new state-of-the-art heuristic for the ITSRSP. The average solution quality of HGS-J is largely better than HGS-O, with a moderate increase (6.1×) of computational
effort. Given the complexity of a tight integration of speed optimization and routing decisions in a metaheuristic, the observed CPU time is very satisfying. Overall, this work demonstrates that a joint speed and routing optimization is highly profitable from an operational cost perspective, and is made possible without too much overhead when using an efficient linearithmic algorithm such as MDA.

References


Liner shipping service scheduling and cargo allocation

David Franz Koza
Department of Management Engineering, Technical University of Denmark
Email: dakoz@dtu.dk

1 Introduction

We address two problems that affect operational costs as well as the service level of a liner shipping company: The service scheduling problem and the cargo allocation problem.

The service scheduling problem aims at determining optimal port call times of liner shipping services. It is a generalization of the simpler speed optimization problem. Different to previous studies (e.g. Guericke and Tierney, 2015) it does not only determine the speed and fuel consumption between subsequent port calls, but also the weekly berth times at each port. In liner shipping networks the schedules of individual liner shipping services establish the transshipment times between pairs of services that call the same port (Figure 1).

The second problem we address is the cargo allocation problem (CAP) or, synonymously, cargo routing problem. The CAP tries to answer the question of how to route containers through a liner shipping network between their origin and destination.

We solve the service scheduling and the cargo allocation problem together to account for the dependencies between both problems. The objective is to minimize total fuel consumption costs minus total revenues from transporting containers while satisfying transit time requirements for each demand. We call the combined problem the liner shipping

![Figure 1: Illustration of schedule dependent transshipment times between two services. The example shows the timeline of a port that is called once per week by each of two different services (red and blue arrows).](image-url)
service scheduling and cargo allocation problem (LSSCAP).

The contribution of this work is twofold: First, we present a model and solution method that integrates the service scheduling problem and the cargo allocation problem; the synchronization of individual services and resulting transshipment times are considered explicitly. Second, we consider the vessels’ payload in the fuel consumption function and show that neglecting payload may result in suboptimal speed profiles and schedules.

2 Problem statement

We consider a given set of services. Each service is defined by a sequence of port calls that define the sailing route of the service and a total duration in weeks, which is equal to the number of vessels operating the (weekly) service. Multiple port calls for the same port may exist. The number of sailing legs on each service equals the number of port calls. We are further given a set of all origin-destination demands defined by a number of containers and a unit revenue for each container if transported. Furthermore, a transit time limit defining a maximum duration between pick-up and delivery is given for every demand.

The goal is to simultaneously find a feasible schedule for each service and to allocate cargo to the services such that the total fuel consumption cost minus the revenues from transporting cargo are minimized. A schedule is feasible if the total round trip time (in weeks) equals the deployed number of vessels and if the speed restrictions of the deployed vessel class are satisfied on each sailing leg. A cargo path is only feasible, if the total transit time including transshipment times at ports (if applicable) does not exceed the maximum transit time of the corresponding demand.

3 Model and solution method

We model the problem over a time-space graph as illustrated in Figure 2. Each vertex represents a port call of a particular service at a particular time. Vertices are connected by directed sailing arcs, if the port call of the destination vertex follows the port call of the origin vertex for a given service, and if the sailing time associated with the arc is feasible. Vertices that represent the same port are connected by transshipment arcs that represent movement in time only. If a cargo path uses a transshipment arc, the containers are transshipped between two different services (vessels) at the corresponding port.

We present a branch-and-price algorithm with an integrated primal heuristic to solve the problem. The pricing problem is a resource constrained shortest path problem and we use a label setting algorithm to solve it. The primal heuristic is invoked at each node in the branch-and-bound tree (if not pruned). It first determines a feasible schedule for all services in the network by solving a binary integer programming model. In the second step
the optimal cargo allocation is determined by solving a multi-commodity flow problem. The primal heuristic generates a feasible solution to the LSSCAP at each node if invoked.

4 Results and conclusion

The branch-and-price solution algorithm is tested on four (unscheduled) liner shipping networks of different size and different transit time restrictions on the origin-destination demands. For small and medium sized networks, the found solutions are close to optimal (< 0.5% gap). For large problem instances larger gaps remain; the integrated primal heuristic, however, quickly finds feasible solutions.

The results show that the approximation of transshipment times (as used in previous studies) can lead to over- or underestimation of cargo transit times. If demands are subject to transit time limits, the underestimation of cargo transit times results in overly optimistic solutions in terms of transported cargo and revenues. Figure 3 compares solution key performance indicators (KPI) for different transshipment time approximation values (48h,
(a) Number of containers transported

(b) Number of transshipped containers transported

(c) Revenue

Figure 3: Comparison of constant transshipment time values against exact transshipment times and how they influence solution KPIs in comparison to the exact model (in %).

Figure 4: Sailing speed per leg in the best found schedules for a network of 20 ports and 11 services. The bar plot is divided by services and the x-axis labels denote the service.

We further analyse the impact of payload on the fuel consumption function and compare results against the case in which payload is neglected, as done in the majority of liner shipping studies. Figure 4 compares the optimal speed on different sailing legs under a simple speed-only dependent fuel consumption function and under a payload dependent fuel consumption function. The results show that the optimal speed per leg may vary between the two fuel consumption functions. Thus, neglecting vessel payloads may not only lead to miscalculated costs, but it may actually result in suboptimal speed profiles.

References

A rich vehicle routing problem for pharmaceutical
distribution

Raphael Kramer
Department of Sciences and Methods for Engineering
University of Modena and Reggio Emilia, Italy
Email: raphael.kramer@unimore.it

Jean-François Cordeau
Department of Logistics and Operations Management
HEC Montréal, Canada

Manuel Iori
Department of Sciences and Methods for Engineering
University of Modena and Reggio Emilia, Italy

1 Introduction

We present and solve a rich vehicle routing problem (RVRP) based on a practical distribution problem faced by an Italian company whose aim is to deliver pharmaceutical products to healthcare facilities in Tuscany. The problem is characterized by having multiple depots, a heterogeneous fleet of vehicles, customer time-windows, time-windows flexibility, periodic demands, incompatibilities between vehicles and customers, a maximum duration for the routes, and a maximum number of customers that can be served per route. An iterated local search (ILS) algorithm enhanced with auxiliary data structures from the literature is proposed to solve the problem. The ILS has been tested on a large number of instances and obtained good results, both on the real RVRP and on a number of VRP variants encountered in the literature.

2 Problem Description

Our RVRP is defined over a graph $G = (N, A)$, where $N$ is the set of nodes and $A = \{(i, j) : i, j \in N\}$ is the set of arcs. $N$ is partitioned in depot nodes ($D$) and customer nodes ($C$). Nodes $i \in D$ can be classified as main depot ($M \subseteq D$) or auxiliary depot ($A \subseteq D$), while nodes $i \in C$ can be classified as hospitals ($H \subseteq C$) or small customers ($S \subseteq C$).
Customers have demands to be attended periodically, with \( q_{ip} \) representing the demand of a customer \( i \in C \) in period \( p \in P \), where \( P \) is the set of periods (i.e., days in our study case). A heterogeneous fleet of vehicles, formalized as a set \( V \) of vehicle types, can be used to perform the deliveries, with \( Q_v \) and \( h_v \) representing, respectively, the capacity and the cost per Km of vehicle type \( v \in V \). In addition, every node \( i \in N \) has a time-window \([e_i, l_i]\) – homogeneous for nodes of the same type –, representing the earliest \((e_i)\) and latest \((l_i)\) times for a vehicle arrival. If \( i \) is a depot, \( e_i \) represents the earliest departure time.

There are two main differences regarding depot types: (i) main depots have time-windows tighter than those for auxiliary depots, with \( e_i \leq e_j \leq l_j \leq l_i \), for \( i \in A \) and \( j \in M \); and (ii) the products to be delivered in period \( p \) by vehicles departing from auxiliary depots have to be supplied in period \( p - 1 \) by a vehicle departing from the closest main depot. It is assumed that a vehicle of capacity \( Q_0 \) is used to perform this supply, in an exclusive route between main and auxiliary depots, with \( \omega_k \) representing the round trip cost associated with the supply of an auxiliary depot \( k \in A \). Equivalently, (i) hospitals have tighter time-windows than small customers, with \( e_i = e_j \) and \( l_j \leq l_i \), for \( i \in S \) and \( j \in H \); and (ii) differ from small customers in the possibility of accepting anticipated deliveries, by using warehouses to stock the products. It is assumed that hospital deliveries can be anticipated for at most one period, implying an additional cost \( \gamma \) for each anticipated deliver.

By denoting \( R_p \) the set of routes in period \( p \), \( d_{ij} \) the distance from \( i \) to \( j \), \( L_{kp} \) the total demand to be delivered in period \( p \) by vehicles departing from the auxiliary depot \( k \), and \( y_p \) the number of anticipated deliveries in period \( p \), the cost of a solution \( s \) can be computed as

\[
f(s) = \sum_{p \in P} \sum_{\sigma \in R_p} \sum_{i=1}^{\lfloor \sigma \rfloor - 1} h_v(\sigma) d_{\sigma_i \sigma_{i+1}} + \sum_{p \in P} \sum_{k \in A} \omega_k \left\lceil \frac{L_{kp}}{Q_0} \right\rceil + \sum_{p \in P} \gamma y_p,
\]

where \( v(\sigma) \) represents the vehicle type associated with a route \( \sigma \), and \( \sigma_i \) represents the \( i \)-th customer visited in route \( \sigma \). In this way, the problem consists in finding a solution \( s \) to deliver pharmaceutical goods to healthcare establishments at each period \( p \in P \) aiming at the minimization of \((1)\), respecting the customers (and depots) time-windows, the vehicle capacities, the customer-vehicles incompatibilities, the maximum duration of routes, and the maximum number of visits per route. Following the VRP notation, this problem can be classified as a Multi-Depot Fleet Size and Mix, Periodic and Site-Dependent Vehicle Routing Problem with Time Windows.
3 Solution method

A multi-start iterated local search (MS-ILS) algorithm is adopted to solve the RVRP. The general ILS framework is characterized by three main components: (i) constructive procedure, (ii) local search (LS), and (iii) perturbation (see, e.g., [1]). The proposed MS-ILS is an ILS that is restarted \( \eta_{\text{iter}} \) times. For each restart, the loop of LS and perturbation is repeated for several times, but stopped if \( \eta_{\text{ils}} \) consecutive iterations without improving the incumbent solution occur. To speed-up the local search and improve the convergence towards good solutions, the proposed algorithm makes use of the auxiliary data structures proposed by [2] and accepts infeasible solutions with respect to time-windows and route durations (but it penalizes them in the objective function).

An initial solution is created iteratively, for each period \( p \in \mathcal{P} \), according to a greedy strategy. The initial routes of day \( d \) are built as follows. For each hospital \( i \in \mathcal{H} \), we create a route containing only \( i \), and associate the route to the closest depot of \( i \) and to the largest vehicle allowed to visit it. Then, a customer \( j \in \mathcal{C} \) with \( q_{ip} > 0 \) is selected at random and inserted in an existing route, in such a way that the increasing cost is minimized. In this phase, time-windows violations are allowed, but violation on routes durations are not. If \( j \) cannot be inserted in any of the existing routes, a new route associated with the closest depot to \( j \) and with the largest vehicle allowed to visit \( j \) is created. The procedure repeats until all customers have been assigned to a route.

The LS procedure is based on a variable neighborhood descent with random neighborhood ordering (see, e.g., [3] and [4]). Seven neighborhoods based on elementary moves were implemented: swap inter-routes; swap intra-route; relocate inter-route; relocate intra-route; 2-opt; 2-opt*, and hospital relocate inter-periods.

The perturbation procedure is composed by two independent methods, but, according to a random choice, only one is executed. The first method executes random swap moves involving two routes. The second one randomly selects a route starting from an auxiliary depot, and imposes it to start from a main depot instead – the one leading to the lowest cost. These two procedures can generate solutions that cannot be obtained by any of the elementary moves considered in the LS, thus helping the algorithm to escape from local optima.

4 Preliminary Computational Experiments

The proposed MS-ILS was coded in C++ and executed on a single thread of an Intel Core i5-5200U 2.2GHz with 16GB of RAM, running under Linux Mint 17.2 64-bit. The following parameter values were adopted: \( \eta_{\text{iter}} = 20 \), and \( \eta_{\text{ils}} = 20 \).

Preliminary computational experiments have been performed on three realistic instances from three areas of Tuscany: center, northwest, and southeast, containing 232,
159, and 161 nodes, respectively. The demands are distributed over 6 periods (from Monday to Saturday). The results are reported in Table 1. The first six columns report the characteristics of the instances; columns cost, #antic., |R|, and time(s) report the solution cost, the number of anticipated deliveries, the number of routes, and the computational time (in seconds), respectively. By comparing the best MS-ILS results with those obtained by the greedy algorithm, it can be verified that the neighborhoods and perturbation procedures have a very significant contribution on the quality of the solution, leading to an improvement of about 25%. Moreover, the small differences between average and best solutions show the robustness of the algorithm.

<table>
<thead>
<tr>
<th>Table 1: MS-ILS results for realistic instances (10 runs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>Center</td>
</tr>
<tr>
<td>Northwest</td>
</tr>
<tr>
<td>Southeast</td>
</tr>
</tbody>
</table>

Additional computational experiments have been performed to evaluate the importance of the anticipated deliveries and the auxiliary depots in reducing the transportation costs. Artificial instances ranging from 100 to 300 nodes and 6 periods were adopted, each of them containing up to 4 main depots and 4 auxiliary depots. The obtained results show an average reduction of about 1% on average by using anticipated deliveries, and of up to 18% by using auxiliary depots. The developed MS-ILS also obtained interesting results on a number of VRP variants from the literature.

References


1 Introduction

Concrete is one of the most important raw materials in the construction industry. It is commonly used in infrastructure works and buildings, where it is used from foundations to roofs. Concrete is manufactured in dedicated sites we will call hereinafter \textit{plants}. Concrete is a fast perishable good that must be produced just in time to meet customer demand requests, since it is not possible to store the concrete beforehand.

The problem of production planning of concrete and dispatching concrete trucks is known in the literature as the \textit{ready-mix concrete dispatching problem} (RMCDP), which is a generalization of both the vehicle routing problem and the parallel machine scheduling problem. The problem becomes more complex if the vehicles or machines are not identical. In particular, this research deals with the case of heterogeneous machines and homogeneous fleet of vehicles from an study case applied on a major provider of concrete in Santiago, Chile.

Problems associated with concrete dispatch have been studied in the literature of operations research for more than 50 years [2]. Naso et al. [6] solves the problem by separating the process into two stages: assigning orders to plants and then assigning trucks, with both steps solved in MATLAB. Yan et al. [8] integrates production and routing. Asbach et al. [1] resolves the RMCDP with heterogeneous vehicles and plants for small instances in CPLEX. Schmid et al. [7] reformulate their previous models and solve them with a hybrid approach combining a variable search algorithm and an optimization model. Hertz et al. [4] proposes a model with no time windows, homogeneous fleet,
several deposits and separated trains of orders, which is solved by means of two MIPs, one assignment and one routing. The problem considered in this work is a variant of the one discussed by Kinable et al. [5].

2 The ready-mix concrete production and dispatching problem

The production of concrete is performed in production plans, where cement, water and other components and additives are mixed up. These plants function as a loading mouth, under which a mixer or vehicle is positioned for loading. Once in the mixing vehicle, the concrete has a maximum delivery time [1] before loosing its properties and even risking to harden inside the truck. At the destination construction site, mixer trucks deliver the concrete. After this process, trucks become idle and go back to a plant to be loaded to fulfill another dispatch. When the requested amount of concrete for a construction site exceeds the capacity of a vehicle, this order is split in several synchronized deliveries. These associated deliveries corresponding to the same destination and product is called a train. Dispatches of trucks belonging to the same train should be coordinated in order to ensure almost regular headways to avoid queueing at the destination or too long times between two of them.

Although, customers, requests, resources and even the fleet, could vary from a day to another, the RMCDP we consider is a deterministic static problem solved daily by the company. Precisely, the problem consists in, given a set of customer requirements, defining the train corresponding to each requirement and simultaneously, assigning each load to an specific plant for being produced at certain moment (schedule), and specifying a ready-mixer truck for final dispatch to the corresponding site.

Our solution scheme comprises four steps as follows:

1. Step 1: Split the requests of the clients into trains. Namely, break each requirement in a set of truckloads and define a schedule for delivery considering the spacing between specific dispatches.

2. Step 2: Assign dispatches to production plants. Link each individual truckload to a production plant. We propose a GRASP heuristic [3] based on a weighted sum of cost of production and transport, plus a penalization for delays, including potential delays of other trucks scheduled later in the train.

3. Step 3: Assign each truckload (dispatch) to a specific mixer truck. We propose a GRASP heuristic based on a weighted sum of transport costs and penalizations for delays, including potential delays of other trucks in the same train and of other
requests to be served at the same plant. Whenever the resulting delay for a deliver exceeds a maximum tolerance, a new truck is inserted in the fleet.

4. Step 4: Reduce unnecessary fleet. Considering the assignment of dispatches to plants and the resulting schedule of Step 3, we try to reduce the fleet one by one, eliminating one of the trucks and reassigning the requests with the reduced fleet. While this process does not deteriorate too much the quality of the schedule, the process is repeated iteratively.

3 Results and conclusions

Scenarios were constructed with real information provided by the industrial partner. The main evaluation criterium for the operation is the \textit{compliance with delivery time windows}. Currently, the company define a symmetric time window of 1 hour centered at the delivery time asked by the costumer. We also report the compliance using symmetric 30 minutes-wide time windows.

We analyzed two configurations: the first one, mimicking the current operation, trucks are preassigned to production plants and when finishing a dispatch, they go back to the same plant for a new load. In the second configuration, we fixed the initial and final plants, but during the day, vehicles can be loaded at any plant depending on the needs of the company. Table 1 reports the percentage of delivers that comply with the time windows.

<table>
<thead>
<tr>
<th>Time window width (min)</th>
<th>Current operation</th>
<th>Optimized, preassigned trucks</th>
<th>Optimized, flexible trucks</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>82</td>
<td>94</td>
<td>99</td>
</tr>
<tr>
<td>30</td>
<td>62</td>
<td>83</td>
<td>93</td>
</tr>
</tbody>
</table>

Along with these performance improvements, also slight reductions on total costs (less than 1%) were obtained, in comparison with the actual operation. The scheme presents improvements in quality of service, also reducing the costs. The scheme provides the possibility of measuring sensitivity with respect to certain patterns and evaluating what would happen to some changes in the company’s current policies. At the current stage, we are working with the industrial partner in developing and implementing a computational decision support system wrapping the presented approach to be used in daily operation.
Acknowledgements

The authors wish to acknowledge the support from the Complex Engineering Systems Institute (CONICYT - PIA - FB0816) and CONICYT/FONDECYT/REGULAR/ N.º 1141313.

References


The Vehicle Routing Problem with Cross-docking and Capacity Smoothing Restrictions

Amalia I. Nikolopoulou
School of Business, Athens University of Economics and Business, Greece
Panagiotis P. Repoussis
School of Business, Athens University of Economics and Business, Greece
School of Business, Stevens Institute of Technology, USA
Christos D. Tarantilis
School of Business, Athens University of Economics and Business, Greece
Email: anikolop@aueb.gr

1 Introduction
In this work we study a new multi-level vehicle routing problem with an intermediate cross-docking facility and with capacity smoothing restrictions at the cross-dock. This network setting is frequently encountered in numerous practical applications in the area of freight transportation and city logistics. Focus is given on the development of a new solution framework for addressing the problem. The goal is to provide a tool that will aid the decision-making process with regards to the performance evaluation of the cross-docking strategy for various practical scenarios.

The freight consolidation from different shippers and carriers associated to some kind of coordination of operations is a common way to achieve a rationalization of the distribution activities. Consolidation usually takes place at intermediate facilities (i.e satellite platforms, warehouses, distribution centers etc.) where products are stored for some period of time before being delivered to the final destinations. Cross-docking, is a common distribution strategy where incoming shipments from several suppliers arrive at an intermediate transshipment point, the so-called cross-dock, and are then immediately transferred to outbound vehicles, without storage in between. Apparently, a challenging part under this network setting is the coordination of inbound and outbound product flows, in a way that the overall network performance is increased.

In the literature, the most well-studied problem variant that involves consolidation of product flows from multiple sources, delivery of products to multiple destinations, cross-docking and coordination of inbound and outbound flows is the so-called Vehicle Routing Problem with Cross-Docking (VRPCD) (see the works of Wen et al. [1] and Tarantilis [2], Santos et al. [3], Grangier et al. [4]). This problem setting assumes one cross-docking facility, one product type and single pairing of every supplier to every customer (one-to-one relationship).

Another common assumption is that the capacity at the cross-dock, representing the amount of products to be handled at the cross-dock (unloading, consolidation, loading) at any point in time, is unlimited. In practice, products might remain for a short period of time in the cross-dock facility before being delivered to their final destinations. This limitation has to do with the actual physical space restrictions as well as the availability of resources (e.g forklifts, workers, etc.) to perform the loading, unloading, packaging and picking activities. This limitation in the cross-dock capacity, has
an impact on how many vehicles can be served at the cross-dock at any point of time and thus, the overall performance and coordination of inbound and outbound activities is affected.

Additionally, the literature has so far considered the total length of routes as the only indicator for measuring the performance of a distribution network. In this work we identify a range of specific industry-related key performance indicators (KPIs), such as the temporary storage level indicator, the truck filling rate indicator and the total vehicle waiting time. Finally, there is a very limited number of research works that examine the effect of the distribution network characteristics and network parameters on the overall efficiency of the cross-docking strategy. We evaluate the network performance for various real-life industrial scenarios including different network characteristics and parameters (geographic distribution of node, capacity of the cross-dock, vehicle route duration, consolidation time at the cross-dock) on the cross-docking performance.

The contribution of this work is three-fold. First, a new generalized vehicle routing problem with a single cross-dock facility and capacity restrictions is introduced. Second, an AMP method is proposed for solving the problem. Key characteristic is the mechanism for identifying and selecting elite components from the reference solutions. Third, various computational experiments are performed to assess the suitability of the cross-docking strategy for various levels of network settings and parameters.

2 Problem description

The problem considered in this work involves a distribution network with a single cross-dock facility. The goal is to design least cost vehicle routes that minimize the total transportation costs. In particular, a set of transportation requests $R = \{1, \ldots, n\}$ has to be satisfied, where $n$ is the total number of requests. Each request $i \in R$ is associated with a quantity $q_i$, which has to be picked up from a pick-up node (supplier) and delivered to a delivery node (customer). Each node (i.e supplier, customer) is assigned a time window, specifying the earliest time and the latest time that the node can be serviced. In addition, each node (supplier, customer) is assigned a predetermined fixed service time $st$, which corresponds to the time required to perform loading (or unloading) operations at a given supplier (or customer). The set of transportation requests are to be fulfilled by two distinct fleets of vehicles, performing inbound (pick-up) and outbound (delivery) operations, with capacity equal to $Q_S$ and $Q_D$, respectively. The (pickup or delivery) routes are subject to the following constraints: (a) all routes start and end at the CD; (b) every pickup node is visited exactly once by a pickup vehicle; (c) every delivery node is visited exactly once by a delivery vehicle, (d) the total product quantity assigned either to a pickup (or a delivery route) may not exceed the vehicle capacity $Q_S$ (or $Q_D$); (e) the total product quantity at the cross-dock facility may not exceed the cross-dock capacity $Q$ at any time; (f) the total travel time of all related pairs of pickup and delivery routes may not exceed the maximum route duration $T$.

Regarding the synchronization between the pickup and delivery routes, the first challenging part is to ensure that the products to be loaded into a vehicle performing an outbound route have already arrived at the cross-dock and have been unloaded by an inbound vehicle. The second challenging part comes from the fact that the capacity restrictions at the cross-dock do not allow all inbound vehicles to perform simultaneously unloading activities at the moment they arrive at the cross-dock. In this case, inbound vehicles have to wait at the cross-dock before they start unloading freight, until some capacity space is freed (due to some products being loaded to outbound vehicles
to perform delivery routes). A list with the vehicles that cannot be served simultaneously at the cross-dock is kept in increasing order with respect to their arrival time at the cross-dock. Whenever some space is freed at the cross-dock, the first vehicle in the list is selected to be served. An example of the accumulated load at a cross-dock with $Q = 25$ for six time periods, is shown in Figure 1.

![Inbound load per time period](image1.png)

![Load at the cross-dock per time period](image2.png)

![Outbound load per time period](image3.png)

**Figure 1.** a) Inbound load at the cross-dock, b) load at the cross-dock, and c) outbound load at the cross-dock, per time period.

### 3 Solution Methodology

An Adaptive Memory Programming (AMP) metaheuristic algorithm coupled with local-search methods has been developed for solving the problem. AMP has produced the best-known results for major vehicle routing problems (Repoussis et al. [5]; Gounaris et al. [6]). The logic behind this framework is to identify solution characteristics frequently found in the search history and use this information to produce high-quality solutions. A critical path in the algorithmic development is the design of adaptive learning mechanisms for the identification of promising solution parts which will be used as building blocks for the generation of provisional solutions. Following the earlier work of Gounaris et al. [6] and given prior experience (Nikolopoulou et al. [7]), the proposed optimization method will be extended to adopt new machine learning mechanisms in order to manage and reconstruct more efficiently high-quality solutions while maintaining an effective interplay between diversification and intensification.

The proposed framework is equipped by a Tabu Search algorithm that employs a set of edge-exchange neighborhood structures and memory components to guide the search. In particular, an iterative improvement local-search scheme equipped with classical edge-exchange neighborhood structures (relocation, swap and 2-opt) is applied. Furthermore, all tentative moves are encoded using the Static Move Descriptor (SMD) strategy (Zachariadis and Kiranoudis [8]). The Tabu Search framework operates according to the best admissible local search move scheme. Specifically, all neighborhoods (defined by the operators described above) of the current incumbent solution are exhaustively explored, and the highest quality feasible neighboring solution is selected. To avoid an over-intensified search, the proposed framework is equipped with a diversification component based on the aspiration criteria of Tabu Search and the Attribute Based Hill Climber (Whittley and Smith [9]).

### 4 Computational Results

We initially validate the proposed algorithm on well-studied VRPCD benchmark instances. Since the existing datasets found in the literature consider unlimited capacity at the cross-dock, we
modified our algorithm accordingly, in order to have a fair comparison. The computational results indicate that the proposed method performs very well and the proposed method is competitive.

Next, a second set of computational experiments, which is the core part of our study, is performed on generated problem instances that consider limited capacity at the cross-dock. These experiments investigate the effect of the distribution network characteristics and network parameters, as well as the effect of the capacity smoothing restrictions. Initially, we study cases where customers and suppliers are located in the same area as well as cases where suppliers and customers are positioned into different geographic regions. Preliminary results indicate that the performance of the distribution strategy is largely dependent on the geographic distribution of suppliers and customers. Furthermore, we investigate the effect of the cross-dock capacity, the vehicle capacity and the tightness of the maximum route duration constraints on the total transportation costs. The computational results suggest that the capacity and route duration limitations have a strong impact on the cross-docking effectiveness. Finally, we measure the cross-dock performance based on several indicators such as the temporary storage level, the truck filling rate and the total vehicle waiting time.

Acknowledgment
This work is supported by the Research Funding Program "Action 2: Support for postdoctoral researchers with a view to promote research, 2017-2018" of the Athens University of Economics and Business (AUEB).

References
Decomposition Methods for Dynamic Load Planning and Driver Management in LTL Trucking

Hugo P. Simão
Department of Operations Research and Financial Engineering, Princeton University, Princeton, United States
Email: hpsimao@princeton.edu

Warren B. Powell
Department of Operations Research and Financial Engineering, Princeton University, Princeton, United States

1 Introduction

The routing and consolidation of freight is one of the critical components of service design in a less-than-truckload (LTL) network [2]. It involves simultaneously determining the least-cost paths from origin to destination, through transfer terminals, for all shipments in a trucking company and their consolidation in trailers moving between successive terminals in the paths. It can be done on a tactical level [3][4], usually called a static load plan, with historical freight volumes over a certain period of time, or it can be done on an operational level [1], usually called a dynamic load plan, with the freight currently present in the system and forecasts of the freight to be picked up in the next day or two. This paper presents a model for the solution of large-scale LTL dynamic load planning problems that includes constraints on the availability of company drivers and on the total amounts of freight that can be moved by rail and/or contracted drivers. Section 2 introduces the model and section 3 presents an application to a large LTL carrier in the United States and some encouraging preliminary results.

2 The model

Given a snapshot of the state of the system (shipments and drivers) at time $t$, the dynamic load planning problem is solved as a look-ahead deterministic optimization problem over a time horizon $H$, starting at time $t$. Usually too large to be solved in a single shot, the problem is decomposed into $T$ sub-problems, each over a time period of length $\tau$, so that $T \times \tau = H$. These sub-problems are solved consecutively and are linked to each other by an underlying simulation procedure.

For each optimization sub-problem spanning the time period $[t', t' + \tau)$, where $t' \in \{t, t + \tau, ..., t + (T - 1)\tau\}$, let $I$ be the set of shipments that need to be handled within that time period (we will omit time subscripts for the sake of clarity). Shipments can be actual ones
already in the system at the snapshot time \( t \), or shipments forecasted to enter the system during the optimization horizon \([t, t + H)\).

Let \( J \) be the set of trailer buckets, with a given origin and destination, that shipments can be loaded onto. Each bucket will be composed by one or more trailers, loaded at the origin and destined to the next load-to point of the shipments. Besides the origin and destination, other distinguishing attributes of these buckets are the service network type (expedited or standard), the cargo type (hazardous or not), and the transportation mode (company drivers, rail, or contracted drivers). Let \( \mathcal{K} \) be the set of different company driver types, each characterized by the driving time range of the dispatch legs in which the drivers primarily drive (for e.g., multi-day, single-day or half-day legs). Let \( \mathcal{M} \) be the set of partitions of the dispatch legs, based on their directions (say, forward or back haul). Finally, let \( \mathcal{N} = \{A, R + C, C\} \) be the set of three partitions of the transportation modes, where \( A \) represents the combination of all modes, \( R + C \) represents the combination of rail and contracted drivers, and \( C \) represents contracted drivers only.

Let \( x_{ij} \) be the 0/1 decision variable on whether shipment \( i \in I \) will be assigned to bucket \( j \in J \), and let \( y_j \) be the integer decision variable on the number of trailers closed in bucket \( j \), with a given upper bound \( U_j \). Let \( J_i \) be the set of trailer buckets that are feasible options for shipment \( i \) and \( I_i \) be the set of shipments \( i \) that can be assigned to bucket \( j \). Further, let \( \tilde{c}_{ij} \) be the estimated cost of moving shipment \( i \) from the destination of trailer \( j \) to its final destination (including mileage and/or transfer costs, and penalties for delivering the shipment late). Let \( \bar{p}_j \) be the estimated line-haul cost of moving one trailer in bucket \( j \) from its origin to destination. Let \( w_{ij} \) be the ton-mileage of shipment \( i \) in the trailer route \( j \) and \( \bar{B}_j \) be the ton-mileage capacity of a trailer in bucket \( j \), and let \( s_j \) be the slack variable in the capacity of bucket \( j \), in ton-miles.

Let \( z_{km} \) be the decision variable on the deficit of available duty hours of company drivers of type \( k \in \mathcal{K} \), driving on dispatch legs in the direction \( m \in \mathcal{M} \). Let \( J_{km} \) be the set of trailers \( j \) whose trailer routes include legs that are primarily associated to drivers of type \( k \), in the direction \( m \). Let \( L_{jkm} \) be the set of legs in the route of trailer \( j \) that are primarily associated to drivers of type \( k \), in the direction \( m \). Let \( \bar{h}_{jl} \) be the estimated driver duty hours associated to a trailer \( j \) traveling in dispatch leg \( l \). And let \( \bar{D}_{km} \) be the estimated available duty hours of drivers of type \( k \) in legs in the direction \( m \).

Let \( v_n \) be the decision variable on the ton-mileage excess in transportation mode partition \( n \in \mathcal{N} \). Let \( r_{ijn} \) be the ton-mileage for shipment \( i \) in trailer \( j \), corresponding to mode partition \( n \). Note that if none of the mode(s) in partition \( n \) corresponds to trailer \( j \), then \( r_{ijn} \) will be set
to zero. Let $\tilde{G}_n$ be the estimated maximum desired percentage of mode partition $n$ during the optimization time interval. And let $\tilde{Q}_n$ be the estimated ton-mileage in mode partition $n$, carried over from the previous optimization time period (or inherent to the snapshot data if $t' = t$). The optimization sub-problem can then be stated as the following MIP:

$$\text{Min} \sum_{i\in I} \sum_{j \in J_i} \bar{c}_{ij} x_{ij} + \sum_{j \in J} (\bar{p}_j y_j + P_1 s_j) + P_2 \sum_{k \in K} \sum_{m \in M} z_{km} + P_3 \sum_{n \in N} v_n$$

subject to:

$$\sum_{j \in J_i} x_{ij} = 1 \quad \forall i \in I \quad (1)$$
$$\sum_{i \in I} w_{ij} x_{ij} + s_j = \bar{B}_j y_j \quad \forall j \in J \quad (2)$$
$$\sum_{j \in J_i} \sum_{l \in L_{jkm}} \bar{h}_{jl} y_j - z_{km} \leq \bar{D}_{km} \quad \forall k \in K; \forall m \in M \quad (3)$$
$$\sum_{i \in I} \sum_{j \in J_i} (r_{ijn} - \tilde{G}_n r_{ijA}) x_{ij} - v_n \leq \tilde{G}_n (\tilde{Q}_A - \tilde{Q}_n) \quad \forall n \in N - \{A\} \quad (4)$$

where $P_1, P_2$ and $P_3$ are “sufficiently large” penalties. If $P_2$ and $P_3$ are set to zero, constraints (3) and (4) are effectively relaxed.

### 3 Application to a large LTL carrier: cost saving estimates

We developed an application of the model to an LTL carrier in the USA with a network of 34 major sorting facilities and 264 smaller freight-handling terminals. It picks up about 40,000 new shipments per weekday and has about 4,500 drivers in the line-haul network.

In order to estimate the available duty hours of the drivers in the network, we also developed a driver dispatch simulation model that runs in tandem with the load planning model. In the operational setting, the models run every hour, with a look-ahead horizon $H$ of 36 hours and an optimization time period $\tau$ of 12 hours. The size of the feasible set of trailer buckets $J$ is crucial for the model to run within the operational constraints. That set obviously has to include all the trailer buckets present in the existing static load plan. We implemented a set of carefully designed business rules to augment the static set with a limited, but promising set of adjusted load paths. The bounds on the amount of freight that can be moved by other modes of transportation (rail or contracted drivers) stem from organized labor rules and agreements.

In order to estimate the potential cost saving impact of the model in the operation of the company, we had to run it in a simulated tactical mode. We obtained the sets of historical shipments picked up by the company over an eight-day period in three months of 2017 (April, May and June) and ran a single-shot 7.5-day look-ahead simulation, starting with a snapshot at
the beginning of the period and using only historical shipments (no forecasts). We ran a simulation without any dynamic load plan (using only the existing static load plan) and another with the dynamic load plan on. We then compared a few selected statistics between the two runs for each of those three months. The statistics were: the percentage of adjusted load paths (the static load plan obviously had zero); the trailer load average; a measure of the empty space inside trailers (“air”); and the total cost of operation (line-haul movement plus transfer-handling). The relative variations of those statistics are shown in Table 1, with a plus sign indicating a relative increase and a negative sign indicating a decrease.

Note that solving the MIP model as formulated in section 3 turned out to be too costly computationally (see the run times in Table 1). As a result, we developed an alternate method of solution, by relaxing constraints (3) and (4) in the MIP and using some heuristics to enforce them after the solution of the MIP. As shown in the table, the operational results for the method with the heuristics are comparable to those of the full MIP method.

<table>
<thead>
<tr>
<th>Month</th>
<th>Solution Method</th>
<th>Run Time (hours)</th>
<th>Adjusted Load Paths</th>
<th>Load Average</th>
<th>“Air” inside Trailers</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/17</td>
<td>MIP</td>
<td>68.7</td>
<td>9.6%</td>
<td>+2.5%</td>
<td>-9.4%</td>
<td>-0.91%</td>
</tr>
<tr>
<td>04/17</td>
<td>Heur.</td>
<td>0.89</td>
<td>9.6%</td>
<td>+2.2%</td>
<td>-8.0%</td>
<td>-0.60%</td>
</tr>
<tr>
<td>05/17</td>
<td>MIP</td>
<td>69.1</td>
<td>9.3%</td>
<td>+1.9%</td>
<td>-7.0%</td>
<td>-0.40%</td>
</tr>
<tr>
<td>05/17</td>
<td>Heur.</td>
<td>0.93</td>
<td>9.2%</td>
<td>+1.9%</td>
<td>-7.0%</td>
<td>-0.57%</td>
</tr>
<tr>
<td>06/17</td>
<td>MIP</td>
<td>65.5</td>
<td>9.8%</td>
<td>+2.3%</td>
<td>-8.2%</td>
<td>-0.44%</td>
</tr>
<tr>
<td>06/17</td>
<td>Heur.</td>
<td>1.04</td>
<td>9.7%</td>
<td>+2.1%</td>
<td>-7.5%</td>
<td>-0.50%</td>
</tr>
</tbody>
</table>

Table 1: Selected statistics comparing the simulated dynamic load plan against the static.

The results presented in Table 1 are modest but encouraging. Future areas of research include explicitly modeling the uncertainty in the estimation of some of the model parameters (like, for e.g., the cost of moving shipments all the way to their final destinations).

References


Service level selection and pricing for multimodal package distribution networks

Florian Martin
Email: florian.martin@wu.ac.at
Vera Hemmelmayr Tina Wakolbinger
Vienna University of Economics and Business, Vienna, Austria

1 Introduction

In order to cope with an ever growing number of parcel shipments, researchers and practitioners have introduced a wide array of optimization models to increase the efficiency of the underlying physical distribution process. Service network design (SND), i.e. tactical planning of operations based on optimized consolidation and routing of shipments and vehicles, has appeared particularly promising for improving capacity utilization of resource-restricted logistics networks.

Still, rising transport volumes are not the only challenge faced by the package delivery industry. Tight competition together with customers’ desire to receive their goods ever more quickly have encouraged express carriers to offer a wide range of transportation services. More granular service segmentation, however, adds a whole new layer of complexity to the task of optimizing the logistics operations. This is especially true once the fulfillment models account for the consequences of carriers’ pricing decisions. While many sophisticated models have been developed to assist network planners in minimizing costs, few approaches account for the interplay between service pricing, customer decisions and the associated restrictions in the distribution process.

As such this paper attempts to fill this research gap by introducing a solution approach that simultaneously determines the ideal service offering and the associated pricing as well as the cost-optimal fulfillment plan. We call this new approach the service selection, pricing and fulfillment problem (SSPF).

2 Problem statement

In order to separate consumers who value quick delivery from those that are less time but more price sensitive, express carriers typically segment their services according to
guaranteed delivery time (i.e. one-day, two-day, etc.). Contingent on the set of services offered, customers will pick a service that maximizes their personal net benefit (i.e. utility minus price) and the shipper will then be obligated to deliver the parcel within the specified guaranteed delivery time. We take the perspective of the shipper who tries to determine the optimal set of services and associated prices in order to maximize profit.

The revenue component of the profit function is an adapted multi-product pricing problem in which the firm, that is assumed to be a local monopolist, offers a product range consisting of a homogeneous product which is differentiated by quality (faster service translates into higher quality). Such a model was first proposed by [1], who also developed a heuristic procedure to solve this NP-complete problem. Customers are assumed to behave rationally and each customer’s utility function is dependent on a random sensitivity parameter for delivery time and is determined outside of the model.

The cost component is derived from an SND problem in which the cost minimizing use of vehicles and routing of shipments is determined. The basic intuition is that shorter guaranteed delivery times increase delivery costs as it reduces opportunities to realize economies of scale. This then translates back into the prices and also into the set of services to be offered. Since pricing and routing decisions typically involve different time scales, we should align the differing planning horizons by generating distribution plans that can be executed repeatedly. To assure this we decided to use a cyclical SND formulation as the basis for our model (see [2, 3]). Joining the revenue and cost side of the model yields a profit maximizing formulation in which the individual choices of the customers translate into the routing constraints and vice versa.

3 Mathematical formulation

The carrier’s transport operations are modeled as a space-time network $G = (H, A)$, with the set of arcs $A$ representing possible connections between the hubs $H$ (only intra-hub logistics operations are considered; $A_h$ denotes holding arcs). Arcs and hubs always refer to the time expanded version of the network, i.e. they denote a physical location at a specific point in time. The schedule length is divided into a set of periods $T = \{1, \ldots, T_{\text{MAX}}\}$. Note that our cyclical formulation implies that the first period is the successor of the last.

Using binary variables $o^s$ the shipper can select any combination of services $s \in S$ and must determine the associated profit maximizing prices $p^s$. Each individual customer $n \in N$ obtains a certain utility $u_n^s$ from choosing service $s$. If price exceeds utility for all offered services, the customer will choose the no-purchase option, making his demand zero. Note that we do not model each shipment as a separate customer as this would require an enormous amount of variables. Instead we consider our set of customers to be representative of a larger population (utilities are typically estimated from a sample of
customers, enabling us to work directly with the results of any discrete choice model). As prices are uniform to all and by assuming that customers form a representative sample with preferences that do not differ among locations, we are able to translate the individual customers’ decisions into arbitrarily large demand volumes as follows: Binary decision variables \( z_n^s \) are one if customer \( n \) decides to use service \( s \). Since each representative customer chooses exactly one service, we can aggregate all decisions and interpret this as a market share. As demand is similarly structured across all locations, this share will apply to all commodities \( k \in K \) and respective demands \( d_k \) between hubs. This way we model a potentially unlimited number of parcels by a modest sample of customer decisions.

When one creates the possible commodity paths and vehicle routes a priori, the corresponding route-path formulation of SSPF can be written as follows:

\[
\begin{align*}
\text{max} & & \sum_{k \in K} \sum_{n \in N} \sum_{s \in S} \frac{d_k}{|N|} p^n z_n^s - \sum_{v \in V} \sum_{r \in R_v} f_v^r y_v^r - \sum_{k \in K} \sum_{p \in P_k} c_p^k x_p^k \\
\text{s.t.} & & \sum_{k \in K} \sum_{p \in P_k} x_p^k \alpha_{ij}^p - \sum_{v \in V} \sum_{r \in R_v} \kappa_v y_v^r \leq 0, \quad \forall (i, j) \in A \setminus A_h \\
& & \sum_{r \in R_v} y_v^r \leq e_v, \quad \forall v \in V \\
& & \sum_{\sigma \in S} (u_n^\sigma z_n^\sigma - p^n z_n^\sigma) - (u_n^0 o^s - p^ s) \geq 0, \quad \forall n \in N, s \in S \\
& & \sum_{s \in S} z_n^s = 1, \quad \forall n \in N \\
& & \sum_{n \in N} z_n^s + p^ s - Mo^s \leq 0, \quad \forall s \in S \\
& & \sum_{n \in N} (1 - z_n^0) \frac{d_k}{|N|} = \sum_{p \in P_k} x_p^k, \quad \forall k \in K \\
& & \sum_{\sigma \in S} \sum_{s \in S} \sum_{n \in N} \frac{d_k}{|N|} z_n^\sigma - \sum_{p \in P_k: v \in \tau^s} x_p^k \leq 0, \quad \forall k \in K, s \in S \setminus \{0\} \\
& & o^0 = 1 \\
& & p^0 = 0 \\
& & z_n^s, o^s, y_v^r \in \{0, 1\} \\
& & x_p^k, p^ s \in \mathbb{R}^+ 
\end{align*}
\]

In the objective (1) the sum over all decisions \( z_n^s \) of representative customers \( n \) on services \( s \), divided by the number of customers \( |N| \), gives the market shares. Multiplying by demand \( d_k \) of commodity \( k \) and price \( p^s \) we obtain revenue. From this we subtract fixed costs \( f_v^r \) for using a vehicle of type \( v \) on route \( r \) and variable costs \( c_p^k \), incurred by sending an amount of shipments \( x_p^k \) of commodity \( k \) on path \( p \). Constraint (2) assures that enough vehicle capacity \( \kappa_v \) is provided to serve all selected paths, with \( \alpha_{ij}^p \) and \( \beta_{ij}^r \), indicating whether arc \((i, j)\) is part of path \( p \) and route \( r \) respectively. Constraint (3)
limits our fleet size to a type-specific maximum of $e_v$. Rational behaviour of customers is enforced by (4), while (5) forces them to decide for exactly one service. Using a big-M formulation, constraint (6) sets customer decisions and prices to zero for services that are not offered. The actual quantity to be transported is obtained by subtracting the market share of the no-purchase option from demand $d_k$ (7). This quantity needs to be equal to the sum of path flows. The delivery time requirements are enforced by (8), stating that a sufficient fraction of each commodity must be delivered on paths with a duration $t^p$ less than or equal to the guaranteed delivery time $\tau^s$ of service $s$. Finally, constraints (9) and (10) specify that the no-purchase option is always offered and that its price is zero.

4 Solution approach

The main difficulty of any profit maximizing model is the non-linear revenue term. By discretizing demand we get a product of binary decision variables $z^s_n$ and continuous price variables $p^s$. Such a product can be linearized by replacing it with a new variable, whose behaviour mimics the non-linear product through a set of big-M constraints (see e.g. [4]). Using a standard MIP solver we were able to solve small problem sizes (up to 5 nodes, 7 periods, 5 services and 200 customers) to optimality within 10 hours on a standard desktop machine (note that routes and paths are restricted to meet operational requirements).

Using the optimal MIP solutions as a benchmark, we develop a heuristic procedure that should enable us to solve realistically-sized problem instances. For this we follow the intuition that small changes to prices and customer assignments cause minor changes in the distribution process. Starting from an initial solution, we thus iteratively reassign customer segments, using the previous iteration’s fulfillment plan as the basis for the current service network design optimization. First experiments using diving and rounding-based heuristics showed promising results.

References


Managing Demand in Dynamic Delivery Operations

Ramon Auad  Damian Reyes  Martin W.P. Savelsbergh  
Alan L. Erera
School of Industrial and Systems Engineering,  
Georgia Institute of Technology, Atlanta, USA  
Email: alan.erera@isye.gatech.edu

1 Meal Delivery Operations and Optimization

Online restaurant aggregators – meal-ordering platforms where diners order meals for delivery from an array of restaurants – are growing quickly [2]. According to Morgan Stanley [4], “online food delivery could grow by 16% annual compound rate in next 5 years.” To capture this emerging market, aggregators are investing heavily [3] in meal delivery networks that promise restaurants and diners a reliable, fast and cost-effective delivery experience.

Successful deployment and operation of meal delivery networks is difficult not only due to the scale of these systems, but also due to the dynamism and urgency of arriving orders [8]. Meal delivery is the ultimate challenge in last mile logistics: a typical placed order should be delivered within an hour (much less if possible), and within minutes of the food becoming ready, thus reducing consolidation opportunities and creating the need for more vehicles operating simultaneously and executing shorter, costlier routes. Recent research has considered optimization of delivery driver operations for restaurant meals, focusing on dynamic route generation [6, 7], and effective heuristic methods have been proposed for matching drivers with routes serving one or more orders in real time.

Meal delivery operations are conducted with delivery drivers in one of two categories. One type, called here contract drivers, are employed by the aggregator for shifts of a given duration. Such contract drivers have flexibility to accept or reject delivery orders assigned to them, and earn minimum guaranteed pay per hour during their shift if they maintain a sufficient order acceptance rate. The other driver type, called here ad hoc drivers, use a more flexible work schedule. Like drivers in popular on-demand ride hailing services like Uber or Lyft, these drivers can declare themselves available for assignments dynamically during a day. Both types of drivers typically enjoy quite a bit of freedom in decision-making, importantly including the decision about where to position them-
selves while waiting for a new order assignment request. Meal delivery operators face significant challenges in *driver supply management*, where the goal is to build and tune operating strategies and incentive mechanisms to ensure that driver supply is available cost-effectively when and where it is needed to serve customer requests.

In this paper, we will assume that the supply of driver resources for a meal delivery operation during a given time period is fixed and known, and instead consider problems of managing demand. *Demand management* problems in meal delivery operations are concerned with the design and configuration of dynamic approaches used to reduce demand at some times in specific geographic regions of a market, to ensure that high levels of customer service are provided to placed orders. Schemes like Uber’s “surge pricing” or Lyft’s “prime time” can decrease trip demand dynamically, which we refer to as *demand moderation*. Restaurant aggregators can also use strategies to moderate demand. Also possible is *demand redirection*, where an attempt is made to dynamically incentivize customers to order from restaurants with more available kitchen capacity or nearby delivery capacity.

For now, we ignore customer disutility created when a preferred ordering option is not available at a given time, and instead focus primarily on providing high levels of customer service to placed orders. Two primary service metrics are order *click-to-door* and *ready-to-door* times. Click-to-door time measures the duration between an order placement by a customer and a delivery to that customer. Ready-to-door time measures duration between the time an order becomes ready at a restaurant and the time it is delivered to a customer. To understand fundamental ideas in demand management, we first consider simple deterministic demand management models for a single restaurant with a simplified customer order geography, and a single driver resource.

## 2 Single Driver Delivery Dispatching on the Half-line

One approach for understanding the service level that we can provide for delivery operations from a single location builds on ideas from a study extending the traveling salesperson problem with release dates (as defined in [1]). In the TSP-rd problem, a single driver can be dispatched on multiple consecutive trips from the origin location (depot, restaurant) during the operating day. In this presentation, we introduce extensions from our earlier work in [5] that add order delivery deadlines to TSP-rd problems defined on the half-line. Let $N = \{1, \ldots, n\}$ be a set of order delivery locations on $\mathbb{R}^+ \equiv [0, +\infty)$, and assume that an origin restaurant is located at $x = 0$. Let the location of order $i \in N$ be given by $\tau_i$, and measure travel distances and times such that a round trip from the depot to $i$ requires $2\tau_i$ distance and time. Assume that stopping for a delivery requires no additional time, so that deliveries to a set of orders $J$ requires $\max_{j \in J} 2\tau_j$ distance and time. Each
order $i$ is ready at the restaurant at time $r_i$ (a release time). Finally, consider two delivery deadlines: $T$ is a common deadline by which the final delivery route of the day must be completed, and $S$ is a maximum service time such that an order ready at time $r_i$ must be delivered by $r_i + S$. The second deadline models a maximum ready-to-door time.

Key results in [1] show that single-driver variants of the TSP-rd are solvable on the real line in $O(n^3)$, including one that minimizes total travel distance subject to deadline $T$ and another which minimizes the latest completion time of any route. When limited to the half-line, we show in [5] provides optimal $O(n^2)$ algorithms for these two variants, as well as polynomial algorithms of complexity $O(n^2)$ and $O(n^3)$ respectively for the same two problems augmented with a maximum ready-to-door time of $S$. We introduce new dynamic programming formulations to derive these results, and proof techniques that differ somewhat from the existing literature.

3 Extending Single Vehicle Delivery Dispatching to Order Coverage Maximization

As a first step toward demand management, we first consider extensions of deterministic single-vehicle delivery dispatching problems to selective variants where orders in $N$ can be individually accepted or rejected. Given a set of known orders $N$, such dispatching problems allow us to determine which orders should be suppressed or redirected in order to achieve certain customer service performance goals.

In the first theorem, we consider the problem to serve as many orders as possible with a single vehicle, while ensuring that all work assigned to the vehicle is completed by the shift end $T$. In this case, the objective and the shift end constraint work together to drive down ready-to-door time:

**Theorem 3.1.** The problem to determine the maximum number of orders, $v^*$, that can be delivered by a single driver where each accepted order $i$ is dispatched no earlier than $r_i$ and delivered before a common deadline $T$ can be solved in $O(n^3)$ time.

In the second theorem, we consider the problem again to serve as many orders as possible, but now focus on ensuring that each accepted order meets a ready-to-door deadline:

**Theorem 3.2.** The problem to determine the maximum number of orders, $v^*$, that can be delivered by a single driver where each accepted order $i$ is dispatched no earlier than $r_i$ and delivered before ready-to-door deadline $r_i + S$ can be solved in $O(n^5)$ time.

Finally, since many demand management strategies may attempt to balance lost or diverted demand with service quality provided to accepted orders, the third theorem addresses a problem focused on providing high service levels while accepting and serving
a minimum number of customers; unlike the two prior, note that this problem may be infeasible:

Theorem 3.3. The problem to determine a schedule consisting of a subset of orders $N' \subset N$ of cardinality no less than a given positive number $A \leq n$, which minimizes the schedule completion time (the latest delivery time of any order plus the time required to return to the restaurant) such that each order $i \in N'$ is dispatched no earlier than $r_i$ and delivered no later than $T$ can be solved in $O(n^2)$ time if the driver is limited to carrying one order at a time, and in $O(n^4)$ time otherwise.

4 Demand Management with Congestion Warnings

In the final component of this talk, we consider a simple approach for demand management under uncertainty. While we do not yet consider an unknown stream of orders, we assume that when we do not have complete control over order acceptance. To do so, consider a model where we must serve all customers that order. When a customer arrives to the system to place an order, the restaurant is flagged as green or red. When the restaurant is green, the customer expects no delay in receiving her order so she always orders. When red, a customer should expect service delays and thus may decide not to order from this restaurant; we assume the order probability in this case is $p \in (0,1)$. For this initial study, we focus on the following problem and related variants:

Problem 4.1. Assume that the restaurant flag can be changed immediately after each customer order. Determine a policy $\pi$ that decides whether to display a green or red flag at time $0$ and at times $r_i$, such that at least $A$ customers place orders and expected schedule completion time is minimized.

This stochastic variant, while very simple especially with a known potential order set $N$ and all customer delivery locations on the half-line, still poses significant challenges that may prevent us from finding an efficient optimization algorithm. We are exploring this problem under two delivery operations assumptions. In the first, we assume that each driver trip from the restaurant may serve only a single order at a time. In the second, we allow the driver to bundle and serve multiple orders in a single trip. Given a set of placed but unserved orders at time epoch $t$ and the time $t_v \geq t$ when the driver will return to the restaurant, policies that specify upper and lower bounds respectively on expected schedule completion time can be determined. An upper bound is given by setting the flag green from time $t$ onward, and computing the schedule minimizing completion time. Similarly, a lower bound is given by solving a problem of the type considered by Theorem 3.3, and then setting (or retaining) the flag red when the next arriving customer is not in $N'$. Heuristic strategies using these bounds and other ideas are currently under development, and will be discussed in the presentation.
References


1 Introduction

The vehicle problem associated with parcel delivery in postal companies is quite rich and complex. One of the practical approaches to coping with this problem is to use a cluster first routing second (CFRS) procedure. This method aims to establish well-designed territories that will remain unchanged for several months, subsequently ensuring that good routes are then built on a daily basis. In this paper, we focus on solving the routing phase, which is a time-constrained variant of a traveling salesman problem (TSP).

While the traveling salesman problem with time windows constraints (TSPTW) is well-known, its application in postal services poses some additional challenges. Among those, for parcel deliveries, we notice that if most commercial points of services have time windows (around 10% of total nodes), private customers do not.

Finding a feasible solution to the TSPTW proved to be NP-hard by Savelsbergh [10] in 1985. It has been extensively studied for the last few decades, from Christofides et al. [5] in 1981, and their branch-and-bound algorithm, to Boland et al. [4] and the dynamically generated time-expanded networks in 2017. To our knowledge, the most efficient way to solve the classic TSPTW is presented by Baldacci et al. [2].

The vast majority of papers on this topic consider, as an objective function, the minimization of the traveled distance (or traveled time (TT)). However, in postal services and probably many other industrial settings, focusing only on TT may be very costly as the drivers are paid while waiting. To address this issue, we consider the minimization of the makespan (MS) of a route. The MS is known to be, mostly in scheduling problems, the amount of time necessary to complete all tasks. It could be seen as the sum of traveled time and waiting time. It is also often assimilated to the completion time of the last task. In some context, when time windows are added, starting the route at time $t = 0$ is questionable, and not mandatory in postal services. In Canada, the average wage for a delivery driver of Post Canada is around 20$/hour, while the vehicle cost can be generously
evaluated at 8$/hour when it is moving. Apart from the cost, building a route where the
drivers have limited to no unnecessary waiting is also desired by the postal services.

The contribution of this paper is to compare and combine approaches based both
on a mathematical programming (MIP) formulation and a constraint programming (CP)
scheduling model. Furthermore, to tackle real industrial problems, we introduce a cluster-
ing approach which reduces the size of the problem. The solution on the reduced graph
can then be expanded to the full size by solving an auxiliary problem.

1.1 Problem Definition and Models
The TSPTW consists of visiting, in a directed graph $G = (V,A)$ where $V$ are the vertex
and $A$ the arcs, each node $i \in V$ once with a single vehicle while minimizing the cost
of its route. Each node $i$ has to be visited during a Time Window $W_i = [R_i, D_i]$ ($R_i$ is
the release time and $D_i$ the deadline of the node $i$) and for a given service duration $s_i$.
Early arrival at a service point $i$ is also allowed but the delivery man has to wait until $R_i$.
Traveling along an arc $(i,j) \in A$ takes $t_{ij}$ time units.

MIP model  This model, based on Dash et al. [6] is known as the time bucket formula-
tion (TBF). The idea is to gather several variables of a time indexed formulation into time
buckets, to lower the combinatorial size of the problem. Each bucket $b = [\underline{b}, \overline{b}]$ represents
a portion of the time horizon. The TBF relies on three types of binary variables : $x_{ij}$
indicates if the arc $(i,j)$ is used, $z^b_i$ indicates if the vertex $i$ is visited in the bucket $b$,
and $y^b_{ij}$ specifies if the arc $(i,j)$ is used and if the vertex $i$ visited in the bucket $b$. As
TBF assumes every visit is performed at the beginning of a bucket, sub-tour elimination
constraints (SEC) and infeasible path cuts (IP) are dynamically added to obtain an ex-
act formulation. Reducing the size of the buckets allows for a more accurate model and
reduces the number of required SEC and IP cuts. However, it also means increasing the
number of buckets and thus the number of binary variables. The TBF formulation is well
adapted to minimizing TT, however it cannot easily cope with minimizing MS.

CP model  This model relies on CP-based scheduling techniques [3]. It models the
problem using interval variables $S_i$ which determine the starting time of each visit, a
transition matrix (the $t_{ij}$) to specify the setup (here travel) time between each visit and
a global constraint $NoOverLap(S)$ that forbids two visits from occurring at the same
time. This global constraint is the mathematical equivalent of the disjunction: $S_i + s_i +
t_{ij} \leq S_j$ OR $S_j + s_j + t_{ji} \leq S_i$ $\forall i, j \in N$ but it is implemented using much more
efficient dedicated propagation and filtering algorithms [3]. The model is implemented
using ILOG CPOptimizer, which also relies on a specialized search engine and lower bound
mechanism using Linear Programming. The dedicated solution approach is well engineered
to minimize the MS of a scheduling problem, and while it can still handle a pure TT objective, it has not been designed to efficiently solve pure routing problems.

2 Bi-objective Approaches

To achieve the goal of minimizing both travel and waiting times, we considered two solution approaches given $Z_{TT}$ (resp. $Z_{MS}$) the variable which represents the TT (resp. MS) in the model and $Z_{TT}^*$ (resp. $Z_{MS}^*$), its value. First we simply solve the CP model with a weighted objective function (WOF) of $Z = 20 \times Z_{MS} + 8 \times Z_{TT}$. Second, we define a three-step approach (TSA), along the line of the $\epsilon$-constraints method [9]. The main idea is to limit one or several objective functions by adding a new constraint to the model. The resolution is decomposed in three phases: (1) solve the CP model minimizing only MS and (record $Z_{MS}^*$). (2) Solve the TBF minimizing TT subject to $Z_{MS} \leq Z_{MS}^* + \epsilon$ (where $\epsilon$ allows for a certain tolerance in absolute time units) and record the optimal route $\sigma$. (3) Solve the CP model again, minimizing MS without modifying the sequence $\sigma$.

3 Clustering to Solve Large Industrial Instances

Instances provided by our partner, having between 450 and 500 customers, would simply require too much time to solve with a direct formulation alone. The idea is to cluster, iteratively, the closest nodes without time windows of the original instance, until a fixed cluster number is reached. At each step, we create a new node located at one of the two original ones. The distance matrix is updated after calculating the shortest path, on average, along all the nodes of the cluster to and from their two closest neighbors.

After solving the clustered instance (with either WOF or TSA), leading to a certain route $\sigma^c$, we rebuild the solution of the original instance while respecting the sequence $\sigma^c$. We add constraints to the model precedence specifying that the partial order on nodes is established by $\sigma^c$. We also investigate a version where added flexibility allows mixing customers for clusters that are visited one after the other in $\sigma^c$.

4 Results and Conclusion

We have evaluated six approaches (optimizing only for TT or MS, WOF and TSA with $\epsilon = 5, 10, 20$) on TSPTW instances for the literature [7, 8, 1] and we have computed for each instance and each method the deviation to the best found solution of all approaches. Optimizing only for TT or MS produced the worst solutions with an average deviation on the instances of up to 3% on average [8]. All bi-objective approaches produced solutions with less than 1% deviation. While WOF is the most competitive in terms of cost, it does
sometimes generates solutions with considerable excess waiting time. TSA however allows to bound the total excess waiting time by using an appropriate $\epsilon$.

Preliminary results on large industrial instances indicate that the clustering approach generates, within 30 seconds, solutions that are as good as, and often better than, those generated by a direct resolution using TBF or CP in around 3 hours of CPU.

References


The multi-path Traveling Salesman Problem with dependent random cost oscillations

Edoardo Fadda*
Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy
* Email: edoardo.fadda@polito.it

Lohic Fotio Tiotsop
Department of Mathematical Sciences, Politecnico di Torino, Turin, Italy

Guido Perboli
Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy

Roberto Tadei
Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy

1 Introduction

Reducing air pollution is nowadays one of the main goals of several institutions and governments of industrial countries. Developing methods able to schedule companies logistic operations and public services in a efficient way is coherent with such objective as it would imply less production of pollutants. Several logistic problems can be assimilated to the traveling salesman problem (TSP). Nevertheless, in real applications the stochastic environment deeply influences the performance of the solution. In [1] it has been proven that it is important to consider the stochastic nature of the distribution network in order to obtain affordable solutions. Nevertheless, since the complexity of the stochastic problem grows very fast, deterministic approximations have been developed. All these approximations assume that cost oscillations are independent and identically distributed. In this paper, we relax such hypothesis by assuming that cost oscillations are still identically distributed but just asymptotically independent. This assumption addresses the traffic congestion effects. In particular, we consider the multi-path traveling salesman problem with dependent random cost oscillations (mpTSP_{do}), where the cost oscillations are stochastic with unknown distribution and between any pair of locations there are usually several alternative paths.
We propose a method to find a deterministic approximation solution of the \( mpTSP_{do} \) and we evaluate its quality and efficiency.

2 The stochastic problem

Let us consider a graph network characterized by a set of locations \( N \), a set of scenarios \( S \) and a set of all possible paths \( P = \{ P_{ij} \} \) connecting location \( i \in N \) to location \( j \in N \). Further, we consider \( c_{ij}^{p} \) the unit deterministic cost associated to the path \( p \in P_{ij} \), \( \Theta_{ij}^{ps} \) the random oscillation of the deterministic cost \( c_{ij}^{p} \) under scenario \( s \in S \) and \( C_{ij}^{ps}(\Theta) = c_{ij}^{p} + \Theta_{ij}^{ps} \) the total unit cost required for travelling from the location \( i \) to the location \( j \) using path \( p \in P_{ij} \) under scenario \( s \). Finally, let us consider variable \( x_{ij}^{p} \) equals 1 if path \( p \in P_{ij} \) is selected and 0 otherwise and variable \( y_{ij} \) equals 1 if location \( j \) is visited directly after location \( i \) and 0 otherwise. The model of the multi-path traveling salesman problem with dependent travel cost oscillations problem is

\[
\min_{x,y} E_{\Theta} \left[ \sum_{i \in N} \sum_{j \in N} y_{ij} \sum_{p \in P_{ij}} \sum_{s \in S} x_{ij}^{p} C_{ij}^{ps}(\Theta) \right] 
\]

subject to

\[
\sum_{j \in N, j \neq i} y_{ij} = 1 \quad \forall i \in N \quad (2) \\
\sum_{i \in N, i \neq j} y_{ij} = 1 \quad \forall j \in N \quad (3)
\]

\[
\sum_{i \in U} \sum_{j \in U} y_{ij} \geq 1 \quad \forall U \subset N \quad (4) \\
\sum_{p \in P_{ij}} x_{ij}^{p} = y_{ij} \quad \forall i \in N \quad j \in N \quad (5)
\]

\[
x_{ij}^{p} \in \{0,1\} \quad \forall p \in P_{ij} \quad \forall i \in N \quad \forall j \in N \quad (6) \\
y_{ij} \in \{0,1\} \quad \forall i \in N \quad \forall j \in N \quad (7)
\]

The objective function (1) minimizes the expected total cost, constraints (2) and (3) ensure that each location is visited once, while constraints (4) prevent the formation of sub-tours. Finally, constraints (5) link variables \( x_{ij}^{p} \) to \( y_{ij} \) and (6) and (7) are the integrality constraints. It is worth noting that the model assume that every node is a customer. In this framework, the scenarios describe different type of travel time with respect to the different travel modes.

3 A deterministic approximation of the stochastic problem

Following the approach of [1] we consider that the cost required for traveling from location \( i \) to location \( j \) is

\[
C_{ij}(\Theta) = \min_{p \in P_{ij}} \left( c_{ij}^{p} + \min_{s \in S} \Theta_{ij}^{ps} \right) 
\]

From this consideration, the variables \( x_{ij}^{p} \) assume value 1 iff \( p \) is the cheapest path. Moreover, from Eq. (8) and from the linearity of the expected value the objective value function
min \sum_{i \in N} \sum_{j \in N} E_{\Theta} [C_{ij}(\Theta)] y_{ij} \tag{9}

Unfortunately, the distributions of the $\Theta_{ij}^{ps}$ are unknown and thus the expected value in (9) is not solvable. In [1] the authors assume that the random cost oscillations $\Theta_{ij}^{ps}$ are independent and identically distributed random variables with unknown probability distribution. Under a mild assumption on that unknown probability distribution, they give a deterministic approximation of the stochastic problem.

In this paper, we want to relax the independence assumption of the random cost oscillations. We assume that these oscillations are asymptotically independent. This assumption addresses the traffic congestion effects where these oscillations cannot be considered independent any more.

**Definition 3.1** Let $X_1$ and $X_2$ be random variables. They are asymptotically independent if

$$\lim_{r \to -\infty} (P(X_1 < r|X_2 < r) - P(X_1 < r)) = 0 \tag{10}$$

It is worth noting that by subtracting from all random cost oscillations $\Theta_{ij}^{ps}$ a constant $\alpha_{|S|}$ and by dividing the resulting variables by any other constant $\gamma_{|S|} > 0$ the solution of the problem does not change. Hence, the cost $C_{ij}(\Theta)$ becomes

$$C_{ij}(\Theta) = \min_{p \in P_{ij}} \left( c_{ij}^{p} + \min_{s \in S} \left( \frac{\Theta_{ij}^{ps} - \alpha_{|S|}}{\gamma_{|S|}} \right) \right) \tag{11}$$

Let $F_{ij}^{p} = P(\Theta_{ij}^{ps} > x)$ be the unknown survival function of the random oscillations $\Theta_{ij}^{ps}$.

We proved in [2] the following theorem

**Theorem 3.1** Assuming that the random cost oscillations $\Theta_{ij}^{p1s}$ and $\Theta_{ij}^{p2s}$ are asymptotic independent $\forall \ p_1, p_2 \in P_{ij}$ under each scenario $s$ and that we choose $\alpha_{|s|}$ and $\gamma_{|s|}$ such that

$$\lim_{|S| \to \infty} \left( F_{ij}^{p \left( \gamma_{|s|} x - \alpha_{|s|} \right)} \right)^{|S|} = \exp \left( -e^{\beta x} \right) \quad \text{for some real number } \beta > 0, \tag{12}$$

then

$$\lim_{|S| \to \infty} P(C_{ij}(\Theta) > x) = e^{-A_{ij} e^{\beta x}}, \quad \text{where } A_{ij} = \sum_{p \in P_{ij}} e^{-\beta c_{ij}^{p}} \forall i \in N \forall j \in N \tag{13}$$

It is worth noting that (12) is a mild assumption as, for suitable constants $\alpha_{|s|}$ and $\gamma_{|s|}$, it holds for several distributions. If $|S|$ is large enough, the limit obtained in (13) can be used as the survival function of the costs $C_{ij}(\Theta)$. Thus, after some manipulations, the expected value in (9) becomes

$$E_{\Theta} [C_{ij}(\Theta)] \approx \int_{-\infty}^{+\infty} x e^{-A_{ij} e^{x}} A_{ij} e^{x} dx = -\frac{1}{\beta} (\ln(A_{ij}) + \gamma) \tag{14}$$
where $\gamma = -\int_0^\infty \log(t)e^{-t}dt \approx 0.5772$ is the Euler constant.

Using (14) and disregarding the constants the following deterministic approximation of the stochastic problem is finally obtained

$$\min_y \sum_{i \in N} \sum_{j \in N} -\frac{1}{\beta} y_{ij} \ln A_{ij}, \quad \text{s.t. (2), (3), (4), (7)}$$

(15)

4 Computational results and conclusions

We compute some numerical experiments in order to evaluate the effectiveness of our deterministic approximation. In particular, we compare the solution of our deterministic approximation with the one of the Perfect Information case, computed by means of a Monte Carlo simulation performed on the stochastic problem. The percentage gap is defined as the relative percentage difference between the optimum obtained with the deterministic approximation and the one provided by the Monte Carlo simulation. In the experiments we used 20 locations and consider the $\Theta$ to be distributed according to a multivariate normal. For the sake of space we present here just a few results in Table 1. For the total number of scenarios $|S| = 100$, the percentage gap turns out to be 3.7%, but it significantly decreases as the number of paths between pairs of locations increases. It is important to note that on average the deterministic approximation needs only 0.1 seconds to be computed, while the stochastic approach needs 36 seconds.

Table 1: Number of paths and percentage gap obtained by using $|S| = 100$

<table>
<thead>
<tr>
<th>number of paths</th>
<th>percentage gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.7</td>
</tr>
<tr>
<td>3</td>
<td>2.25</td>
</tr>
<tr>
<td>5</td>
<td>0.76</td>
</tr>
</tbody>
</table>

References


Algorithms for the Traveling Salesman Problem with Time-Dependent Service Times

Valentina Cacchiani
Department of Electric, Electronic and Information Engineering “Guglielmo Marconi”,
University of Bologna, Bologna, Italy
Email: valentina.cacchiani@unibo.it

Carlos Contreras-Bolton
Department of Electric, Electronic and Information Engineering “Guglielmo Marconi”,
University of Bologna, Bologna, Italy

Paolo Toth
Department of Electric, Electronic and Information Engineering “Guglielmo Marconi”,
University of Bologna, Bologna, Italy

1 Introduction

We study the Traveling Salesman Problem with Time-Dependent Service Times (TSP-TS), an extension of the well-known NP-hard Asymmetric TSP (ATSP), recently introduced in [4]. Differently from classical works on ATSP, in TSP-TS the customer service duration depends on the time at which the service starts at the customer, and therefore cannot be simply included as a constant in the travel time. Several real-world applications exist for TSP-TS, since service times can depend on the time period of the day (e.g. parking or personnel availability). Some works on the time-dependent TSP include this variability in the travel times ([1]), instead of in the service times, but cannot easily tackle TSP-TS, as mentioned in [4]. In [4], a compact Mixed Integer Programming (MIP) model has been proposed, and lower and upper bounds have been derived to improve its performance. Our contribution is to propose a Branch-and-Cut (BC) and a Genetic Algorithm (GA) to solve TSP-TS. The proposed BC and GA algorithms are compared with the model proposed in [4], showing their effectiveness.

2 Problem Definition and Mathematical Model

In TSP-TS, we are given a complete directed graph $G = (N, A)$, where $N = \{0, 1, \ldots, n, n+1\}$ is the set of nodes and $A$ is the set of arcs. Each node in $N \setminus \{0, n+1\}$ represents a
customer to be visited, while nodes 0 and \( n + 1 \) correspond to the depot, that is duplicated for convenience. Each arc \((i, j) \in A\) is assigned a travel time \( t_{ij} \). Each customer \( i \in N \setminus \{0, n + 1\} \) requires a service time, defined as a continuous function \( s_i(b_i) \), where \( b_i \) represents the start time of service at customer \( i \) \((b_0, s_0(b_0), \text{ and } s_{n+1}(b_{n+1}) \) are equal to 0). TSP-TS calls for determining a Hamiltonian path from node 0 to node \( n + 1 \) visiting each customer \( i \in N \setminus \{0, n + 1\} \) exactly once and having minimum total duration, given by the sum of the total travel and service times. We introduce binary variables \( x_{ij} \) \((i \in N, j \in N)\) assuming value 1 if node \( j \) is served immediately after node \( i \) (and 0 otherwise), non-negative variables \( b_i \) \((i \in N \setminus \{0, n + 1\})\), and non-negative variables \( g_{ij} \) denoting the number of arcs on a path from the depot 0 to arc \((i, j) \in A\). The MIP model proposed in [4] reads as follows:

\[
\min \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ij} + \sum_{i \in N \setminus \{0, n + 1\}} s_i(b_i) \tag{1}
\]

\[
\sum_{j \in N \setminus \{i\}} x_{ij} = 1, \quad i \in N \setminus \{n + 1\}, \tag{2}
\]

\[
\sum_{i \in N \setminus \{j\}} x_{ij} = 1, \quad j \in N \setminus \{0\}, \tag{3}
\]

\[
b_i + s_i(b_i) + t_{ij} - M(1 - x_{ij}) \leq b_j, \quad i \in N, \quad j \in N, \tag{4}
\]

\[
\sum_{j \in N \setminus \{0\}} g_{ij} - \sum_{j \in N \setminus \{0, n + 1\}} g_{ji} = 1, \quad i = 1, \ldots, n, \tag{5}
\]

\[
0 \leq g_{ij} \leq nx_{ij}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n + 1. \tag{6}
\]

\[
b_i \geq 0, \quad i \in N, \tag{7}
\]

\[
x_{ij} \in \{0, 1\}, \quad i \in N, \quad j \in N. \tag{8}
\]

The objective function (1) requires the minimization of the sum of the total travel times and of the total service times. Constraints (2) and (3) impose, respectively, to have exactly one outgoing and one ingoing arc from/to each node, except for the depot. Constraints (4) require that the start time of service at node \( j \) must be at least the start time of service at node \( i \) plus the service time \( s_i(b_i) \) at node \( i \) plus the travel time from \( i \) to \( j \), if node \( j \) is visited immediately after node \( i \). These constraints use a large positive constant \( M \), whose value is determined by a lower bounding procedure (see [4]). Finally, constraints (5)-(6) are the Gavish and Graves (GG) constraints [2], while (7) and (8) are the variable domain constraints.

### 3 Branch-and-Cut and Genetic Algorithms

We propose a BC algorithm, based on model (1)-(8), whose continuous relaxation is strengthened by replacing the GG constraints (5)-(6) with the explicit subtour elimination
constraints (SECs):

\[
\sum_{i \in S} \sum_{j \in N \setminus S} x_{ij} \geq 1, \quad S \subseteq N \setminus \{n + 1\}, \quad 0 \in S, \quad |N| \geq 2.
\]  

(9)

The SECs are separated by using the separation procedure proposed in [3]. In addition, we propose an improved lower bound on the total service time obtained by taking into account, for each node, the minimum travel time of the arcs entering the node and the minimum travel time of the arcs leaving the node. Furthermore, we propose a GA to quickly derive heuristic solutions. GA is based on the optimal solution of the continuous relaxation of the improved model and on the combination of effective operators for ATSP, such as Order Based Crossover, Distance Preserving Crossover, Exchange Mutation, Scramble Mutation, Displacement Mutation and Inversion Mutation. The solution values obtained by GA are used also as upper bounds on the total route duration within BC, and to improve the \( M \) values in (4). In addition, improved lower and upper bounds on the start time of service at each customer \( i \) are derived by taking into account the values of the GA solution and of the shortest paths from 0 to \( i \) and from \( i \) to \( n + 1 \).

4 Computational Results

Model (1)-(8), and algorithms BC and GA were implemented in C++ and all the experiments were performed on an Intel(R) Core(TM) i7-6900K with 3.20 GHz and 64 GB of RAM, using GNU/Linux Ubuntu 16.04 (by using single thread) and CPLEX 12.7.1 as MIP solver. We consider the 22 instances proposed in [4], that come from the TSPLIB library. They contain up to 45 nodes and have a symmetric travel time matrix. We report the results obtained when considering small linear service times with \( s_i = 5(10^{-3})b_i + 3(10^{-2}) \).

The comparison of the three methods is shown in Table 1. We report, for each instance, the value \( Opt \) of its optimal solution. For model (1)-(8) and algorithm BC, we report the percentage gap \( RG\% \) of the lower bound computed at the root node from \( Opt \) and the corresponding computing time \( RT \), as well as the total computing time \( TT \) (both methods always obtain the optimal solution). In addition, for GA, we report the average percentage gap \( AG\% \) of the solution values found by GA, over 10 runs, from \( Opt \), and the corresponding computing time \( AT \), and the minimum gap \( MG\% \). Finally, in the last row, we show average values over all the instances. Computing times are expressed in seconds.

The results show that BC can significantly reduce the gap of the lower bound from \( Opt \) at the root node with respect to that of [4], while the computing times of the two methods at the root node are comparable. Both methods can solve all the instances to optimality, but the total computing times of BC are much smaller than those of [4]. Finally, we can observe that, by taking the best value out of 10 runs, GA always finds the optimal solution, despite it requires a short computing time, and, GA, on average, has a very small gap.
<table>
<thead>
<tr>
<th># instance</th>
<th>Opt</th>
<th>RG%</th>
<th>RT</th>
<th>TT</th>
<th>BC</th>
<th>RG%</th>
<th>RT</th>
<th>TT</th>
<th>AG%</th>
<th>AT</th>
<th>MG%</th>
</tr>
</thead>
<tbody>
<tr>
<td>burma14</td>
<td>228.83</td>
<td>3.88</td>
<td>0.08</td>
<td>0.32</td>
<td>1.36</td>
<td>0.17</td>
<td>0.18</td>
<td>0.00</td>
<td>0.11</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>ulysses16</td>
<td>271.74</td>
<td>4.77</td>
<td>0.12</td>
<td>2.18</td>
<td>1.44</td>
<td>0.12</td>
<td>0.23</td>
<td>0.00</td>
<td>0.12</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>gr17</td>
<td>238.39</td>
<td>3.53</td>
<td>0.13</td>
<td>1.44</td>
<td>0.89</td>
<td>0.14</td>
<td>0.16</td>
<td>0.00</td>
<td>0.15</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>gr21</td>
<td>237.11</td>
<td>2.16</td>
<td>0.16</td>
<td>0.38</td>
<td>1.84</td>
<td>0.18</td>
<td>0.26</td>
<td>0.00</td>
<td>0.21</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>ulysses22</td>
<td>306.44</td>
<td>7.26</td>
<td>0.20</td>
<td>19.84</td>
<td>2.90</td>
<td>0.19</td>
<td>1.40</td>
<td>0.01</td>
<td>0.22</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>gr24</td>
<td>249.09</td>
<td>2.09</td>
<td>0.23</td>
<td>1.99</td>
<td>1.46</td>
<td>0.25</td>
<td>0.50</td>
<td>0.00</td>
<td>0.27</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>fri26</td>
<td>247.99</td>
<td>2.87</td>
<td>0.36</td>
<td>2.62</td>
<td>1.37</td>
<td>0.25</td>
<td>0.46</td>
<td>0.00</td>
<td>0.32</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>bays29</td>
<td>345.49</td>
<td>2.73</td>
<td>0.42</td>
<td>8.33</td>
<td>1.56</td>
<td>0.31</td>
<td>1.21</td>
<td>0.46</td>
<td>0.39</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>att30</td>
<td>253.85</td>
<td>9.43</td>
<td>0.41</td>
<td>223.61</td>
<td>3.06</td>
<td>0.33</td>
<td>10.15</td>
<td>0.00</td>
<td>0.42</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>dantzig30</td>
<td>324.21</td>
<td>7.73</td>
<td>0.27</td>
<td>165.18</td>
<td>3.03</td>
<td>0.35</td>
<td>5.87</td>
<td>0.31</td>
<td>0.42</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>eil30</td>
<td>323.40</td>
<td>2.29</td>
<td>0.28</td>
<td>3.42</td>
<td>1.57</td>
<td>0.35</td>
<td>0.90</td>
<td>0.33</td>
<td>0.43</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>gr30</td>
<td>283.91</td>
<td>6.58</td>
<td>0.38</td>
<td>14.00</td>
<td>2.63</td>
<td>0.31</td>
<td>1.52</td>
<td>0.11</td>
<td>0.44</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>hk30</td>
<td>324.20</td>
<td>7.73</td>
<td>0.34</td>
<td>168.18</td>
<td>2.31</td>
<td>0.34</td>
<td>15.61</td>
<td>0.47</td>
<td>0.44</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>swiss30</td>
<td>342.50</td>
<td>4.72</td>
<td>0.33</td>
<td>7.65</td>
<td>2.33</td>
<td>0.34</td>
<td>1.06</td>
<td>0.00</td>
<td>0.46</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>eil35</td>
<td>363.39</td>
<td>3.76</td>
<td>0.49</td>
<td>44.19</td>
<td>1.56</td>
<td>0.57</td>
<td>10.81</td>
<td>0.43</td>
<td>0.65</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>gr35</td>
<td>281.82</td>
<td>6.67</td>
<td>0.61</td>
<td>149.94</td>
<td>3.10</td>
<td>0.49</td>
<td>20.64</td>
<td>0.87</td>
<td>0.65</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>swiss35</td>
<td>373.60</td>
<td>4.48</td>
<td>0.50</td>
<td>26.10</td>
<td>2.79</td>
<td>0.53</td>
<td>5.86</td>
<td>0.00</td>
<td>0.69</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>eil40</td>
<td>410.35</td>
<td>4.77</td>
<td>0.81</td>
<td>217.95</td>
<td>1.79</td>
<td>0.86</td>
<td>50.45</td>
<td>0.33</td>
<td>0.90</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>dantzig42</td>
<td>257.37</td>
<td>6.38</td>
<td>0.80</td>
<td>2302.25</td>
<td>3.49</td>
<td>0.71</td>
<td>118.89</td>
<td>0.43</td>
<td>1.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>swiss42</td>
<td>351.15</td>
<td>5.76</td>
<td>0.85</td>
<td>2073.20</td>
<td>2.93</td>
<td>0.77</td>
<td>134.15</td>
<td>0.03</td>
<td>1.02</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>eil45</td>
<td>448.11</td>
<td>3.61</td>
<td>1.02</td>
<td>2062.93</td>
<td>2.16</td>
<td>1.00</td>
<td>772.53</td>
<td>0.12</td>
<td>1.19</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>308.74</td>
<td>4.92</td>
<td>0.41</td>
<td>341.57</td>
<td>2.16</td>
<td>0.40</td>
<td>52.56</td>
<td>0.19</td>
<td>0.50</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison with small service times.

from *Opt*. Computational results we executed on the same set of instances with medium and large linear service times, and with quadratic (non-increasing) service times show that BC and GA achieve a good performance also in these settings.

References


An Exact Algorithm to Solve the Vehicle Routing Problem with Stochastic Demands under an Optimal Restocking Policy

Majid Salavati-Khoshghalb
CIRRELT and Département d’informatique et de recherche operationnelle, Université de Montréal, Montreal, Canada

Ola Jabali
CIRRELT and Dipartimento di elettronica, informazione e bioingegneria, Politecnico di Milano, Milano, Italy

Walter Rei
Département de management et technologie, École des sciences de la gestion, Université du Québec à Montréal, Montreal, Canada

Michel Gendreau
CIRRELT and Département de mathématiques et de génie industriel, Polytechnique Montréal, Montreal, Canada
Email: michel.gendreau@cirrelt.net

1 Introduction

The Vehicle Routing Problem (VRP) is one of the central problems in logistics. Since the late 1950’s, it has been the object of a considerable amount of research efforts. In the VRP, the goal is to find a set of routes serving a given set of customers at minimum cost (often, minimum distance). To each customer is associated a given demand for some homogeneous product. Each route should begin and end at a specified depot and serve a subset of customers whose total demand does not exceed the capacity of the vehicle performing this route. In general, the fleet is assumed to be homogeneous with all vehicles having capacity $Q$.

Most of the research on the VRP has focused on the deterministic version of the problem and on its extensions, in which all problem parameters are known precisely before making the routes. However, in practice, several parameters of the problem can be uncertain. One way of addressing this situation is through the use of probability distributions to describe the uncertain parameters, which thus become random variables. This gives
rise to the so-called Stochastic VRPs (SVRPs). While this class of problems has received significantly less attention than deterministic VRPs, there have been nonetheless several efforts to tackle different stochastic variants of the VRP. The most commonly studied involve stochastic demands, stochastic customer presence, or stochastic travel and service times. See [4] for a comprehensive exposition.

In this talk, we focus on the Vehicle Routing Problem with Stochastic Demands (VRPSD), where only customer demands are stochastic and specified through probability distributions; all other problem parameters are assumed to be deterministic and known. As for the actual demand of each customer, it is usually assumed that it is only observed upon arrival at the customer location.

Several modeling paradigms have been proposed to formalize the problem and the way in which it is solved (see [4]). In this work, we adopt the a priori optimization approach, which was extensively discussed in [1]. Under this approach, the overall decision-making process is broken into two steps that correspond respectively to the planning of the routes and their execution. While the planning stage is rather obvious, it is important to understand what may happen when one tries to perform (execute) a planned route. We first remark that while one could plan routes that would be feasible for all possible realizations of the stochastic demands, in practice, this would be highly inefficient. This means that it is usual (and, on average, effective) to plan routes that cannot deal with the maximal demand of their customers. However, this also means that for some demand realizations, a route may fail. More precisely, a route failure occurs when the demand of a customer exceeds the residual capacity of the vehicle. Whenever a failure occurs, corrective actions, called recourse actions, must be taken and some associated costs, called recourse costs, need to be incurred. The objective in the VRPSD is to minimize the total expected costs (usually measured in distance), which consist of planned route costs, which are deterministic, and expected recourse costs. It is interesting to note that, under the a priori approach, the VRPSD can be cast into the framework of a two-stage stochastic integer program with recourse (see [2] for a comprehensive coverage of stochastic programming).

Several recourse policies have been proposed. The earliest and most used one, which is called the classical recourse policy, was introduced in [3]. Under this policy, the driver follows the planned route until the vehicle returns to the depot (no-failure case) or its capacity is depleted at some customer location. This depletion may occur in two ways: (i) A failure occurs, and then vehicle must perform a back-forth (BF) trip to the depot to replenish its capacity in order to complete the service; or (ii) The vehicle capacity is exactly depleted by a customer demand, in which case, the vehicle returns to the depot to restock and then continues to the next customer (except if this is the last customer).

An alternative recourse policy is the optimal restocking policy, in which vehicles can also perform preventive return (PR) trips to the depot in anticipation of costly failures.
This policy was originally proposed by [9] and made famous by [8]. For a given planned route, the optimal restocking policy can easily be determined by solving a straightforward dynamic programming recursion, which allows to determine for each customer an optimal threshold on the residual capacity after serving this customer that is used to decide whether or not a PR trip should be executed. The objective of the underlying dynamic program is the minimization of the total expected costs of performing the route.

While implementing the optimal restocking policy for a given route is easy, the integration of this policy into an exact solution methodology is extremely challenging. In fact, for several years, there have been no successful attempts in this direction. Very recently, Louveaux and Salazar González [7] have proposed an exact solution approach, based on the Integer L-shaped method, to solve the VRPSD under an optimal restocking recourse policy. It should be noted that, while this paper provides bounding procedures applicable to instances in which customer demand distributions are not identical, much of this work focuses on the case where all customers have identical demand distributions and all the reported computational results cover only this case.

The purpose of this talk is to present an exact algorithm to solve the VRPSD under an optimal restocking recourse policy. An important feature of our approach is that it allows for the consideration of different demand distributions for the customers in a computationally effective way, as long as they are discrete and with finite support.

2 Model

The general formulation of our model for the VRPSD under an optimal restocking policy follows existing two-stage formulations for the VRPSD under the classical recourse policy. Namely, the first stage of the model is almost identical to the well-known 2-index formulation for the undirected Capacitated VRP, with \( x_{ij} \) \( (i < j) \) decision variables, which denote the number of times each edge \((v_i, v_j)\) is traversed in the first stage (see, for instance, [6] for further details). The only important difference to notice is the addition of a term \( Q(x) \) to denote the cost of expected recourse actions to the objective function.

Generally speaking, in VRPSD models, the second-stage model, which corresponds to the application of the selected recourse policy, is not described mathematically, since no optimization takes place at this point: one simply has to derive an appropriate computational procedure to compute the value of the expected recourse costs for any first-stage solution \( x \). In our case, we must recall that our recourse policy involves the optimization of the thresholds that govern the decisions to return preventively to the depot. We thus have an explicit second-stage model, which will be presented in detail at the conference.
3 Solution Approach and Computational Results

Our solution approach relies on the general structure of the Integer L-shaped algorithm, augmented by various effective bounding procedures that are used to derive lower bounding functional (LBF) cuts at integer and fractional solutions as in [6, 5]. A key element of our algorithm is the fact that we are able to derive efficiently effective bounds for the optimal restocking expected cost. These bounds will be described in detail at the conference.

The method was applied to three sets of instances, including instances with asymmetric distribution for demands. Our experiments showed that we were able to solve optimally several of the instances proposed in [7], as well as instances with different distributions for some customers. The largest instances solved have up to 60 customers and four vehicles.

References


Dynamic Time Window Allocation and Sizing for Service Routing Applications

Justin C. Goodson
Department of Operations & Information Technology Management
Saint Louis University, St. Louis, MO
Email: goodson@slu.edu

Marlin W. Ulmer
Carl-Friedrich-Gauß Department,
Technische Universität Braunschweig, Braunschweig, Germany

Barrett W. Thomas
Department of Management Sciences,
University of Iowa, Iowa City, IA

1 Motivation

Firms providing on-location services, such as utility repair and home-attended delivery companies, typically communicate an expected arrival time or a self-imposed service time window (TW) to customers at the time a request is made. At time of communication, the providers do not know for certain when and where additional service requests might be realized in the future. Thus, arrival times or TWs need to be determined under incomplete information about the set of customer eventually to serve. Even though the literature analyzes TW allocation and sizing under incomplete information in travel time (Vareias et al. 2017) or customer demand (Spliet and Gabor 2015, Spliet and Desaulniers 2015), research on TW allocation and sizing with stochastic requests is still unexplored.

In previous research, we presented an offline simulation method to approximate expected arrival times by means of simulation (Ulmer and Thomas 2017). The results show that this approximated arrival time has been proven to be a good basis for TW allocation. In Ulmer and Thomas (2017), we showed how suitable TW allocation can be achieved by centering the TW around the expected arrival time. Yet, we also note that a fixed TW size does not service all customers equally. In particular, because the variability in the arrival time distribution may change from one customer to the next, a static TW size inevitably leads to differences in service levels across the customer base. Hence, individualized TW sizes may improve the customers’ service experience.
The task of time window sizing requires careful consideration of service level and customer satisfaction. For example, a wide TW may ensure service will occur during the promised period, but may require the customer to wait an unreasonable amount of time. Alternatively, a narrow TW may appeal to the customer, but may be difficult for the firm to satisfy leading to many violations and ultimately customer dissatisfaction. Thus, the provider has two competing goals; narrow TWs and a low number of violations.

To address this tradeoff, this paper develops state-dependent methods for TW sizing. To do so, we size TWs with respect to the corresponding variance of the arrival time distribution. We show that dynamically sizing TWs as a function of arrival time variance can significantly improve a firm’s service level relative to the case of a static TW size.

2 Model

We consider the problem of allocating and sizing TWs in the context of the vehicle routing problem with stochastic requests (VRPSR). During a capture phase, a set of customers request service. These stochastic requests are handled by an exogenous acceptance and routing policy leading to a sequence of routing plan updates over the course of the capture phase. The accepted customers are served in a subsequent delivery phase based on the final routing plan. At a given VRPSR decision epoch, for each new request accepted for service, the decision maker communicates to the customer a TW \([l, u]\) within which the service is supposed to occur. Parameter \(l\) is the beginning of the TW, parameter \(u\) is the end of the TW. We assume that a customer is willing to accept a certain range of TW sizes. In our preliminary tests, we limit the range of TW sizes between 30 minutes and at most 120 minutes: \(30 \leq u - l \leq 120\).

The objective for this problem is twofold: First, the providers aim on minimizing the expected number of TW violations to reduce customer inconvenience. Second, the providers want to minimize the expected average TW size to increase the customers’ experience. Thus, the objectives are competing.

3 Method

Calculating the values \(l\) and \(u\) directly based on a current state is computationally challenging. First, because customers are waiting for response, the real-time runtime in a state is highly limited. Second, as we show in Ulmer and Thomas (2017), the arrival time distribution is complex. Thus, we need to approximate.

In Ulmer and Thomas (2017), we used a state space aggregation in combination with nonparametric approximation and simulations to determine the expected arrival time \(a\) for a customer accepted at a current state. To this end, we aggregated a state to a set of three features reflecting the current routing and the potential for future changes: the
current travel duration to the new customer, the overall travel duration, and the free
time to incorporate new requests. The resulting three-dimensional vector space was then
partitioned and the average arrival time per partition was approximated by simulation.
The aggregation allowed an offline simulation and a storage of the values allowing real-time
responses to customers without additional online calculations.

The aim of our preliminary study is to show the potential of dynamically sizing TWs
for customer service. Thus, we present a first, intuitive method of sizing TWs based on
the arrival time variances. First, we extend the nonparametric approximations of Ulmer
and Thomas (2017) to also capture the variance $\sigma$ of the mean arrival time $a$ experienced
in a specific state. This variance is an indicator for the certainty in our arrival time
approximation. With the objective of minimizing violations, a small variance may allow
for a narrow TW while a large variance may require a wider TW. The approximation of
the variance follows the same procedure as the approximation of the arrival times.

Using the approximations of expected arrival time $a$ and variance $\sigma$, we center the
TW-parameters around the arrival time: $l = a - r$ and $u = a + r$ with radius $r$. The
radius of TW $[l, u] = [a - r, a + r]$ depends on the variance $\sigma$ with $r = 15 + \min\{\sigma^p, 45\}$
minutes. In case of a variance of 0, the TW size is set to 30 minutes. In case of a very
large variance, the TW size is 120 minutes. The parameter $p$ controls the average size
of the TWs and shifts the focus between the two competing objectives. A small value $p$
leads to narrower TWs but more violations, a large value $p$ leads to wider TWs but less
violation.

4 Results

We test our method on customer locations of Iowa City for different instance settings
varying in the number of requests and the service time at the customer. We compare
our method with a fixed TW size of 60 minutes, centered around $a$ as proposed in Ulmer
and Thomas (2017). We address the competing objectives by looking at each of them
individually and moving the other one in the constraints. We first analyze the number
of violations in case the average TW size is 60 minutes. We also compute the average
TW size necessary to achieve the same percentage of violations as the benchmark policy
with fixed TW sizes of 60 minutes. The corresponding values for $p$ are set via a stochastic
gradient procedure.

We consider three schemes to dynamically size TWs. The “continuous” method sizes
TWs as describe above. Recognizing TWs of non-integer duration may be perceived as
odd by customers, methods “15” and “30” restrict the TW sizes to increments of 15 and
30 minutes, respectively. Preliminary results are encouraging. For each method, Figure 1
depicts the percent reduction over the fixed 60 minutes TW size in number of violations as
well as amount of violation. Notably, each method to dynamically size TWs significantly improves upon a fixed sizing scheme. Further, restricting TWs to 15-minute increments offers improvement comparable to the unrestricted case. Further, not shown in the figure is the 6% reduction in average time window width that is possible with the proposed scheme versus the fixed width benchmark.

Even with a straightforward method solely based on the variance, we show that dynamic sizing of TWs offers a huge potential to increase customer satisfaction. Based on these preliminary results, the next steps will focus on developing more sophisticated, state-dependent methods to approach the values for $l$ and $u$. Particularly, we will draw on concepts of machine learning such as artificial neural nets to determine suitable values and TW sizes.

**References**


\[Figure 1: \text{Improvement of Dynamic TW sizing Compared to Fixed TW sizes}\]
Exact and approximate solution methods for the vehicle routing problem with stochastic and correlated travel times

Fausto Errico
École de Technologie Supérieure de Montréal and GERAD, Montréal, Canada
Email: fausto.errico@etsmtl.ca

Guy Desaulniers, Andrea Lodi, Borzou Rostami
École Polytechnique de Montréal and GERAD, Montréal, Canada

1 Introduction

We consider a variant of the capacitated vehicle routing problem (CVRP) where travel times between locations are assumed to be stochastic. The time duration of a vehicle route is given by the sum of travel times between each visited location, and it is also stochastic. Depending on the particular travel time realization, the duration of a route can be affected by a large variability. In many real-life contexts, however, it is desirable to attenuate the dispersion of a route duration around its expected value. For example, if a customer expects a visit at a given time, it is beneficial to provide service at a time instant close to the expected one. In this context, a decision maker might prefer to design vehicle routes with lower variability to the expenses of a slightly higher expected route duration. Most of the literature on stochastic vehicle routing problems assumes that random variables are statistically independent. This assumption conflicts, however, with real-life contexts, where statistical dependence is rather the norm. For example, several empirical studies found that strong correlation exists, both positive and negative, not only among adjacent links of a node network, but also among geographically distant links (see [2]). In this paper we consider the CVRP with stochastic and correlated travel times (VRPSCT). We assume that the first and the second moments of the travel time probability distributions are known. To the best of our knowledge, we are the first to address this problem setting, although some literature on the quadratic shortest path and quadratic traveling salesman problems are related to our setting (see [1], for example). For the VRPSCT we introduce a new mathematical formulation, which turns out to be a parametric quadratic binary program with an exponential number of variables. We then develop a Branch-price-and-cut algorithm (BPC). The proposed method, beyond providing optimal solutions for instances
of moderate size (up to 30 customers from our dataset), can be suitably exploited in an approximate solution framework. In fact, by reformulating the original problem via an eigenvalue decomposition of the quadratic component in the objective function, we can apply the BPC algorithm to obtain very good quality approximate solutions. The proposed method also provides performance guarantees.

2 The VRPSCT

Let us consider a graph \( G = (V, A) \) with vertex set \( V = \{0, 1, 2, \ldots, n\} \) and arc set \( A \), with \(|A| = m\). Each vertex \( i \in V_c = V \setminus \{0\} \) represents a customer having a nonnegative demand \( q_i \), while vertex 0 corresponds to a depot. Each arc \((i, j) \in A\) represents a link between two nodes \( i, j \in V \). Let \( t_{ij} \) be a random variable representing the travel time on arc \((i, j)\) with mean \( \mu_{ij} \) and the standard deviation \( \sigma_{ij} \). Moreover, let \( \rho_{ijrs} \) represent the correlation coefficient between travel times on arcs \((i, j), (r, s) \in A\). Our aim is to find a set of \(|K|\) routes where both the total expected travel times and the total variance are minimized in such a way that each customer is visited exactly once, each route starts and ends at the depot, and vehicle capacity is respected for each route. Let \( R^k \) be the set of feasible routes for vehicle \( k \in K \). For each route \( p \in R^k \) let \( \mu^k_p = \sum_{(i,j)\in p} \mu_{ij} \) be the expected travel time of route \( p \). Introducing binary variables \( z^k_p \) with value 1 if route \( p \in R^k \) is assigned to vehicle \( k \in K \), we formulate the VRPSCT as the following mean-variance model:

\[
\text{MP1}_\alpha: \min \quad (1 - \alpha) \sum_{k \in K} \sum_{p \in R^k} \mu^k_p z^k_p + \alpha \sum_{k \in K} \sum_{(i,j) \in A} \sum_{(r,s) \in A} C_{ijrs} x^k_{ij} x^k_{rs} \\
\text{s.t.} \quad \sum_{k \in K} \sum_{p \in R^k} a_{ip} z^k_p = 1 \quad \forall i \in V_c \\
\sum_{p \in R^k} b_{ijp} z^k_p = x^k_{ij} \quad \forall k \in K, (i, j) \in A \\
\sum_{p \in R^k} z^k_p = 1 \quad \forall k \in K \\
z^k_p \in \{0, 1\} \quad k \in K, p \in R^k \\
x^k_{ij} \in \{0, 1\} \quad \forall k \in K, (i, j) \in A
\]

for some \( 0 \leq \alpha \leq 1 \), where \( a_{ip} \) (resp. \( b_{ijp} \)) takes value 1 if route \( p \) visits customer \( i \in V_c \) (resp. arc \((i, j) \in A\)) and 0 otherwise and \( C_{ijrs} = \rho_{ijrs} \sigma_{ij} \sigma_{rs} \) for each pair of arcs \((i, j), (r, s) \in A\) with \((i, j) \neq (r, s)\). Observe that coefficients \( C_{ijrs} \) form a positive semidefinite matrix, implying that the objective function (1) is convex. Finally, it should be noticed that the vehicle index \( k \) in formulation \( \text{MP1}_\alpha \) is necessary to account for the fact that only arcs travelled by the same vehicle contribute to the quadratic component of the objective function.
3 A branch-price-and-cut algorithm

We develop a BPC algorithm to solve $MP_1\alpha$. Considering that $MP_1\alpha$ has an exponential number of binary variables, the continuous relaxation cannot be solved directly. Hence, an iterative column generation procedure is applied in such a way that at each iteration a restricted master problem is solved. Given a solution of the restricted master problem, we verify its optimality by solving a subproblem searching for routes with negative reduced cost. Let us consider the continuous relaxation of $MP_1\alpha$ restricted to a subset of the routes in $\bigcup_{k \in K} R^k$. This problem is a linearly constrained convex quadratic program that can be solved by commercial software packages. Let $(\bar{z}, \bar{x})$ be the optimal solution of this problem. Taking into account the convexity of the problem, the associated reduced costs $\bar{\mu}_k^p$ of a given route $p \in R^k$ can be computed as follows:

$$\bar{\mu}_k^p = (1 - \alpha)\mu_k^p - \sum_{i \in V_c} \pi_i a_{ip} + \sum_{(i,j) \in A} \sum_{(r,s) \in A} 2\alpha b_{ijp} C_{ijrs}^k x_{rs}^k - \beta^k.$$

where $(\pi, \beta)$ are the Lagrangian vectors corresponding to constraints (2) and (4). For each $k \in K$, we should minimize (7) over the set of feasible routes. This can be done for each $k \in K$ by solving the classical elementary shortest path problem with resource constraints.

4 Reformulation based on eigenvalue decomposition

As previously noticed, the matrix $C$ of the covariance coefficients is positive semidefinite. It is well known from the theory of linear algebra that a positive semidefinite matrix can be decomposed as $C = U^T U$ where $U \in \mathbb{R}^{\kappa \times m}$ and $\kappa$ is the rank of $C$. In matrix notation, the quadratic component of the objective function (1) is $(X^k)^T C X^k = (X^k)^T U^T U X^k = (Y_k)^T Y_k$, where $UX^k = Y_k$, and $Y_k \in \mathbb{R}^{\kappa \times 1}$ for each $k \in K$. Model $MP_1\alpha$ becomes:

$$MP_1\alpha(R) : \min \ (1 - \alpha) \sum_{k \in K} \sum_{p \in R^k} \mu_k^p z_{ip}^k \ + \ \alpha \sum_{k \in K} \sum_{\ell \in T} y_{\ell k}^2 \ \text{s.t.} \ \ (2) - (6)$$

$$\sum_{(i,j) \in A} U_{\ell,ij} x_{ij}^k = y_{\ell k} \ \ell \in T, \ k \in K. \ (8)$$

where $T = \{1, 2, \ldots, \kappa\}$. By projecting out the $x$ variables, we obtain:

$$MP_1\alpha(R) : \min \ \sum_{k \in K} \sum_{p \in R^k} \mu_k^p z_{ip}^k \ + \ \sum_{k \in K} \sum_{\ell \in T} y_{\ell k}^2 \ \text{s.t.} \ \ (2), (4), (5)$$

$$\sum_{p \in R^k} \sum_{(i,j) \in A} U_{\ell,ij} b_{ijp} x_{ip}^k = y_{\ell k} \ \ell \in T, \ k \in K. \ (9)$$

The algorithm developed in Section 3, with minor modifications, can be adopted to solve the above reformulation. Depending on the rank value of matrix $C$, model $MP_1\alpha(R)$ might be easier to solve than $MP_1\alpha$ (if $\kappa < m$). However the main advantage of model
5 Computational results and conclusions

We performed a preliminary computational experience on a modified data set from the class C101 of the Solomon’s database, where time windows have been ignored and the number of vehicles is fixed to the minimum feasible value. For space limitations, we omit the details of the instance generation, but we remark that matrices $C$ were dense and with full rank, which makes the resulting optimization problems very difficult. Table 1 reports a comparison between the exact algorithm of Section 3 and the approximate approach of Section 4 where different cardinalities of the subset $\bar{T} \subset T$ have been considered. In particular, in the preprocessing phase, we selected the $|\bar{T}|$ components corresponding to the largest eigenvalues. We observe that the exact BCP algorithm is able to solve instances with up to 30 nodes. The approximate algorithm obtained the optimal solution for all instances whose optimal value was known ($Gap_u = 0$), while the provided lower bounds were tight ($Gap_l \leq 1.40\%$). Furthermore, the approximate algorithm performed fairly well on larger instances. We are currently investigating the possibility to embed the approximate algorithm in a row-and-cut generation scheme to obtain an exact method.

References


Routing mobile medical facilities using transactional data

Burcin Bozkaya and Cemre Gokalp
School of Management
Sabanci University, Istanbul, Turkey
Email: bbozkaya@sabanciuniv.edu

F. Sibel Salman
Department of Industrial Engineering
Koc University, Istanbul, Turkey

Eda Yucel
Department of Industrial Engineering
TOBB University of Economics and Technology, Ankara, Turkey

1 Introduction

We study a problem related to the delivery of mobile medical services on a regularly scheduled basis. We consider the problem from the perspective of the mobile service provider company with the aim of providing the most revenue to the company by maximizing the reach to potential customers, while limiting the transportation costs at the same time. The problem involves deciding where the mobile units will be positioned on each day of the planning horizon so that the service can be brought closer to potential customers determined with respect to where they have made medical expenditures in the past. The schedule consisting of the days in the planning horizon is expected to be repeated so that a visit frequency will arise according to the length of the planning horizon. A score is assigned to each potential stop based on the density of the medical expenses around it and it is assumed that a vehicle captures a portion of the score of nearby regions in addition to the full score of a visited point. The objective is to maximize the sum of the scores captured, either fully or by partial coverage, subject to a limit on total travel cost.
The maximum coverage objective was first used in facility location problems and later on incorporated into routing problems, where vehicle tours are constructed by selecting which points to stop by such that a demand location is covered if it is within a prespecified distance of a point in one of the tours. Two competing objectives are maximizing the total covered demand and minimizing the total tour length. Current and Schilling [2] defined the Covering Salesman Problem (CSP) that minimizes the total cost of the tour of a single vehicle such that every vertex is within a prespecified distance from the tour. Several studies generalized the CSP with multiple vehicles. Current and Schilling [3] introduced the Maximal Covering Tour Problem (MCTP) that maximizes the total demand within a prespecified travel distance from a tour stop. Ozbaygin et al. [5] maximized the amount of demand covered subject to a distance limit as in MCTP. Demands of vertices on the tour are fully covered while only a percentage of the demand of a vertex is covered if it is within a specified distance of some tour stop. The coverage function in our problem resembles the one in [5], yet we solve a multi-vehicle problem. Our coverage function is as defined in [4] for facility location. That is, the score of an unvisited point decreases with respective to the distance from a visited point and is set to zero after a distance limit. Here we allow multiple coverage of an unvisited point by visits in the vicinity. Our problem generalizes the MCTP since we construct multiple tours to maximize the total coverage (score values) while limiting the total traveling cost by a constraint. As such, our problem is in the class of vehicle routing problems with profits (see [1] for a review) and closely related to the Team Orienteering Problem, where a fixed prize is collected from the visited vertices. In our problem the prize collected by locating a mobile unit at a node depends on the locations of the other mobile units as well.

A novel aspect of our study is that the locations to be visited and covered, as well as the coverage amounts (scores) are determined by an analysis of customer credit card transaction data coming from a bank. Hence we feed our optimization model with parameters extracted from big data. This aspect, in addition to the ones discussed above, makes our study unique compared to the existing related studies.

2 Description of the model and a summary of the analyses

We formulate a mixed integer program to maximize the prize (score) captured by the routes of a fleet of medical vehicles (mobile units) over a planning horizon considering full and partial coverage from nearby districts. The model decides the visit schedules and the routes of the vehicles over the planning horizon. Days are divided into weekdays and weekends and the set of stop locations differ according to these two. A score is determined for each location. Full and partial coverage distance limits are defined for each location. A partial coverage ratio is determined for each location pair a priori as a function of their
distance and their partial coverage distance limits so that the ratio is higher if the distance is smaller. There exists a cost of travel between each pair of locations and a total cost limit. The model is formulated with four types of binary decision variables where the one with the highest number of indices has four indices. Continuous variables are also defined to keep track of the scores collected. Constraints include those defined for each pair of locations, each vehicle and each day. The model can be solved in reasonably small run times and the run times increase when the total cost limit becomes tighter, the number of days in the schedule and the number of vehicles increase. We solved the model with 158 locations and up to four vehicles and four weeks to optimality using CPLEX 12.6.2.

We aim to guide the operational decisions on identifying the routes of a mobile medical service operating in Istanbul. We used one-year credit card transactions of customers of a large bank in Turkey in the specific category of medical expenses, specifically, dental and eye exam expenses, to extract information about the distances that the bank customers travel from their home and work locations to the merchants at the points of transaction. The mobile medical service provider can determine the fleet size, identify the service locations to be visited by its vehicles on weekends and weekdays, decide the visiting frequency, i.e., the periodic visiting horizon in terms of weeks, and determine the vehicle routes for that visiting horizon by solving the model.

We analyzed 454,170 credit card transactions related to medical service expenses of 165,307 unique customers living in Istanbul. The transactional data given to us include the date of the transaction together with the working status (i.e., student, retired, or working), home, and work addresses of the customers and the address of the merchant. The origin address of a transaction is defined as the work address of the customer for working customers on weekdays and the home address of the customer for non-working customers on all days and for working customers on weekends. Then, for each transaction of interest, the distance traveled for the medical service is found as the distance between the origin address of the transaction and the merchant’s address.

Next we identified the potential districts (38 districts excluding the islands). In order to identify the service locations for each district for weekdays and weekends, separately, we consider the weekday and weekend transactions, respectively, that have an origin address in that district and we calculate the number of transactions and the mean and standard deviation of the traveled distances. The business and entertainment places, main parking areas, public transportation points, restaurants, ATMs, and common public areas are selected as potential service locations for all days. For weekdays, schools and bank branches are also included in the set of potential service locations. We assume that the mobile medical service provider wants to show up at at most four locations in each district, therefore, through k-means algorithm with \( k = 4 \) we calculated the number of clusters, i.e., the number of service locations, and the places of service locations for each district for
we weekends and weekdays.

We generated the actual road distances between the service locations by taking the shortest path lengths in the street map of Istanbul, using ArcGIS package. In order to estimate a priority score for a weekday or weekend service location, first, a score is estimated for each district for weekdays and weekends, separately, then, the weekday (weekend) score of the district is distributed among the weekday (weekend) service locations within that district. The corresponding score of a district is proportional to the percentage of traveled distances originating from that district in all weekday (weekend) transactions. As service locations within a district are obtained through k-means clustering algorithm, where each service location corresponds to a cluster center; the score of a service location is estimated as the product of its cluster weight in its district and the district’s weekday (weekend) score.

We analyze four coverage models with different settings of full and partial coverage to see the effect of partial coverage. We also analyze the effects of traveling cost limit on the total covered score to observe the trade-offs. Targeting a high visit frequency, which leads to a short visiting horizon, requires a larger fleet. We observe that even with two vehicles and two weeks of visiting horizon, the service provider can achieve a total coverage percentage of 96.8%. If the service provider has two vehicles, visiting horizons of three and four weeks generate nearly the same coverage percentage, thus, a visiting horizon of three weeks is suggested since the show up frequency is larger in that case. Another observation is that the full coverage percentage generally increases and the partial coverage percentage generally decreases as the fleet size increases.

References


A set partitioning heuristic for the home health care routing and scheduling problem

Florian Grenouilleau
Département de mathématiques et génie industriel,
Polytechnique Montréal, Montreal, Canada

Antoine Legrain
Department of Industrial and Operations Engineering,
University of Michigan, Ann Arbor, United-States

Nadia Lahrichi
Département de mathématiques et génie industriel,
Polytechnique Montréal, Montreal, Canada
Email: nadia.lahrichi@polymtl.ca

Louis-Martin Rousseau
Département de mathématiques et génie industriel,
Polytechnique Montréal, Montreal, Canada

1 Introduction

Home health care services improve patients' quality of life by helping them remain independent and in their own homes, often surrounded by family and friends, while maintaining their regular habits. From a governmental point of view, home care services decrease hospital congestion by freeing up hospital beds, which also results in reducing costs for these institutions (4). In 2012, in Canada, more than 2.2 million people received home care services (8).

In this paper, we investigate the home health care routing and scheduling problem (HHCRSP) within a Canadian context. The problem is in determining the assignment of a set of home visits to a set of caregivers over the course of a week and the routing of these caregivers’ workdays. The HHCRSP can be described as a multi-depot vehicle routing problem with time windows and time-dependent travel issues. Moreover, the home care context adds constraints focusing on the caregivers’ skills and the patients’ requirements (both mandatory and optional), as well as the management of the caregivers’ work time contracts. Finally, the HHCRSP has a major concern which is the continuity of care, corresponding to the upkeep of a strong patient-caregiver relationship.
Recently, (1; 3) presented two reviews of the HHCRSP. To cope the low scalability issues of the exact approaches, methods based on heuristics have been developed using frameworks such as large neighborhood search (LNS) (2) or two-phase algorithms (5).

In this work, we present a set partitioning heuristic (SPH) based on the heuristic concentration principle. The goal of our SPH is to solve a set partitioning formulation of the HHCRSP using the columns (feasible routes) generated by a LNS (7). Due to the necessity to produce high quality solutions in a small computational time, the SPH solves a linear relaxation of the set partitioning formulation and a constructive heuristic is then applied to build an integer solution based on the solution found.

2 Problem Definition

We propose to formulate the HHCRSP as a set partitioning problem (SPP) that aims at selecting the best routes for each caregiver among a set Ω of daily feasible caregivers’ routes. Each route ω ∈ Ω takes into account the patients’ mandatory requirements, the forbidden caregivers, the caregivers’ skills, the time-windows, and the time-dependent travel times. The SPP’s objective function comprises the cost of the selected routes, the cost of the unscheduled visits and the penalties related to the overtime or idle for each caregiver. Each feasible route is assigned a length, a travel time and number of missing optional expertises for the visited patients. The cost $c_\omega$ of each route $\omega \in \Omega$ is defined as a weighted sum of the soft constraints’ penalties. The constraints of the set partitioning model ensure that each patient is visited at most once per day and the required number of time over the week and compute the overtime or idle time associated to each caregiver according to his/her worktime’s contract.

3 Solution Method

The proposed SPH is a matheuristic based on the resolution of the SPP. This method is based on the heuristic concentration principle (6). The aim of the heuristic concentration is to keep the best solutions found by a heuristic procedure and then use a set partitioning that combines parts of these solutions to create a better one. We use LNS to find the possible SPP’s routes.

Due to the computational time required to solve the SPP with a great number of routes, the SPH solves the relaxation of this model by relaxing the integrity of the ”selected-route” variables. Then, a constructive heuristic (HeurSPP) is applied and a new integer solution is built based on the relaxation’s result. Moreover, to extend the LNS’ search space in addition of the classic operators, we have developed five new operators (3 for the destruction and 2 for the repair). These new operators are based on
elements such as the flexibility of the patients, the dual values associated with some SPP’s constraints or the random selection of the patient to remove/repair.

4 Computational Results

This section presents two sets of computational experiments: first, we compare our $SPH$ (the new operators ($NW$) and $Heur_{SPP}$ the constructive heuristic) in comparison with a method using a classic LNS with the classical operators ($CL$) on generated instances (three sets (Small, Medium, Large) of 20 instances); then, we analyze the improvements permitted by our $SPH$ on real instances (four of four weeks) provided by Alayacare, our industrial partner.

At each LNS’ iteration, we destroy between 2% and 5% of the scheduled visits and the size of a segment is $10^3$ iterations. Finally, the penalties’ weights have been fixed after preliminary evaluations in collaboration with Alayacare.

According to table 1, we observe that, on average, the new operators ($CL + NW$ scenario) find better solutions. Furthermore, we observe that the $Heur_{SPP}$ ($CL + NW + Heur_{SPP}$ scenario) shows improvements of 16%.

On the real-world instances, we focus here on two major indicators, the total travel time ($TT$) and the continuity of care ($CC$).

<table>
<thead>
<tr>
<th>Instance</th>
<th>Current solution</th>
<th>$SPH$’s solution</th>
<th>$\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$TT$ $CC$</td>
<td>$TT$ $CC$</td>
<td></td>
</tr>
<tr>
<td>154_325_11_40</td>
<td>4361.16 60%</td>
<td>2431.62 75.94%</td>
<td>-44.24% +15.94%</td>
</tr>
<tr>
<td>141_340_11_40</td>
<td>4549.03 62.33%</td>
<td>2833.18 79.05%</td>
<td>-37.72% +16.72%</td>
</tr>
<tr>
<td>148_311_11_35</td>
<td>3832.94 71.69%</td>
<td>2571.29 85.98%</td>
<td>-32.92% +14.29%</td>
</tr>
<tr>
<td>150_324_11_40</td>
<td>3686.57 64.43%</td>
<td>2464.22 82.10%</td>
<td>-33.16% +17.67%</td>
</tr>
<tr>
<td>Mean</td>
<td>4107.43 64.61%</td>
<td>2575.08 80.77%</td>
<td>-37.01% +16.16%</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the actual solutions with those produced by our approach

According to the Table 2, our approach improves the solutions both in terms of travel time and continuity of care. On average, the proposed algorithm reduces the total travel time by 37.01% and increases the continuity of care by 16.16%.
5 Conclusion

The HHCRSP is a complex problem due to the simultaneous management of the assignment (requirements, skills, continuity of care, forbidden assignments) and routing (travel time, work time contracts, impact of the traffic) constraints. Nevertheless, we have proposed a set partitioning heuristic able to cope with all these requirements.

According to the results, we observed that the new operators and the constructive heuristic permit a dramatic reduction in terms of solutions’ costs for the generated instances (respectively 13.76%, 20.82% and 14.39% for the small, medium and large sets). On the real instances, the algorithm permitted, on average, a 37% reduction in travel time and a 16% increase in the continuity of care.

References


A matheuristic for the Biomedical Sample Transportation Problem with Interdependent Pickups

Marta Toschi, Ettore Lanzarone
Consiglio Nazionale delle Ricerche (CNR), Milan, Italy

Ana María Anaya-Arenas, Valérie Bélanger, Angel Ruiz
Centre interuniversitaire de recherche sur les réseaux des entreprises, la logistique et le transport, Montréal and Québec, Canada
Email: AnaMaria.AnayaArenas@cirrelt.ca

1 Introduction

We study a Laboratory Network, formed by a set of sample collection centers (SCCs), visited by patients, and a central laboratory (lab). Although large hospitals often own onsite laboratories, most of the SCCs in a network are not equipped for performing samples’ analyses. Therefore, given specific collection periods at the SCCs, one or several pickups need to be scheduled to take the samples to the lab. This is one of the many challenges faced by our partner, Quebec’s Ministry of Health and Social Services (i.e. the MSSS), who seeks to improve their service to the population and the system’s efficiency by optimizing the sample transportation plan. To create routes that ensure safe and timely shipment of all samples over the province is an interesting, yet difficult, problem to solve.

Inspired by the MSSS context, we study the Biomedical Sample Transportation Problem (BSTP), a Vehicle Routing Problem (VRP) in healthcare logistics, which is challenging due to the short lifetime of samples. According to their biological characteristics and the type of tests to be performed, a sample has a maximum timespan to be analyzed before it perish (i.e. a sample has to be taken to the lab no later than $T_{max}$ minutes after been collected). This short lifespan causes a interdependence between all the pickups and the routing decisions. In addition, this interdependence is also related to the beginning of the collection period of each center (i.e. when the first sample is collected) and the number of visits to be scheduled.

VRP with time restrictions have been studied for a long time in the literature (e.g. VRP with time windows or the pick up and delivery problem). However, fewer contributions consider temporal precedence constraints and/or synchronization between visits, which
are common in healthcare logistics [1]. To the best of our knowledge, only two papers consider interdependence between the pickups and the routing decisions like is done in this paper. Doerner et al. [2] proposed a model to build interdependent routes when collection periods are known whereas Anaya-Arenas et al. [3] introduced some flexibility to collection periods, but imposing a fixed number of pickups at each center and considering softer time restrictions than the ones faced in practice. Although the latter showed the benefits of introducing flexibility in the SCCs’ opening schedule, there is still a need for a model that will address simultaneously several aspects of the real problem. Hence we propose in this study to model the BSTP as a VRP with interdependent pickups, in which both the number of visits to be performed at each SCC and the beginning of their collection period is optimized along with the routing decisions. This is a much more interesting, but also harder, problem to solve, and we propose a matheuristic algorithm to tackle the real instances provided by our partner.

2 Mathematical model

The proposed VRP with interdependent pickups is defined over a complete digraph \( G = (V, A) \), where \( V = \{v_0, v_1, \ldots, v_P\} \) is the set of nodes composed by the \( P \) potential pickups over all the SCCs and the laboratory \((v_0)\), and \( A \) is the arc set. We define \( N = 1, \ldots, n \) as the set of indexes of the SCCs, \( Q = 1, \ldots, P \) as the set of indexes pickups and \( I_g \) as the set of indexes of pickups associated to SCC \( c_g \). \( T_{i_{\text{max}}} \) denote the maximum time limit to analyze the samples collected in pickup \( i \), \( T_{\text{max}} \) the maximum over all \( T_{i_{\text{max}}} \), and \( \delta_g \) the difference between them (if any). Finally, \( O_g \) is the length of the collection period of SCC \( c_g \), \( \alpha_g \) is the current beginning of \( c_g \)'s the collection period and \( \phi_g \) is the flexibility granted to this decision.

As in the classical VRP formulation, arc variables \( x_{ij} \) are set to one if pickup \( i \) is performed right before pickup \( j \). However, to introduce the flexibility with respect to the number of pickups and SCCs’ opening schedule, new variables are defined. The first group of variables is related to pickups: binary variables \( y_i \) are activated only if the pickup \( i \) is performed, and binary variables \( z_i \) are activated only if the pickup \( i \) is performed after the end of the collection period and no other pickups are required at the SCC (i.e. \( y_{i+1} = 0 \)). The second group is related to SCCs’ schedule and timing of pickups: decision variables \( a_g \) and \( b_g \) are the beginning and end of the collection period for SCC \( c_g \), variable \( u_k \) correspond to the time that pickup \( k \) is performed, and \( f_k \) is the maximum remaining time to bring the samples picked up at node \( i \) back to the lab. The objective is to minimize the total route duration. Besides the classical constraints sets for the VRP, we formulate the following constraints to ensure the precedence and the maximum timespan restriction.
on the samples life utility.

\[
\begin{align*}
\alpha_g - \phi_g &\leq a_g \leq \alpha_g - \phi_g & g \in N \\
b_g &= O_g + a_g & g \in N \\
b_g - u_k &\leq -M(z_k - 1) & g \in N, \quad k \in I_g \\
u_k - b_g &\leq 0 & g \in N, \quad k \in I_g \\
\sum_{k \in I_g} z_k &\geq 1 & g \in N \\
y_k &= 1 - z_{k-1} & k \in I_g \setminus \{I_g\}_1, g \in N \mid |P_g| > 2 \\
f_0 &\geq +f_i - M(1 - x_{i0}) + t_{i0} + \tau_i & i \in Q \\
f_j &\geq +f_i - M(1 - x_{ij}) + u_j - u_i & i \in Q, \quad j \in Q, \quad j \neq i \\
f_0 &= T_{\text{max}} & \\
f_k &\leq T_{\text{max}} & k \in Q \\
f_k - (u_k - a_g) &\geq \delta_k & k = (I_g)_1, g \in N \\
f_k - (u_k - u_{k-1}) &\geq \delta_k & k \in I_g \setminus \{I_g\}_1, g \in N \mid |P_g| > 2
\end{align*}
\]

Constraints (1) and (2) are related to the flexibility in the opening time for each SCC. Constraints (3)-(6) ensure that the entire operation period of each SCC is covered and no samples are left at the SCCs. Constraints (7)-(10) control the timing over all pickups and, finally, constraints (11) and (12) ensure that the time available for transportation and the time samples are kept in the SCC respect the maximum time limit. The corresponding model was solved with CPLEX version 12.6.0.0. However, the solver was not able to tackle the real-data instances efficiently (within a CPU time of 10 hours or a memory limit of 50GB), which motivated the development of an approximated method.

## 3 Matheuristic

We propose a matheuristic with two main phases. The construction phase is first applied to obtain a feasible solution. Due to the strong interdependence and our large-size instances, we propose a **clustered-based decomposition** to create subproblems that can be solved to optimality by CPLEX in a short computational time. We hence adapt a *k*-means algorithm, with a relevant distance function that considers both spatial and temporal compatibility, and define an appropriate number of clusters using iteratively the elbow method for each instance. After each subproblem has been solved, the solutions are merged to find a first feasible solution for the global problem. This solution will be improved in the second phase through a **fix-and-optimize variable neighbourhood search**. Different attributes of the solution are considered to define three neighbourhood structures: the route structure, the the pickup time and the number of pickups. To explore
4 Numerical Results

We compile 25 real-data instances provided by our partners from eight service areas in the province of Quebec, each one with different schedules, geographical density and distribution. We report in Table 1 preliminary results for four instances, reporting their size in number of SCCs (\(N\)) and potential pickups (\(P\)); CPLEX objective function (\(\text{Obj}\)), computational time in seconds (\(\text{CPU}\)), memory consumption (\(\text{GB}\)), and optimality gap (\(\text{Gap}\)); and finally, the results of our algorithm: objective function (\(\text{Obj}\)), computational time in seconds (\(\text{CPU}\)) and the gap to CPLEX best known solution (%\(\text{BKS}\)), for both its first solution (found with the clustering method) and the final one. As it is shown in Table 1, CPLEX fails to solve our problem efficiently, even for small instances. However, we have encouraging results with our algorithm, which is able to improve CPLEX best solution (found after five hours) by more than 3% (in average) and this in less than an hour. Notice that even if we were inspired by Quebec’s BSTP, our algorithm could be later applied to other context as VRP with perishable items or a generalization of the Dial-and-Ride Problem.

References


A Combinatorial Benders’ Cuts based Exact Method for the multi-trip Containers Drayage Problem

Maurizio Bruglieri
Dipartimento di Design
Politecnico di Milano, Milano, Italy

Simona Mancini
Dipartimento di Matematica e Informatica
Università degli Studi di Cagliari, Cagliari, Italy

Ornella Pisacane
Dipartimento di Ingegneria dell’Informazione
Università Politecnica delle Marche, Ancona, Italy
Email: simona.mancini@unica.it

Keywords: Set Partitioning, Containers Drayage, multi-trip Vehicle Routing, Pickup and Delivery

1 Introduction

Containers Drayage refers to the movement of containers between terminals (e.g., sea ports, intermodal terminals, inland ports, border points) and customers. Two types of services may be requested by customers: import and export. Import customers require the delivery of containers from the terminal, while export customers require a pick-up of containers to the terminal, [1]. The most used containers are of 20ft and 40ft size. Moreover, the European Community regulation imposes that either two containers of 20ft or one of 40ft can be loaded onto a truck. Therefore, we can distinguish the services required by the customers into 4 groups: pickups of 20ft and 40ft containers; deliveries of 20ft and 40ft containers. In addition, the loading restrictions impose that the maximum number of customers that can be served in a single trip is equal to 4. A trip defined as a path starting from the terminal, serving a set of customers and returning to the terminal. This fact, combined with the observation that the inland destinations are usually not very far from the terminal, yields to trips with a limited duration. Therefore, it is
realistic to consider that the trucks may perform more than one trip during a working day. In the literature, the Container Drayage Problem (CDP) has been addressed under the assumption that each truck performs a single trip. The aim of this work is to extend the CDP to the more realistic case in which multiple trips for truck are allowed.

2 Problem Description

We introduce the multi-trip CDP, (MCDP), that aims to route a fleet of $m$ trucks, minimizing the total travel distance such that the liner shipping companies can save their money providing more sustainable transportation services. A route is performed by one truck within a maximum duration $T_{max}$ and serves a subset of customers. It is then seen as either a single trip or the composition of more trips. A trip starts from the terminal and returns to it after handling a subset of customers. Unlike the traditional multi-trip Vehicle Routing Problem, in the MCDP, due to the constraints on the truck capacity and the maximum number of customers for trip, the number of feasible trips is somehow limited. Indeed, there are only two feasible combinations of 4 nodes: $0 \rightarrow -20 \rightarrow -20 \rightarrow +20 \rightarrow +20 \rightarrow 0$; $0 \rightarrow -20 \rightarrow +20 \rightarrow -20 \rightarrow +20 \rightarrow 0$. In analogous way, it is easy to see that there are six feasible combinations of 3 nodes, seven of 2 nodes and four single node trips. Therefore, the total number of feasible trips is only 19. Figure 1 shows a MCDP feasible solution for an instance with 16 customers who request of: delivering 40ft containers (blue circles); picking 40ft containers up (yellow circles); delivering 20ft containers (red circles) and finally, picking 20ft containers up (black circles). The green triangle denotes the terminal. The feasible solution contains two trips with 4 nodes, two trips with 3 nodes and finally one trip with 2 nodes. Moreover, the two trips indicated by red dashed arcs are the route of a truck; the two trips denoted by black dashed arcs are the route of a second truck and finally, the trip represented by blue dashed arcs is the route of a third truck.

Figure 1: A feasible solution for the drayage problem.
3 The Combinatorial Benders’ cuts based Exact Approach

In this work, we firstly model the MCDP through a trip based formulation (TB-MCDP). Then, we present a Combinatorial Benders’ Cuts (CBCs) approach to solve the TB-MCDP.

3.1 A Trip-Based mathematical formulation for the MCDP

The MCDP is represented on an undirected graph $G = (V, E)$ with $V = C \cup \{0\}$ where $C$ is the set of the customers and 0, the terminal. For each $[i,j] \in E$, the travel time and the travel distance are known. For each $c \in C$, the type of service required and its service time are given. We exhaustively generate $K$, the set of all the non-dominated feasible trips. A trip $k$ is feasible if its duration $t_k \leq T_{max}$ where $t_k$ takes into account both travel and service times. A trip $k$ dominates a trip $w$ if they handle the same customers and $t_w > t_k$. Finally, $d_k$ indicates the length of the trip $k$ while the set $J$ denotes the $m$ trucks of the fleet. The MCDP is modeled as a TB-MCDP through the following decision variables: $X_k$ equal to 1 if the trip $k \in K$ is selected, 0 otherwise; $Y_{kj}$ equal to 1 if the trip $k \in K$ is performed by the truck $j \in J$, 0 otherwise.

$$\begin{align*}
\min \sum_{k \in K} d_k X_k \\
\sum_{k \in K} a_{ck} X_k &= 1 \quad \forall c \in C \\
\sum_{j \in J} Y_{kj} &= X_k \quad \forall k \in K \\
\sum_{k \in K} t_k Y_{kj} &\leq T_{max} \quad \forall j \in J
\end{align*}$$

The objective function (1) consists in minimizing the total travel distance. Each customer has to be handled in exactly one trip (2); only the selected trips can be assigned to trucks (3) and each selected trip must be assigned to one and only one truck. Finally, the total duration of trips assigned to the same truck cannot exceed $T_{max}$ (4). Constraints (5) and (6) specify the variables domain.

3.2 Combinatorial Benders’s Cuts for the TB-MCDP

In a CBCs approach, [2], the problem is split into a Master Problem (MP) and a Slave Problem (SP). The MP can be either a relaxation or a simplified version of the problem and involves a subset $S$ of variables. In the SP, instead, the value of the variables previously optimized in the MP is maintained fixed while the remaining ones are optimized. If the
SP is infeasible, a *Combinatorial Benders cut* is added to the MP in order to prevent the variables in $S$ assuming simultaneously the same values they achieved in the optimal solution of the MP. Otherwise, the obtained solution is optimal. In the literature, this approach has been applied to solve Mixed Integer Programming (MIP) problems in which the MP contains the integer variables while the SP becomes a linear programming problem. But, it can be generalized to cases in which even the SP is a MIP problem. The main issue of the CBCs approach is to properly define both the MP and the SP such that they can be quickly solved to the optimality since they must be iteratively solved for several times during the optimization process. For applying the CBCs approach to the TB-MCDP, we define the MP as a relaxation of the problem that does not consider any limitation on the number of trucks, while simply aims finding the best set $K'$ of trips covering all the customer requests. This problem can be formulated as a classical Set Partitioning problem, obtained from TB-MCDP model by removing constraints (3)-(4). The only variables involved in the MP are the $X$ ones. Instead, in the SP, the $Y$ variables are introduced in order to assign the trips to the trucks. The SP aims to find if, given the subset of selected route $K'$, a feasible assignment to the truck exists, such that all the trips in $K'$ are assigned to a truck and the maximum duration of each route is respected. If the SP is feasible, then the solution obtained is optimal. Otherwise, the following CBC is added to the MP in order to prevent all the trips in $K'$ from being simultaneously selected:

$$\sum_{k \in K'} X_k \leq |K'| - 1 \quad (7)$$

4 Results and Conclusions

We introduced the MCDP that extends the classical CDP by considering that each truck may perform more than one trip during the working day. This assumption properly describes situations coming out from the reality. We modeled the MCDP as TB-MCDP and we proposed for it a CBCs approach. Preliminary comparisons, between solving directly the TB-MCDP and using the CBCs approach, obtained on instances with different layouts and features, show a significant reduction of the computational times.

References


Integrated Scheduling of Drayage and Long-haul Transportation in Synchromodality

Arturo E. Pérez Rivera and Martijn R.K. Mes
Department of Industrial Engineering and Business Information Systems,
University of Twente, Enschede, The Netherlands
Email: \{a.e.perezrivera,m.r.k.mes\}@utwente.nl

1 Introduction

We study the integrated scheduling of drayage and long-haul transportation in synchromodality. Synchromodality is a management paradigm where the combination of transport modes to bring a freight from its origin to its destination is not fixed up front, but decided at various moments using the latest status of the transportation network and freight demand. Traditionally, drayage and long-haul have been scheduled separately, which does not take advantage of the flexibility in synchromodality. Consider a Logistic Service Provider deciding on which of two terminals to bring a freight to for the start of the long-haul. If the focus is on drayage, the closest terminal to the origin of the freight is most likely to be chosen, even if the furthest terminal has better consolidation opportunities for the long-haul. However, if the focus is on the long-haul, the terminal with the best consolidation opportunities is most likely to be chosen at the expense of drayage operations. The integration of separate scheduling activities using up-to-date network information is key to improve the performance of intermodal transportation [1].

Scientific research about the integration of scheduling drayage and long-haul transportation has been limited [2], although potential cost savings have been identified. Studies about scheduling drayage, such as [3], usually model the richness of constraints in the drayage process but do not include the impact/interaction with the long-haul. Studies about scheduling freight in long-haul transportation, such as [4] and [5], usually include flexibility in the selection of the initial terminal for freight but do not explicitly model the richness of drayage operations and the interaction between the two decision processes. Studies that consider the interaction between drayage and long-haul scheduling decisions are scarce and usually focus on specific applications. For these reasons, we believe the contribution of our work is two-fold. First, we propose and evaluate various forms of integrating separate models/heuristics for scheduling drayage and long-haul transportation. Second, we assess the performance of these integrated approaches against separated scheduling under various cost settings.
2 Model Formulation and Solution Approach

We consider the stochastic optimization problem of scheduling drayage and long-haul freight transportation over a finite horizon \( t \in T \). Each day \( t \), random freights \( F_t \) arrive for drayage. A drayage schedule \( x_D^t \) is built for these freights with costs \( z_D^t(x_D^t) \). This schedule includes: (i) the routing of trucks that pick-up freight and execute other drayage operations, and (ii) the assignment of a terminal to freight that continue on the long-haul. We refer to these latter freights, together with freights in the terminals delivered by the drayage earlier and not yet sent on the long-haul, as long-haul freights \( f_L^t \) (see Figure 1). A long-haul schedule \( x_L^t \) is built for these freights with costs \( z_L^t(x_L^t) \). This schedule includes the selection of freights to consolidate/postpone at each long-haul terminal and service.

We consider that all drayage freights must be transported to a terminal on the day of arrival, but that at the terminals they may be postponed for their long-haul.

\[
\begin{align*}
\min_{\pi \in \Pi} & \ E \left[ \sum_{t \in T} (z_D^t(x_D^t, \pi) + z_L^t(x_L^t, \pi)) \right] \\
\text{subject to} & \ f_L^0, P^D, \Gamma
\end{align*}
\]

The goal is to minimize the total expected costs in (1), where \( x_D^t, \pi \) is a drayage schedule dependent on a long-haul policy \( \pi \in \Pi \), \( f_L^0 \) represents the initial long-haul freights at terminals, \( P^D \) describes the stochastic arrival process of freights for drayage (i.e., \( P^D \rightarrow F_t \)), and \( \Gamma \) is a function that defines the long-haul probabilities from the drayage decisions. In previous work, we modeled the drayage part as a full-truckload pickup-and-delivery problem with time-windows (FTPDPPTW) [6] and the long-haul part as a Markov Decision Process (MDP) model [7]. Although these are separate models, they have mechanisms that can connect them directly and indirectly as follows. The FTPDPPTW model includes an assignment cost that depends on long-haul freights at each terminal and the assignment decision of freights picked-up. Thus, a direct connection to the long-haul performance is possible. The MDP model includes an indirect connection to the drayage through the arrival probabilities of freight at the origins of the long-haul modes (i.e., terminals). This means that the arrival of freight to the long-haul modes not only depends on \( P^D \), but also on the decisions of the drayage \( x_D^t \). In the following, we describe how the connection between the separate models can be realized.
To connect the FTPDPTW and MDP models, we propose a sequential and an iterative integration approach. In *sequential integration*, the overall distributions of freight arrivals are split among the long-haul modes based on what we typically expect from the drayage operations. These new distributions are used in the MDP model, and the optimal values of each state are returned to the FTPDPTW model in the form of costs. In *iterative integration*, a simulation of the drayage and long-haul scheduling under sequential integration is done, and the arrival rate of freight per long-haul mode is measured. These measurements are used to re-define arrival distributions, repeatedly, until a stopping criterion is met.

![Diagram](image)

**Figure 2:** Integration of heuristics for scheduling drayage and long-haul transportation

Due to computational complexity of the models, we use a Math-Heuristic (MH) for the FTPDPTW and Approximate Dynamic Programming (ADP) for the MDP. The integration of these heuristics can be seen in Figure 2. The MH uses various cuts based on the assignment cost resulting from the Value Function Approximation (VFA) of ADP. In turn, ADP learns the VFA based on the observed arrivals from the simulation and the use of the MH, in case of the integrated approach. The details on how the MH and ADP work separately can be found in [6] and [7]. In Figure 2, we observe two challenges: the definition of arrival distributions per long-haul mode and the stopping criteria. In the first challenge, the overall probability distributions must be mapped to the long-haul modes based on drayage scheduling. The second challenge involves the assessment of when the VFA to pass to the MH is good enough, i.e., results in stable drayage and long-haul scheduling decisions.

### 3 Preliminary Results and Conclusions

To explore the performance of our proposed integration, we use two instances. The setup in these instances is the same, with the exception of the drayage and long-haul costs. Each instance contains three long-haul modes, 25 drayage trucks, 10 possible origins, 12 possible destinations, and a horizon of one week. Demand and time-window settings are based on the experiments considered in [6] and [7]. Instance 1 has drayage costs that are approximately 90% of total costs. Instance 2 has drayage costs that are approximately 10% of the total costs.

We test two versions for Sequential Integration, which we call A and B. In A, we split all probability distributions evenly across the terminals. In B, we split them according to our best estimate. For the Iterative Integration, we start with an even split of probability
distributions such as in A, and a fixed stopping criterion of three iterations. In addition, we test a separate scheduling method combining the benchmark heuristics used in [6] and [7]. We show the results for all experiments in Table 1.

Table 1: Comparison of costs across different scheduling methods

<table>
<thead>
<tr>
<th>Scheduling method</th>
<th>Instance 1</th>
<th>Difference</th>
<th>Instance 2</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate Benchmark</td>
<td>79,414</td>
<td>-</td>
<td>102,822</td>
<td>-</td>
</tr>
<tr>
<td>Sequential Integration A</td>
<td>79,732</td>
<td>0%</td>
<td>103,522</td>
<td>1%</td>
</tr>
<tr>
<td>Sequential Integration B</td>
<td>78,971</td>
<td>-1%</td>
<td>94,752</td>
<td>-8%</td>
</tr>
<tr>
<td>Iterative Integration</td>
<td>78,813</td>
<td>-1%</td>
<td>96,585</td>
<td>-6%</td>
</tr>
</tbody>
</table>

In Table 1, we observe that the savings of an integrated approach, with respect to separate scheduling, range between 1% and 8%. The largest savings are in Instance 2, in which long-haul costs are higher than drayage costs. We also observe that Sequential Integration B is better than A, meaning that splitting the overall probability distributions has an impact on the integration quality. Furthermore, Sequential Integration B outperforms Iterative Integration in Instance 2. This is related to the challenges discussed in Section 2.

Finally, our preliminary results show opportunities for the integration of drayage and long-haul scheduling in synchromodality. However, we observed that research on what conditions to use for initializing probability distributions for the long-haul and what criteria to use for stopping the iterative approach is necessary. In ongoing experiments, we are studying (i) mechanisms to define arrivals for long-haul modes, and (ii) convergence rates of observed arrivals and performance under various stopping criteria.

References

Optimizing barge utility in hinterland container transportation

Bernard Zweers
Stochastics,
CWI, Amsterdam, The Netherlands

Sandjai Bhulai
Department of Mathematics,
VU Amsterdam, Amsterdam, The Netherlands

Rob van der Mei
Stochastics
CWI, Amsterdam, The Netherlands
Email: zweers@cwi.nl

1 Introduction

Using barges in hinterland container transportation has a lot of advantages compared with using trucks. Transportation by barge is cheaper and more eco-friendly than trucking a container and it does not contribute to road congestion. Nevertheless, in practice we do not see the modal shift from trucks to barges as one would expect. Two main reasons for this, are that large quantities of containers are needed for a barge to outperform truck in terms of costs and that barges are often delayed, because of the congestion at deep-sea terminals. To overcome these issues a good planning of the barges is required.

We present a real-life operational planning problem encountered by an inland terminal and solve this problem with an integer linear program (ILP) and a two-stage heuristic. The inland terminal need to ship containers located at multiple deep-sea terminals in one port to one inland terminal. The containers can either be transported by barge or by truck, but the goal is to ship as many containers as possible by barge and visit as few deep sea terminals with a barge. Major factors in deciding which barge to use for which containers are the demurrage costs and holding costs. Demurrage costs are the costs that need to be paid to store a container at the deep-sea terminal. Holding costs incur when a container has to be stored at the inland terminal before it can be shipped to its final destination. Furthermore, we impose a penalty for visiting a deep-sea terminal with a barge. All in all, a trade-off has to be made between the barge utility, the demurrage and
holding costs and the number of terminals that are visited by barge.

As pointed out by [3], operational planning problems in multimodal freight transporta-
tion has received little attention. In the literature that discusses operational problems (e.g. [1], [2]) the routes of the barges are determined. In our model, we do not need construct a route, because the available time slots for a barge at a terminal are usually that restrictive that only a few routes are feasible and a very good route can easily be found.

2 Solution Approach

2.1 Integer linear program

For every container \(i = 1, \ldots, n\), we know the estimated arrival date \((ETAi)\) at the deep-
sea terminal and the call date \((CDi)\) at which it should be at the customer. Moreover, every container has a size in TEU \((\text{teu}_i)\) and a deep-sea terminal at which it is located \((Ti)\). Each container can be assigned to a truck \((v = 0)\) or one of the \(b\) barges \((v = 1, \ldots, b)\). Each barge \(v\) has a maximum capacity in TEU \((C_v)\) and a minimum load \((L_v)\). The day a barge \(v\) is at the deep-sea port is denoted by \(day_v\), which is before the end of the planning horizon \((t_{max})\). We assume that there is an unlimited amount of trucks and that each truck can only ship one container. The demurrage and holding costs are for every \((i, v)\) denoted as, respectively, \(dm_{iv}\) and \(hc_{iv}\). The costs of shipping a container with barge \(v\) is given by \(sc_v\) and the costs of transportation by truck by \(sc_0\). The visit of terminal \(r\) with barge \(v\) is penalized with \(tv_{rv}\).

\[
\begin{align*}
\min & \quad \sum_{v=1}^{b} \left( sc_v \left( C_v - \sum_{i=1}^{n} \text{teu}_i X_{iv} \right) \right) + \sum_{i=1}^{n} s_{c0} X_{i0} + \sum_{v=0}^{n} \sum_{i=1}^{n} \left( dm_{iv} + hc_{iv} \right) X_{iv} + \\
& \quad \sum_{v=1}^{b} \sum_{r=1}^{R} tv_{rv} Z_{rv},
\end{align*}
\]

such that:

\[
\begin{align*}
L_v & \leq \sum_{i=1}^{n} \text{teu}_i X_{iv} \leq C_v & & v = 1, \ldots, b; \\
\sum_{v=0}^{b} X_{iv} & = 1 & & i : CD_i \leq t_{max}; \\
\sum_{v=0}^{b} X_{iv} & \leq 1 & & i : CD_i > t_{max}; \\
X_{iv} & \leq Z_{rv} & & \forall i \& v = 1, \ldots, b \& r = Ti; \\
X_{iv} & = 0 & & \forall i \& v = 1, \ldots, b : day_v < ETA_i \text{ or } day_v > CD_i; \\
X_{iv} & \in \{0, 1\} & & \forall i \& \forall v; \\
Z_{rv} & \in \{0, 1\} & & \forall r \& v = 1, \ldots, b.
\end{align*}
\]
The objective function (1) consists of four sums. The first sum ensures that the utility of the barges is maximized, the second sum minimizes the transportation costs by truck, the third sum the holding and demurrage costs and the fourth sum the penalties paid for visiting a terminal by barge. The constraint (2) ensures that the capacities of the barges are satisfied. Constraints (3) and (4) make sure that if a container has a call date before the end of the planning horizon, it definitely is transported and otherwise it might be transported. Constraints (5) and (6) force that a container cannot be transported by a barge that is not visiting the terminal or which day is before the ETA or after the call date of the container.

2.2 Heuristic method

For large real-life instances, the running time of the ILP is sometimes be too long to be used in practice, thus we also present a heuristic based on the ILP. As there are many more containers than terminals, the number of Z-variables in the ILP is small compared with the number of X-variables. Therefore, the number of integrality constraints decreases significantly if the X-variables are relaxed. This gives rise to the following two-phase heuristic:

Step 1: Solve the ILP with relaxing the constraints in (7), i.e., $0 \leq X_{iv} \leq 1$. Let $\bar{Z}$ be the optimal outcome of the Z-variables.

Step 2: Fix $\bar{Z}$ and force all $X_{iv}$-variables to be binary and solve the remaining ILP.

In other words, this heuristic determines in Step 1 which terminals are visited, given fractional assignments of containers. In Step 2, an optimal allocation of the containers to barges and trucks, given the set of terminals that are visited, is found which is an application of the Generalized Assignment Problem (GAP). In [4], it is shown that for a linear relaxation of the GAP, the number of fractional assignments is at most the number of machines scheduled to minimum or maximum capacity. Since the trucks have no capacity the number of containers that are split between two or more barges or a barge and a truck is at most $b$. Since the number of barges is rather small compared to the number of containers, the solution after Step 1 is almost feasible and is a lower bound of the optimal solution. Consequently, Step 2 of the heuristic usually produces quickly a solution that has a value close to the optimal solution.

2.3 Computational results

We have tested the ILP and the two-stage heuristic using CPLEX on twelve real-life instances with about 750 containers and five barges. The heuristic produces for ten instances the optimal solution and for the other two it costs are at most 0.03% higher than the optimal solution. The running time of the ILP for eleven instances was only a few seconds, but the heuristic was on average a factor of two faster. For one instance the ILP needed
45 minutes to be solved, while the heuristic produced the same solution in a few seconds. If we double the number of barges, one instance takes more than eight hours to be solved by the ILP, but less than thirty seconds for the heuristic. For the other instances, the heuristic is on average more than ten times faster than the ILP and the solution is at most 0.5% of the optimal solution.

3 Conclusion

We present an ILP-model that can be used to solve a real-life operational problem encountered by an inland terminal. In current practice the planning is made by hand and implementing the ILP could lead to a cost reduction of about 20-30%. The ILP makes a trade-off in maximizing barge utility, minimizing demurrage and holding costs and minimizing the number of terminals visited with a barge. The balanced solution leads to a more reliable and economic efficient solution. A drawback of the ILP is that its computation time might be too long. Hence, we have developed a two-stage heuristic that produces (almost) optimal solutions and that is better scalable than the ILP.

References


An Analysis of Territory Design For Routing Problems with Time Windows and Balance Requirements

Francesco Bertoli, Philip Kilby
CSIRO, Data61 and the Australian National University (ANU)
108 North Road, Acton ACT 2601, Australia
Email: francesco.bertoli@data61.csiro.au

1 Introduction

Transport companies often face the tactical problem of designing a strategy to effect regular deliveries in a service area over a certain horizon. Even though, theoretically, one could solve a set of vehicle routing problems associated with each day, this might not be the most efficient solution from both practical and strategical points of view. Often, companies wish to pre-assign customers, or sectors of the service area, to drivers. This way a driver visits the same customers and drives within the same area every day. This is desirable to enhance both consistency of service, and thus customers’ satisfaction, and to increase drivers’ familiarity with their service area, which significantly increases efficiency [7].

One common approach [3] is to split the service area in several sectors (or clusters of customers) called territories, to be pre-assigned to drivers. This implicitly achieves consistency and decreases the routing complexity of future daily routing problems. However, almost all works in the literature do not explicitly consider the routing component of the problem. Recently, some authors have shifted their attention to methods including the routing decisions [6, 1]. This has several advantages. For instance, it allows us to consider complex operational constraints, such as time windows, which greatly affect the design of effective territories. The approach proposed in this work considers routing decisions and uses these in order to design effective territories.

The problem addressed in this paper is motivated by a real-world application from a grocery delivering company operating in Australia. We present a general approach and an extensive computational analysis. The goal of the present work is not only to solve the problem at hand, but also to investigate how much different factors affect the design of effective territories. We isolate and study how several instances’ features (such as
time windows density and width, demand variation) influence the territories design. More importantly, we include balance requirements in our method. These play a key role in many real-world problems [2] and cannot be neglected. To the best of our knowledge no routing-based method in the literature directly considers balance requirements. Furthermore, we analyse how a routing based method performs on a problem which is not embedded in a Euclidean plane. Lastly, we compare our results with a “master routes” algorithm that was used in a previous collaboration with the same industrial partner.

2 Problem Formulation

We consider a time horizon made of several days $D$, a set of customers $C$, a single depot and a set of $K$ homogeneous vehicles. Each customer may place some requests during the horizon. Moreover, associated with every customer there is a time windows and a service time, which are the same every day. Every route must start and end at the depot and has to respect the depot’s time window.

A solution is defined by three components: a partition of $C$ into $|K|$ territories, and a set of $|K|$ routes for each day, satisfying capacity and time windows constraints, and a one-to-one assignment of routes to territories. We point out that we do not enforce all requests to be satisfied. We are also given a route measure $M(k)$ for route $k$, that could represent, for instance, its duration or value. The $M$-gap of a solution on a given day is $\max_k(M(k)) - \min_k(M(k))$.

The cost of a solution is a weighted sum of the following components: the total distance travelled, the sum of $M$-gap over all days, the number of requests that are not satisfied and the number of violations of the partition. A violation occurs if a route $r$ visits a customer $c$ but $r$ and $c$ are assigned to different territories. The goal is to find a solution minimising the cost. Note that we take a soft approach. Instead of imposing consistency of service we penalise the violations of the partition.

We note that our formulation has the advantage of balancing the routes, rather than the territories. Most balancing measure are route-defined (e.g., workload or profit). Balancing these measure without explicitly considering the routes can only be done using estimates, which often are unreliable.

3 Solution Method

The proposed heuristic algorithm is simple, flexible and easy to extend to other operational constraints. We refer to our algorithm as Routing Based Territory Approach (RBTA).

RBTA is based on the Adaptive Large Neighbourhood Search scheme (ALNS) [5]. ALNS has been proved to be a very efficient and flexible technique when applied to routing problem [4]. In ALNS, an initial solution is gradually improved by alternately destroying
and repairing the solution. ANLS uses one out of a number of destroy, respectively repair, heuristics. These are chosen randomly, progressively adjusting the probability of selecting a method, based on how well it has performed previously. In order to escape local minima a Simulated Annealing scheme is used. Our method differs from ALNS in several aspects. Besides a few ad hoc destroy and repair heuristic, there are two main differences: the cost computation and a specific operator we apply after a fixed number of operation to eliminate symmetry. In particular, we progressively increase the weight of some cost components, both in the algorithm and in each repair phase.

4 Computational Analysis

We have available some data from our industrial partner. Due to legal restrictions, we cannot publish the data. However, we created 26 instances that we make available. To evaluate the quality of a solution we use several statistics. The routing quality is measured by the average over all days of the distance travelled (TD) and the number of requests that were not satisfied (U). The degree of balance is measured by the average, over all days, of the $M$-gap. The service consistency is measured by reporting the average over all customers of: the number of different drivers visiting a customer (DD), the proportion of days a customer is served by the driver visiting it the most times (CF).

We present 4 experiments. For each one we run our algorithm on all instances and for each statistics, in Table 1, we report the average over all instances. In experiment 1 and 2 we solve each day separately, i.e., we solve $|D|$ independent single-day routing problems. Conversely, in experiments 3 and 4, we consider the whole problem, therefore penalising the violations as described above. Moreover, in experiments 1 and 3, we do not consider any balance measure, while in experiments 2 and 4, we use the duration of a route as balance measure $M$.

From the results, it is clear how our algorithm is effective both in reducing the balance gap and increasing service’s consistency.

<table>
<thead>
<tr>
<th></th>
<th>TD</th>
<th>$M$-gap (%)</th>
<th>CF</th>
<th>DD</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>experiment-1</td>
<td>548.63</td>
<td>-</td>
<td>0.71</td>
<td>2.07</td>
<td>0.0</td>
</tr>
<tr>
<td>experiment-2</td>
<td>604.07</td>
<td>0.04</td>
<td>0.64</td>
<td>2.33</td>
<td>0.02</td>
</tr>
<tr>
<td>experiment-3</td>
<td>570.57</td>
<td>-</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>experiment-4</td>
<td>700.9</td>
<td>0.05</td>
<td>1.0</td>
<td>1.02</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Table 1 The averaged results for all experiments. We do not report $M$-gap for the experiments where no balance measure $M$ is involved.
5 Conclusion

In this work, we propose a general method to solve territory design problems for routing applications. We analyse both the performance of the method, the effectiveness of the approach and how this is affected by operational constraints, such as time windows and other features of the instance at hand. In a significant contribution, we consider balance requirements in our problems, and analyse their interplay with the consistency requirement. The method is easily extendable to consider other routing constraints.

References


Multi-directional local search for the leximax-VRP

Fabien Lehuédé
IMT Atlantique, LS2N UMR CNRS 6004, Nantes, France
Email: fabien.lehuede@imt-atlantique.fr

Olivier Péton
IMT Atlantique, LS2N UMR CNRS 6004, Nantes, France

Fabien Tricoire
Institute for Production and Logistics Management,
Johannes Kepler University Linz, Austria

Most vehicle routing problems (VRPs) studied in the Operations Research literature deal with the design of a set of routes of minimal cost to serve a set of customers. In many practical cases, companies seek cost minimization as well as the optimization of criteria related to vehicles and drivers. In particular, preserving equity among drivers through a good balance of their workload is often sought. Historically, two main equity measures, have been proposed in the VRP literature. First, min-max approaches, such as proposed by Golden et al. [1], propose to minimize the length of the longest route. An unfortunate consequence of this approach is that all solutions with the same longest route length have the same equity measure. The second equity measure, recently designated by the term range [4], considers the difference between the longest route and the shortest route [3]. An important drawback of this measure is its non-monotonicity. As recently exposed by Halvorsen-Weare and Savelsbergh [2] or Matl et al [4] among other drawbacks: the range of a solution can be improved by increasing, even in an inconsistent way, the length of its shortest route of a solution without decreasing the length of the others.

As recently summarized by Ogryczak et al. [5] in a survey on fair optimization and networks, the lexicographic minimax approach has been used in domains such as network optimization, facility location and network optimization to produce equitable or fair solutions. The lexicographic minimax, based on the leximax operator, refines the min-max approach: informally speaking, when a minimal value has been found for the longest route, the lexicographic minimax considers the second longest route, the third longest route, and so on, until all ties have been broken.

In this presentation, we introduce the leximax-VRP as a refinement of the CVRP with
two objective functions: the sum of routing costs and the lexicographic minimax over routes durations.

1 Solution method

We propose to integrate the lexicographic minimax approach in a multi-objective optimization framework called Multi-Directional Local Search (MDLS) [7].

1.1 MDLS principle

MDLS offers a very simple local search framework but it still competes with state-of-the-art methods when solving multi-objective optimization problems. In MDLS, a local search \( LS_j \) is defined for each objective \( j \). This local search is later performed in order to improve solutions with respect to objective \( j \). A set of non-dominated solutions is kept in an archive and returned at the end of the algorithm. An iteration consists in (i) selecting a solution from the archive, (ii) performing local search on this solution for each objective/direction, thus producing a new feasible solution in each direction and (iii) updating the archive using newly produced solutions.

1.2 Local search components

In our algorithm, we consider that local search consists of one Large Neighborhood Search (LNS) iteration. Several ruin and recreate operators are defined for each objective. Hence, at each iteration, for each objective (i) a ruin and a recreate operator are randomly selected in the set of operators for that objective and (ii) a new solution is produced using the selected operators.

The cost operators are defined according to the classical LNS operators for the VRP [6]. The set of ruin operators that we use are: random removal, worst removal, related removal and route removal. The recreate operators for the cost objective are the cheapest insertion heuristic and the k-regret heuristic for \( k = 2, 3, 4 \).

We introduce lexicographic minimax operators which extend the classical operators to the lexicographical minimax approach. The ruin operators include the random removal and the related removal as well as the following two operators:

- worst max removal: at each iteration this operator removes, from the longest route, the customer that decreases the most the length of this route.

- longest route removal: at each iteration, all customers from the longest route are removed.

Two sets of recreate operators have been designed to guide the search towards lexicographic minimax efficient solutions:
Figure 1: Boxplots charts for the number of solution of the reference front that have been found and within 2% distance for each run and each instance in one hour.

- The \textit{leximax cheapest insertion} and \textit{leximax k-regret} extend the cheapest insertion and k-regret heuristics to the lexicographic minimax approach.

- The \textit{min-max cheapest insertion} and \textit{min-max k-regret} extend cheapest insertion and k-regret, but using only the duration of the longest route as a criterion to guide the search and the solution cost increase to break ties.

As the lexicographic minimax heuristics involve sorting route vectors to compare solutions, our objective is to assess whether the faster min-max heuristics can be more efficient to guide the search towards lexicographic minimax solutions.

2 Experiments

To illustrate the experimental study in this abstract, we present the comparison of three configurations, which all include cost operators and lexicographic minimax ruin operators. Configuration \textit{leximax} integrates the \textit{leximax cheapest insertion} and \textit{leximax k-regret}. Configuration \textit{max} integrates \textit{min-max cheapest insertion} and \textit{min-max k-regret}. Configuration \textit{all} includes all of these recreate operators for the lexicographic minimax objective.

Our MDLS is evaluated on the Christofides CVRP instances, which are traditionally used to benchmark the VRP with load balancing. 10 runs of 60 minutes are performed for each configuration and each instance and a reference set is constructed by taking the non-dominated union of the sets returned by each run for each configuration over all experiments. To evaluate configurations, we consider two indicators: the percentage of
solutions from the reference front found in a given run, and the percentage of solutions found which are within 2% of a solution from the reference front. A solution \( x_1 \) is within 2% of another solution \( x_2 \) if, when the cost and all route lengths of \( x_2 \) are multiplied by 1.02, then \( x_1 \) dominates this transformed solution. This information is summarized in the two plots from Figure 1. On the left plot, for each instance and each configuration, we represent the percentage of solutions of the reference front that are found on each run. The distribution of the performance evaluation of each run is displayed with a Box Plot. For the majority of instances, the percentage of solutions of the reference front found for each run remains quite low. Comparing the three configurations, it is not possible to clearly state that one dominates the other. The right plot shows the percentage of solutions of the reference front that lie within a 2% distance of a solution returned by each run. This plot shows that the returned approximation returned by the all and lexicmax configurations remains within close distance of the reference front. Comparing the various configurations, guiding the search with insertion criteria based on the longest route is less effective than with the proposed lexicmax based heuristics.

References


Workload Equity in Vehicle Routing: 
The Impact of Alternative Workload Resources

P. Matl  
Department of Business Administration,  
University of Vienna, Vienna, Austria

R.F. Hartl  
Department of Business Administration,  
University of Vienna, Vienna, Austria

T. Vidal  
Department of Computer Sciences  
Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil  
Email: piotr.matl@univie.ac.at

1 Introduction

Workload equity and balanced resource utilization are becoming increasingly important in real-world logistics systems. This stems from the recognition that logistics is not exclusively cost-driven, and that the marginal cost of more balanced operations can be offset by gains through lower overtime hours, higher employee satisfaction, better customer service, and more efficient use of available capacity. These considerations are relevant in a wide variety of practical vehicle routing problems (VRPs), including service technician routing, vendor-managed inventory systems, waste collection, parcel delivery, public transportation, and volunteer organizations, among others (cf. the survey in [1] for a detailed overview and analysis). The heterogeneity of these applications is reflected in the many different forms of “balance” objectives which have been proposed.

The implications of choosing one balance objective over another have only recently started to be explored. Abstracting from the particularities of specific applications and considering instead prototypical VRPs, several studies have examined how the structure of optimal VRP solutions changes depending on the chosen balance criterion, in single-objective ([2, 3]) as well as in bi-objective contexts ([4, 1]). All of these articles conclude that the choice of a balance criterion has a significant effect on the resulting VRP solutions, and that this choice should therefore not be made arbitrarily.
However, the generality of the above studies is restricted. Specifically, only a particular class of balance criteria has thus far been examined – namely those balancing tour distances. Yet the articles reporting on practice also describe real-world VRPs in which alternative definitions of workload are more appropriate, e.g. the number of stops in small package delivery, service time in technician routing, and load/demand in groceries distribution. In this work we generalize and extend previous studies to multiple workload resources, and re-assess accordingly previous conclusions and guidelines for formulating a balance objective.

2 Analytical Study

Based on the work of [1], we categorize balance objectives according to:

- the workload resource which is to be balanced, and
- the equity function used to quantify the degree of balance.

Workload resources are either constant-sum or variable-sum, depending on whether the total workload to be distributed is the same for all solutions, or not. Equity functions can be classified in various ways, and we identify two characteristics which directly interact with the type of workload resource – monotonicity, and compatibility with the Pigou-Dalton transfer principle.

Based on this analysis, we point out some general guidelines for formulating a balance objective. They can be summarized as follows:

1. For the general case of variable-sum resources, the monotonicity property is critical for the choice of the equity function – a non-monotonic equity function may imply a preference even for workload allocations in which all outcomes are worse than those of another feasible allocation. This contradicts the minimization of each workload, leading to unintended optimization outcomes. On the other hand, the transfer principle is of only limited importance: it will apply only in the comparatively rare case when two competing allocations have the same sum of outcomes.

2. For the special case of constant-sum resources, the monotonicity property is irrelevant, because every increase in one outcome necessitates a decrease in at least one other. The Pareto-optimality of every allocation is therefore guaranteed regardless of the selected equity function. In contrast, the transfer principle is more relevant, because the constant-sum character of the resource guarantees that all allocations are connected by mean-preserving transfers and thus potentially comparable.
3 Numerical Study

Despite the generality of the analytical guidelines, models satisfying them can still differ significantly in ways which do not lend themselves to a purely analytical treatment. We therefore extend our analysis with a numerical study examining all combinations of three potential workload resources – distance/duration, load/demand, and stops/customers – and six alternative equity functions – minmax, lexicographic minmax, range, mean absolute deviation, standard deviation, and the Gini coefficient.

We consider the bi-objective problem of optimizing both cost and balance in order to generate and analyse the types of compromise solutions which would be found in practice with standard constraint-based or weighted-sum approaches. In addition, by conducting our analysis separately on smaller instances solved to optimality, and then on larger instances solved with a state-of-the-art VRP heuristic, we investigate the extent to which previous observations and conclusions still hold as instance size grows, which has thus far not been considered by previous studies. Our observations can be summarized as follows:

**Number of Compromise Solutions.** More complex equity functions provide more potential compromise solutions, as does balancing distance compared to load, and load compared to stops. In practice, if balance considerations are handled with constraints, then a larger set of compromise solutions increases the likelihood that a solution close to a specific constraint value actually exists.

**Marginal Cost of Balance.** The trade-off between cost and balance depends primarily on the workload resource, and only to a lesser extent on the equity function. However, the marginal cost of balance was found to be low for all examined combinations of resource and function – near-optimal balance could be achieved within 5% to 10% of the cost-optimum – and it was observed to decrease with instance size.

**Model Overlap.** For a given resource, most of the solutions identified by an equity function are not unique to that function and found by at least one other alternative, i.e. various combinations of balance criteria generate at least some identical compromise solutions. However, no single equity function finds much more than half of all the potentially “well-balanced” solutions for a given resource (the superset of solutions identified by all considered functions).

**Model Agreement.** It seems intuitive that even if different models do not produce identical solutions, there might be at least some degree of agreement between them as to which VRP solutions are “well-balanced”, and which are not. We find that this is only partly the case. Solutions which are well-balanced in terms of one resource are usually not
well-balanced in terms of another. However, for the same resource, solutions optimizing one of the examined equity functions tend to be of high quality also for the other functions. This underlines the importance of choosing the proper workload resource for the given application – once a resource has been selected, the equity function has limited impact.

**Solution Similarity.** Most multi-objective methods – and particularly genetic algorithms – implicitly assume that Pareto-optimal solutions share common characteristics, especially if they have similar objective function values. We confirm that this intuition largely holds in the context of balanced VRPs. For each instance, the pairs of solutions closest to each other in the objective space were found to have a median of 75% to 80% of their edges in common, compared to only 55% for random pairs. This lends some credibility to methods which assume some common structure among high-quality solutions.

## 4 Conclusion

From a managerial point of view, we have found that balance – in many different forms – can be significantly improved at only limited extra cost, allowing the non-monetary benefits of balance to be realized in practice. However, we do stress that certain types of functions (non-monotonic) are not compatible with certain types of resources (variable-sum), as they lead to unintended optimization results. Overall, our study emphasizes the importance of selecting the correct resource to balance. As outlined above, we observed that the subsequent choice of the equity function has relatively less impact on the solutions found – a conclusion which counter-balances the focus of previous studies which considered thus far only the choice of the equity function.

## References


A Heuristic Algorithm for the Periodic Rural Postman Problem with Irregular Services

Enrique Benavent, Ángel Corberán
Department of Statistics and Operational Research,
University of Valencia, Valencia, Spain

Demetrio Laganà
Department of Mechanical, Energy and Management Engineering,
University of Calabria, Arcavacata di Rende (CS), Italy
Email: demetrio.lagana@unical.it

Francesca Vocaturo
Department of Economics, Statistics and Finance “G. Anania”
University of Calabria, Arcavacata di Rende (CS), Italy

1 Introduction

Periodic routing problems are traditionally classified in two classes: periodic vehicle routing problems (PVRPs), where the entities requiring a service are located on the vertices of the graph representing the street network, and periodic arc routing problems (PARPs), where these entities are located along the links (arcs or edges) of the graph. PVRPs have been studied extensively. On the contrary, the scientific literature on the PARPs is still scarce.

Recently, Benavent et al. [1] have introduced and studied a specific PARP named periodic rural postman problem with irregular services (PRPP-IS) by referring to a mixed graph. The problem consists of designing a set of minimum-cost vehicle routes, one for each day of a given time horizon, satisfying the service requirements. Specifically, the requirements concern some links of the graph (required links) that must be traversed a specified number of times in some sub-periods of the time horizon. In order to explain the irregularity plainly, the authors refer to a time horizon including 7 days (a week) and to a graph where, e.g., some required links must be traversed once during the first four days of the week and once during the weekend (from Friday to Sunday). Note that a route associated with a specific day may serve no required link (i.e., may be empty), for reasons of convenience, if this condition is compatible with the service requirements. Some practical applications of the PRPP-IS can be found in road maintenance operations and road network surveillance.
Benavent et al. [1] propose a mathematical model and an exact algorithm based on branch-and-cut to solve the PRPP-IS. An extensive experimental phase shows that their algorithm is effective on instances of small and medium size. These instances were derived by the authors from the mval dataset introduced by Belenguer et al. [2] for the mixed capacitated arc routing problem and include from 24 to 138 required links.

In order to find a solution of good quality for larger instances of the PRPP-IS, we propose a multi-start algorithm based on local search. Before presenting this algorithm, we provide more details on the problem.

2 Problem Statement

Let $G = (V, E, A)$ be a strongly connected mixed graph in which $V = \{1, \ldots, n\}$ represents the set of vertices, including the depot (vertex 1), $E$ represents the set of edges, and $A$ represents the set of arcs. Let $E_R \subseteq E$ and $A_R \subseteq A$ be the sets of edges and arcs requiring a service, respectively. We use $L = E \cup A$ and $L_R = A_R \cup E_R$ to represent the set of all links and required links, respectively. There is a nonnegative cost $c_l = c_{ij}$ associated with the traversal of each link $l = (i, j) \in L$. Moreover, there is a service cost $c^s_l = c^s_{ij}$ associated with each required link $l = (i, j) \in L_R$ (with $c^s_l \geq c_l$). We consider a finite and discrete time horizon $H = \{d_1, \ldots, d_{|H|}\}$, where each $d_h \in H$ represents a period (or day). For each required link $l \in L_R$, there exists a set $P_l$ of disjoint sub-periods, where each sub-period is a subset of $H$. Each sub-period $T \in P_l$ has associated a frequency $f^l_T \leq |T|$ that indicates the number of times that link $l \in L_R$ must be serviced along the days of $T$.

We assume that a required link must be serviced at least once over the time horizon, but it cannot be serviced more than once in the same day.

The PRPP-IS consists of finding a set of routes, one for each day, in such way that the service requirement of each link is satisfied. Let $K$ be the set of all route indices (note that a route index $k \in K$ corresponds to a day $d_h \in H$, and vice versa). The aim of the problem is minimizing the total cost, i.e.,

$$
\sum_{k \in K} \sum_{l \in L_R} (c^s_l - c_l) y^k_l + \sum_{k \in K} \sum_{(i, j) \in E} c_{ij} \left(x^k_{ij} + x^k_{ji}\right) + \sum_{k \in K} \sum_{(i, j) \in A} c_{ij} x^k_{ij},
$$

where $x^k_{ij}$ is a decision variable representing the number of times that link $(i, j) \in L$ is traversed in route $k \in K$ from $i$ to $j$, and $y^k_l$ is a binary variable that takes value 1 if link $l \in L_R$ is serviced in route $k \in K$, and value 0 otherwise.

3 Solution Approach

In order to find a solution of good quality for the PRPP-IS, a heuristic solution approach has been designed. Heuristic methods aimed at finding (near) global optimal solutions
typically require some type of diversification to escape from local optima. One way to achieve diversification is re-starting the solution process, i.e. using a multi-start framework \[3\]. We briefly present a multi-start algorithm for the \textit{PRPP-IS} according to the traditional scheme in which each global iteration consists of two-phases. In particular, initial solutions are sequentially generated through a cluster-first route-second procedure including random elements (\textit{constructive phase}). The initial solutions are improved by using an iterative procedure (\textit{improvement phase}). In this phase, the iterations are called local. Note that each global iteration of the multi-start algorithm yields its own solution (probably, a local optimum) and the best one represents the final output.

\subsection{Constructive Phase}

The steps of the constructive phase are: \(i\) generation of a feasible assignment or clustering, \(ii\) sequencing.

A feasible assignment can be generated by using a greedy procedure that assigns a \textit{score} to each required link. Specifically, for each required link, the score is equal to the minimum number of days included in a sub-period associated with it. Imagine, e.g., that \(H = \{1, 2, 3, 4, 5, 6, 7\}\) and a required link \(l\) is associated with sub-periods \(T = \{1, 2, 3, 4\}\) and \(T' = \{5, 6, 7\}\). The score for this link is equal to \(\text{min}\{4, 3\} = 3\). The assignment procedure sorts the required links by non-decreasing values of score. Note that, generally, the required links with the largest score are the most flexible ones. Before assigning the required links with the largest score, the procedure assigns the other required links to some routes in order to have a feasible service requirements fulfilment. Image, e.g, that for the above-mentioned link \(l\) we have: \(f^l_T = 1\) and \(f^l_{T'} = 2\). Feasibly assigning it to some routes means selecting a route/day in set \(\{1, 2, 3, 4\}\), and two different routes/days in set \(\{5, 6, 7\}\). The selection is based on weighted random numbers. In particular, routes with more elements assigned to them have more chances to be selected. Then, a check on the number of routes with at least an assignment (\textit{active routes}) is made. Let \(\rho\) be this number. In particular, an attempt to “empty” active routes by re-assigning their links is carried out if \(\rho\) is higher than the number of routes needed to serve all required links. On the contrary, empty routes are marked as active if \(\rho\) is lower than this number. Finally, the procedure assigns the links with the largest score to only active routes. The assignment is based on weighted random numbers too but, in this case, routes with fewer elements have more chances to be selected, in order to balance them.

In the second step of the constructive phase, the required links assigned to each route have to be sequenced. Practically, for each day of the time horizon, a \textit{rural postman problem} on \(G\) has to be solved \[4\]. We deal with this problem through a heuristic procedure based on the feasibility pump scheme suggested by Fischetti et al. \[5\]. Very preliminary experiments have been carried out on a small dataset of 34 instances. The results show
that the constructive phase of the proposed algorithm is able to find feasible solutions with an average gap of 5.68 percent of the best lower bound provided by the branch-and-cut designed by Benavent et al. [1].

3.2 Improvement Phase

The improvement phase consists of iteratively changing the current solution by operators opportunely defined within a local search framework. We use both intra-route and inter-route operators to move from a solution for the PRPP-IS to another solution. For instance, we use the $\lambda$-interchange operator (see the article of Osman [6] for a description of the $\lambda$-interchange generation mechanism applied to a different routing problem).

Of course, a feasibility check is needed whenever inter-route operators are applied in order to avoid moves of required links in inappropriate routes/days.

4 Concluding Remarks

The heuristic algorithm was coded in C++ and preliminary experiments were carried out. On the basis of the computational results obtained by referring to instances introduced in [1], the solution approach described in Section 3 seems to be effective. In particular, several optimal or near-optimal solutions were reached.

Future research activities are aimed to strengthen the improvement phase of the algorithm and extend the computational experiments.

References


1 Introduction

The purpose of this study is to solve a multi-period garbage collection problem involving several waste types, where each type is collected by a separate vehicle and a certain coordination of the collection is done. This problem is referred to as the Coordinated Capacitated Arc Routing Problem (C-CARP), and was first presented in [1]. The problem is a variation of the Capacitated Arc Routing Problem (CARP), which was first presented by [2]. [3] provides a thorough survey on the CARP and its variants.

Due to the variations in volumes and nature of the waste types, not all of them require the same collection frequency. For example, while general or organic garbage may require weekly or bi-weekly collection, paper and cardboard may have to be collected every four weeks, glass and metal every three weeks, and plastic every two weeks. Under the current practice, the collection days for the various types are unsynchronized. For example, a given household may have general and organic waste collected every Monday, paper and cardboard every fourth Tuesday, glass and metal every third Thursday, and plastic every second Monday, as illustrated in Figure 1 for a 12 week planning period.

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Week 8</th>
<th>Week 9</th>
<th>Week 10</th>
<th>Week 11</th>
<th>Week 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>General organic</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paper and cardboard</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glass and metal</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plastic</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Example of an inconvenient schedule for the household.
The idea of the C-CARP, is to coordinate the collection of the different waste types such that, seen from the individual household perspective, the frequency of collection may vary, but the collection will always occur on the same day of the week. An example of such a collection schedule, which corresponds to the one shown in Figure 1 is given in Figure 2. As the citizens must move the waste bins to the curbside before the collection, this coordinated schedule has the benefit that the citizens only need be concerned with waste once a week, Monday in our example.

More formally, the C-CARP is defined on an undirected connected graph $G = (N, E)$, where $N$ is the set of nodes and $E$ is the set of edges. A traversal cost $c_{ij} > 0$ is associated with every edge, $(i, j) \in E$. A demand $q_{ij}^f \geq 0$ is associated with every waste type, $f \in F$ and every edge $(i, j) \in E$ such that it must be collected every $l_f$ days. We denote by $\tau$ the number of days in a week.

If no coordination is required, the problem in our example can be solved as four independent CARPs, where the set of routes for general and organic waste is partitioned into $\tau$ sets, one for each day of the week, repeated on a weekly basis, whereas the set of routes for paper and cardboard are separated into $4\tau$ sets of routes, each of which are repeated every four weeks.

In order to describe the coordination, we consider the weekdays $d = 1, \ldots, \tau$. Two situations occur from a coordination perspective. For any edge $(i, j)$, we require that every waste type $f \in F$, which must be collected weekly or less frequently, ($l_f \in \{\tau, 2\tau, \ldots\}$), is serviced on the same weekday $d$. For the more frequent waste types of the edge ($l_f \in \left\{\frac{\tau}{2}, \frac{\tau}{3}, \ldots\right\}$), we require that one of the weekly services occurs on that same weekday $d$, while all other services are not coordinated. We also require that the service of the same waste type on a given edge is performed by the same vehicle.

The primary objective of the C-CARP is to minimize the number of vehicles used which, due to the difference in the frequency of service, is not necessarily the same as minimizing the number of routes, while the secondary objective is to minimize the total routing cost.

The study is part of a larger research project regarding waste transportation in Denmark, where waste collection falls under the responsibility of the counties. We obtained data from six out of 98 counties that represent several areas of Denmark, ranging from rural to urban, and the scale of the graphs is very large.
2 Solution approach

We have developed a heuristic for solving the C-CARP. In the following, we outline our algorithm. A key point in our procedure is to notice that if we first assign edges to weekdays Monday, Tuesday, ..., then the routing of the vehicles each day are relatively independent of each other as long as we are careful regarding the waste types needing service several times per week. In this presentation, we focus on the less frequent waste types.

We aim to create \( \tau \) districts such that for each waste fraction, the number of routes needed to collect the waste in each district is as balanced as possible, and such that the total routing cost over all districts is minimized. The creation of balanced districts means that the total number of vehicles needed for each waste type, which equals the maximum number of routes over the districts divided by the number of weeks between collection, is minimized.

The first phase of our algorithm is thereby the creation of \( \tau \) weekday districts. This is done by a greedy approach, where edges are iteratively assigned to one of \( \tau \) districts initiated by each of their seed nodes. In the construction of the initial districts, we consider both balance of the workload and density in terms of the area covered by the district, though priority is given to the latter to ensure relatively short routes in the district. During this process, we keep track of the boundary nodes of the districts.

Having obtained \( \tau \) districts, we initiate a process where we move individual edges near the boundaries between the districts. To this end, we repeatedly identify a district which requires more than the average number of routes (and more than a lower bound dictated) for at least one waste fraction. We then seek to move one or more edges adjacent to the boundary from that district to a neighbouring district without increasing the number of routes in the receiving district beyond the average (or the lower bound). We repeat this process until we cannot decrease the estimated number of routes any further. Next, we repeatedly identify a district requiring less than the average number of vehicles and seek to move edges from neighbouring districts to this district. The whole process is repeated while improvement in the balancing can be obtained.

After the final districts are created, we solve, for each waste fraction and each district, the resulting CARP using the FastCARP algorithm of [4], which is designed to solve large scale CARPs relatively quickly. The idea of the algorithm is to first create a giant tour, then cut the tour into vehicle routes which are then separated into \( \sqrt{k} \) groups, where \( k \) is the number of routes. Then, the algorithm repeatedly merges two groups, performs some route optimization on the joint set, and separates the two groups again. This process continues in a cyclic manner until a time limit is reached.
We have used the algorithm to solve the instances from the six counties in Denmark. These instances have up to 11656 nodes, 12691 edges, and 8651 required edges. There are five different combinations of collection intervals, with some instances having both frequent and non-frequent waste types and others having only non-frequent types. The vehicle fleet for each type of waste is homogeneous. We compare our results to those obtained by running the same algorithm on each waste type individually, thereby obtaining results for the non-coordinated case.

Acknowledgements

This project was funded by the Danish Council for Independent Research - Social Sciences. Project ‘Transportation issues related to waste management’ [grant number 4182–00021] and by the Natural Sciences and Engineering Research Council of Canada [grant number 2015–06189]. This support is gratefully acknowledged.

References


Bidirectional labelling for the European Union
Truck Driver Scheduling Problem

Asvin Goel
Kühne Logistics University, Hamburg, Germany
Email: asvin.goel@the-klu.org

Christian Tilk
Johannes Gutenberg University Mainz, Mainz, Germany
Email: tilk@uni-mainz.de

Extended Abstract

Approaches for solving vehicle routing problems for long-distance haulage require that for each route generated within the course of the solution process, compliance of the route with applicable hours of service regulations for truck drivers must be validated to ensure the route’s feasibility. The problem of finding a feasible truck driver schedule for a given route is the so-called truck driver scheduling problem (TDSP). For hours of service regulations in the European Union, solving the TDSP is a time-consuming task for which no polynomial complexity bound is known. Current approaches for the EU-TDSP [e.g. 1, 2, 3, 4] rely on forward labelling approaches in which the driver state is represented by a multi-dimensional label and resource extension functions (REFs) are used to update labels with respect to the activities conducted by the driver. Unfortunately, the number of alternative labels which need to be considered by a forward labelling algorithm can grow significantly, resulting in substantial computational effort for creating and storing labels as well as for comparing labels and eliminating those which are dominated. This contribution presents a backward labelling method for generating schedules complying with EU hours of service regulations and shows how the labels generated with the backward method can be combined with labels generated by the forward labelling method presented by [3] in order to obtain a bidirectional labelling method for the EU-TDSP.

According to EU hours of service regulations, a truck driver must not drive for more than four and a half hours without a break and not more than nine hours without a rest. A break is a period of 45 minutes during which the driver does not work. This break can be taken in two parts, the first having a duration of at least 15 minutes and the second of at least 30 minutes. A rest is a period of at least 11 hours duration during which the
driver can sleep. The required rest must be taken within 24 hours after the previous rest. Alternatively, a rest can be taken in two parts, whereas the first part must have a duration of at least three hours and the second of at least nine hours. A detailed overview of the regulations is given in [2]. Figure 1 shows the forward and backward REFs that can be used to account for all driver activities conducted on the trip from one customer location to another. At the start of the trip, the driving time between the customer locations is initialised. Then for each driver activity conducted on the trip, a respective REF is used with a parameter $\Delta$ indicating the duration of the activity. Eventually, when the next customer location is reached, the service time and potential waiting time due to time window constraints are added. The difficulty in solving the EU-TDSP is that neither the sequence in which driver activities should be conducted nor their durations $\Delta$ are known in advance.

An auxiliary network for forward labelling is presented by [4] and it is shown how the best activity durations $\Delta$ can be computed. With this auxiliary network, the EU-TDSP can be solved by a forward labelling approach. The approach has been implemented within a branch-and-price approach for the EU vehicle routing and truck driver scheduling problem (EU-VRTDSP), i.e., a variant of the well-known vehicle routing problem with time windows in which each route must comply with EU hours of service regulations. An extension of this work considering additional national regulations aiming at preventing night work is presented in [3]. The times during which work is considered as night work differs across the member states of the European Union and respective provisions had been ignored in the literature by all previous works. Furthermore, [3] shows how duration-related costs, such as labour costs, can be considered within the labelling approach. However, as the number of non-dominated labels and the respective computational effort grows signif-

Figure 1: Forward and backward REFs for a route segment between two customer locations
significantly when minimising schedule durations, no satisfactory computational results for the EU-VRTDSP were obtained.

This contribution presents a backward labelling approach for the EU-TDSP, appropriate REFs and an auxiliary network for which the best activity durations can be uniquely determined. We show how the backward labels can be combined with forward labels in order to obtain a bidirectional labelling approach. We implement the forward, backward, and bidirectional labelling approaches in an exact branch-and-price-and-cut algorithm for the EU-VRTDSP. The branch-and-price-and-cut algorithm is based on the approach by [5] for United States hours of service regulations and extends this approach in order to consider the optimisation goal of minimising realistic costs composed of distance-related costs, such as fuel costs, and duration-related costs, such as labour costs.

Table 1 shows the results of our computational experiments for 25 and 50 customer instances. The experiments have been performed on a single core of a standard PC with an Intel(R) Core(TM)i7-5930k clocked at 3.5 GHz and a memory of 64 GB. The table contains the number of customers per instance, the time considered as night time by the different national regulations, the number of instances solved to optimality within the run time limit of two hours, and the average runtime in seconds (which is computed only over those instances that are solved to optimality by all three labelling variants). Due to the structural difference in the auxiliary network caused by the asymmetry of the regulations, the backward labelling approach requires significantly more time than the forward labelling approach. However, we can see that the bidirectional labelling approach can solve significantly more of the instances to optimality and the average running time for solving the instances is much smaller than for both uni-directional variants. With bidirectional labelling, additional 28 of 168 instances can be solved to optimality compared to forward labelling, and additional 51 of 168 instances compared to backward labelling. Bidirectional labelling is on average between 2 and 10 times faster than forward labelling and between 4 and 16 times faster than backward labelling.

<table>
<thead>
<tr>
<th></th>
<th>Forward labelling</th>
<th>Backward labelling</th>
<th>Bidirectional labelling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>CPU time (in s)</td>
<td>Solved</td>
</tr>
<tr>
<td>25</td>
<td>51 of 56</td>
<td>409.81</td>
<td>46 of 56</td>
</tr>
<tr>
<td>25</td>
<td>46 of 56</td>
<td>439.40</td>
<td>42 of 56</td>
</tr>
<tr>
<td>25</td>
<td>45 of 56</td>
<td>365.20</td>
<td>41 of 56</td>
</tr>
<tr>
<td>50</td>
<td>23 of 56</td>
<td>340.59</td>
<td>23 of 56</td>
</tr>
<tr>
<td>50</td>
<td>22 of 56</td>
<td>515.87</td>
<td>17 of 56</td>
</tr>
<tr>
<td>50</td>
<td>19 of 56</td>
<td>712.28</td>
<td>14 of 56</td>
</tr>
</tbody>
</table>

Table 1: Comparison of forward, backward, and bidirectional labelling approaches
The bidirectional labelling approach can quickly evaluate routes and thus accelerates the exact branch-and-price-and-cut algorithm. The capability of combining forward and backward labels can also significantly speed up heuristic solution approaches for the EU-VRTDSP. Instead of re-evaluating a modified route from the first change to the end of the route using a forward labelling approach, the bidirectional approach allows to update labels locally, i.e., from the first change until the last change in the route. By merging forward and backward labels stored in appropriate memory structures, a large share of the computational effort required by a uni-directional labelling method can be saved. Therefore, we expect bidirectional labelling to become a standard component in local search based heuristics for the EU-VRTDSP.

References


A Lagrangian Heuristic for
Integrated Timetabling and Vehicle Scheduling

Antonio Frangioni, Laura Galli
Department of Computer Science,
University of Pisa, Pisa, Italy
Email: frangio@di.unipi.it

Samuela Carosi, Leopoldo Girardi, Giuliano Vallese
Department of OR development,
M.A.I.O.R. Srl, Lucca, Italy

1 Introduction

Planning of a public transportation system is a complex process that consists of several phases, such as strategic planning (e.g., network design), tactical planning (e.g., line planning, timetabling), operational planning (e.g., vehicle/crew scheduling) and real time control. Due to the hardness of the problem, these phases are usually performed in sequence, as breaking down the problem into different steps reduces its complexity. However, this is a heuristic approach that determines the number of vehicles and drivers needed only in a late stage of the planning process, while buses and crews are expensive resources that need efficient utilization. Despite numerous studies addressing each of the above steps individually, to our knowledge, the literature for integrated approaches is rather scarce, e.g., [1, 2]. In this paper we propose an approach for simultaneously dealing with timetabling and vehicle scheduling. We present an integer, flow-based formulation of the problem and a heuristic solution approach based on Lagrangian relaxation. We test the approach on real-world instances provided by M.A.I.O.R. (MAIOR), a leading company producing services and advanced software systems to public transport authorities and operators, and we show that the approach provides solutions of better quality than those usually adopted by operators in acceptable running times.

2 Problem description

We describe the timetabling and vehicle scheduling problem that MAIOR addresses. A public transportation network PTN is given, usually under the form of a graph where the
nodes correspond to stops or depots, and the links correspond to direct bus transits. A
line \( l \) is a path in the PTN between two terminus \( A \) and \( B \); generally, lines run in both
directions, i.e., a line is actually a pair of lines (from \( A \) to \( B \) and from \( B \) to \( A \)). The
frequency of a line specifies how often bus service should be offered, as measured by the
ideal time interval between two subsequent transits of buses of the line at a specific point
(stop or terminus), called “pilot node”. The time horizon (one day) is divided into time
slots, and the service frequency typically varies depending on such time slots.

Since line design is a decision at the strategical level, we assume that the set of lines \( L \)
is already given, together with the desired frequencies. Trips are paths, corresponding to
lines in the PTN, which have to be operated by the same bus at a certain time. Each trip
\( c \in C \) is characterized by a start and end time, as well as an arrival and departure time at
each stop of the corresponding line, including the arrival time \( \pi_c \) at the pilot node. The
problem takes as input a set of possible trips \( C = \bigcup_{l \in L} C_l \), partitioned according to the
different lines (\( \bigcap_{l \in L} C_l = \emptyset \)), and it is formed by two parts. The timetabling (TT)
problem consists in selecting, for each line \( l \), a subset of trips \( \bar{C}_l \subseteq C_l \), such that the corresponding
frequencies are as close as possible to the desired ones. The goal of bus scheduling (BS) is
instead to find a cost-minimal assignment between buses and trips such that each trip is
covered by exactly one bus and the schedules of all buses are “feasible”; this entails time
compatibility, i.e., two trips \( c_1 \) and \( c_2 \in C \) can be served by the same bus if the start time
of the second trip is greater than the end time of the first trip plus a minimum stopping
and/or driving time, and the possible impact of regulations.

Overall, this results in the bus routes together with a timetable for each of the buses.
Note that a single bus might very well cover several trips and serve different lines. Therefore,
the goal of the integrated problem is to provide a solution that optimally balances
cost (i.e., minimize the number of buses used) and user satisfaction (i.e., minimize the
distance from the actual frequencies and the desired ones).

3 Solution approach

The TT problem is separable for each \( l \in L \), as frequencies are defined “per line”. From
the trips \( C_l \) we construct a directed graph \( G_{lTT} = (N_{lTT}, A_{lTT}) \), where the nodes correspond
to the trips \( (N_{lTT} = C_l) \) and an arc between two trips \( c_1 \) and \( c_2 \) means that they are
subsequent trips of the line. The cost of the arc \( (c_1, c_2) \in A_{lTT} \) depends on the difference
between the ideal time interval (in the corresponding time slot) and \( \pi_{c_2} - \pi_{c_1} \), with a
simple formula whose details are not crucial. The arc exists only if \( \pi_{c_2} - \pi_{c_1} \) belongs to
a given interval, i.e., pairs of trips “too close” or “too far apart” cannot be chosen. It is
trivial to see that \( G_{lTT} \) is acyclic; therefore, the TT problem can be easily solved, for each
line \( l \), as an acyclic shortest path (SP) problem.
The BS problem is not separable, as a single bus route can cover different lines. We construct a single compatibility graph $G^{BS} = (N^{BS}, A^{BS})$ having two nodes $c^-$ and $c^+$ for each trip $c \in C$, that represent the start and the end of trip $c$, respectively, plus two nodes $O^+$ and $O^-$ representing buses leaving and returning to the depot, respectively. Then, we define four types of arcs: (i) $(c^-, c^+)$, meaning that the corresponding trip $c$ is covered by a bus; (ii) $(c^+, d^-)$, the compatibility arc defined if the same bus route can cover trip $c$ and then trip $d$ in sequence; (iii) $(d^+, O^-)$, meaning that the bus returns to the deposit right after completing trip $d$, and (iv) $(O^+, c^-)$, meaning that the bus starts trip $c$ right after leaving the deposit (and reaching the corresponding terminus). Finally, a return arc $(O^-, O^+)$ is added to define a circulation problem, whose capacity is equal to the fleet cardinality and whose cost represent the cost of a bus leaving the deposit. It is possible to better control when buses leave/return to the deposit by defining different nodes $O^+_h$ and $O^-_h$ for different time slots, but these details cannot be discussed here. Clearly, the BS problem can be solved as a minimum cost network flow (MCF) on $G^{BS}$, but taken alone its optimal solution would be the all-0 flow since there is no constraint requiring trips to be covered.

All in all, the integrated model combines the BS graph and the TT graphs (one for each line) to yield the following MILP model:

$$\min \quad \alpha x + \sum_{l \in L} \beta_l y^l$$

$$\sum_{(j,i) \in A^{BS}} x_{ji} - \sum_{(i,j) \in A^{BS}} x_{ij} = 0 \quad i \in N^{BS}$$

$$\sum_{(j,i) \in A^{TT}_l} y^l_{ji} - \sum_{(i,j) \in A^{TT}_l} y^l_{ij} = b^l_i \quad l \in L, \ i \in N^{TT}_l$$

$$0 \leq x_{ij} \leq u_{ij} \quad (i,j) \in A^{BS}$$

$$y^l_{ij} \in \{0,1\} \quad l \in L, (i,j) \in A^{TT}_l$$

$$\sum_{l \in L} \sum_{(i,j) \in B(c)} y^l_{ij} = x_{c^-, c^+} \quad c \in N^{TT}_l$$

In the formulation, the deficits $b^l_i$ for TT are all zero except in a dummy source/sink node. The capacities $u_{ij}$ for BS are all one except that of the return arc $(O^-, O^+)$. For each $c \in C$, $l(c)$ is the line to which $c$ belongs, and the set $B(c)$ contains all arcs of $A^{TT}_{l(c)}$ entering the node representing $c$. It has to be remarked that choosing the coefficients $\alpha$ and $\beta$ of the objective function is nontrivial, as correctly doing so is crucial for obtaining the desired compromise between the two contrasting objective functions of the problem.

Our solution approach relies on the Lagrangian relaxation of the linking constraints (6): relaxing (6) we obtain one MCF and $|L|$ SP independent problems that can be efficiently solved. The Lagrangian dual is solved using a Bundle method. The algorithm uses the primal and dual information of the relaxations to guide a fixing heuristic that progressively selects trips and constructs integer solutions, i.e., the bus routes.
4 Computational Results

We tested our algorithm on 12 real-world instances provided by MAIOR, with number of lines ranging from 2 to 8, see Table 1. We consider two versions of our heuristic: “h-Bundle” solves the Lagrangian dual using the Bundle method, while “h-Clp” instead solves the continuous relaxation using the Clp, an open-source LP solver. We call “BSol” (i.e., best solver), the heuristic that produces the best solution amongst the two versions, and “BTS” (i.e., best time solver) the corresponding time (ranging from a few minutes to 6 hours). We compare the solution produced by best solver with solutions produced manually by TT and BS experts (“Manual”), as well as using the CPLEX MILP solver on the full formulation (1)-(6) with time limit corresponding to 1, 2 and 4 times BTS. The results show the percentage gain “X%” of the objective value of our solutions. On small instances h-Clp performs better, while on larger instances h-Bundle is often preferable. The obtained solutions are much better than the manually obtained ones, and the improved quality is perceived by experts of the field. For smaller instances, and allowing much longer times, CPLEX sometimes finds better solutions (“−X%” highlighted in bold), but in general the heuristic approach is competitive. Note that “X+” means that the cost of the solution is X times larger, and “nA” means that CPLEX could not find any feasible solution within the time limit. We can conclude that our method is competitive on real-world instances with respect to both manual solutions and a general MILP solver like CPLEX.

References


A VMND algorithm for the multi-period vehicle routing problem with due dates

Homero Larrain
Departament of Transport and Logistics Engineerings
Pontificia Universidad Católica de Chile, Santaigo, Chile
Email: homero@ing.puc.cl

Leandro C. Coelho
Canada Research Chair in Integrated Logistics and CIRRELT
Université Laval, Québec, Canada

Claudia Archetti
Department of Economics and Management
University of Brescia, Brescia, Italy

M. Grazia Speranza
Department of Economics and Management
University of Brescia, Brescia, Italy

1 Introduction

The Multi-period VRP with due dates, introduced in [1], is a routing problem where customers place an order that can be delivered between a given release date and a due date. For some customers, the due date may fall after the end of the planning horizon. These customers are deemed optional, and their orders can be postponed with an associated cost.

The solution of the MVRP gets exponentially harder as the instances grow in size. To be able to work with bigger instances, we implemented a Variable MIP Neighborhood Descent (VMND) algorithm, introduced in [3]. This hybrid algorithm works by enhancing an exact solution method by performing a local search every on every new best solution for the problem. To take advantage of the implementation of the exact solution of the problem, local searches are performed by solving constrained versions of the exact solution of the original problem.

2 Methodology

Three different formulations are introduced for the MVRPD in [1]. The authors found that their load-based formulation was the most efficient one to solve the problem. We base our VMND algorithm on a slightly improved version of this model. The general structure of our VMND is in Figure 1. The algorithm starts by solving an initialization stage where it tries to find a starting
solution, and then it keeps switching between two phases: an exact solution phase, and a local search phase.

In the exact solution phase, a branch-and-bound algorithm works on the solution of the main problem. Every time this phase begins, the algorithm resumes the solution where it left the previous time. This phase duration is capped by a maximum time parameter $\mu$, which is set when exiting the local search phase. If a new solution is found during the allotted time, the algorithm switches to the first neighborhood of the local search phase. Otherwise, it resumes the local search on a neighborhood that has not yet been explored for the current solution.

In the local search phase, a local search is performed on the current best solution, using one of the predefined neighborhoods. For this implementation we defined two different neighborhoods: “periods” and “$k$-vicinity”. These two neighborhoods are shown in Figure 2.

The “periods” neighborhood explores solutions that differ with a reference one in only one particular period. Figure 2 shows this neighborhood from a given current solution, parameterized for period $t_p = 2$. To solve this problem as a constrained MIP, all the arcs in periods $t \neq 2$ are set to their current value using bounds, leaving only variables in $t = 2$ free for the optimization. The “$k$-vicinity” neighborhood, for its part, defines a vicinity as a seed customer $i_p$ and its $k - 1$ nearest neighbors and optimizes the variables involving only these customers. Figure 2 illustrates the “$k$-vicinity” neighborhood from the current solution for a given customer $i_p$ and $k = 3$. On our implementation of the VMND for the MVRPD we used $k = 20$ across all instances.

Before entering the main phases of the VMND algorithm we call an initialization stage that attempts to find an easy feasible starting solution for the problem. This was implemented

---

**Figure 1:** Diagram of the VMND algorithm implemented for the MVRPD.

---

...
by solving two simple subproblems as constrained versions of the model for the main problem. Subproblem 1 forces each mandatory customer to be visited on their release date. In subproblem 2, mandatory customers are allowed to be visited either on their release date, or one period after. If both of these subproblems fail to provide an initial solution, the algorithm starts on the exact solution phase until it finds a feasible solution.

Figure 2: Example of a solution of the MVRPD and two of its neighborhoods.

3 Computational experiments

The algorithm was tested in a set of eighty instances, based on the benchmark instances from [2]. Our instances range from 25 to 100 customers and from 3 to 6 planning periods. Inventory costs assume two different values. For each combination of these parameters we randomly generated five different instances. All the instances were solved using B&B and VMND on machines equipped with Xeon processors running at 2.77GHz and 96GB of memory, limiting the running time to two hours. Average results are summarized in Table 1.

<table>
<thead>
<tr>
<th>Inv. Cost</th>
<th>N. of customers</th>
<th>N. of periods</th>
<th>B&amp;B Upper bound (GAP)</th>
<th>VMND Upper bound (GAP)</th>
<th>Cost reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>25</td>
<td>3</td>
<td>3641.3 (0.0%)</td>
<td>3641.3 (0.0%)</td>
<td>0.0%</td>
</tr>
<tr>
<td>High*</td>
<td>50</td>
<td>3</td>
<td>4874.9 (2.9%)</td>
<td>4841.6 (0.0%)</td>
<td>0.7%</td>
</tr>
<tr>
<td>High</td>
<td>75</td>
<td>3</td>
<td>11127.7 (11.6%)</td>
<td>10587.3 (3.3%)</td>
<td>4.9%</td>
</tr>
<tr>
<td>High</td>
<td>100</td>
<td>3</td>
<td>14845.8 (16.6%)</td>
<td>13689.8 (4.9%)</td>
<td>7.8%</td>
</tr>
<tr>
<td>High</td>
<td>25</td>
<td>6</td>
<td>5778.6 (0.0%)</td>
<td>5778.6 (0.0%)</td>
<td>0.0%</td>
</tr>
<tr>
<td>High</td>
<td>50</td>
<td>6</td>
<td>7919.5 (10.4%)</td>
<td>7906.5 (2.3%)</td>
<td>0.2%</td>
</tr>
<tr>
<td>High</td>
<td>75</td>
<td>6</td>
<td>8286.7 (21.4%)</td>
<td>8129.7 (5.5%)</td>
<td>1.9%</td>
</tr>
<tr>
<td>Low</td>
<td>25</td>
<td>3</td>
<td>2286.4 (0.0%)</td>
<td>2286.4 (0.0%)</td>
<td>0.0%</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>---</td>
<td>--------------</td>
<td>--------------</td>
<td>-----</td>
</tr>
<tr>
<td>Low</td>
<td>50</td>
<td>3</td>
<td>3230.2 (1.7%)</td>
<td>3223.0 (0.0%)</td>
<td>0.2%</td>
</tr>
<tr>
<td>Low</td>
<td>75</td>
<td>3</td>
<td>4469.3 (16.0%)</td>
<td>4269.3 (9.7%)</td>
<td>4.5%</td>
</tr>
<tr>
<td>Low</td>
<td>100</td>
<td>3</td>
<td>5972.3 (29.1%)</td>
<td>5269.7 (10.7%)</td>
<td>11.8%</td>
</tr>
<tr>
<td>Low</td>
<td>25</td>
<td>6</td>
<td>4051.6 (0.0%)</td>
<td>4051.6 (0.0%)</td>
<td>0.0%</td>
</tr>
<tr>
<td>Low</td>
<td>50</td>
<td>6</td>
<td>5637.3 (15.0%)</td>
<td>5523.4 (2.4%)</td>
<td>2.0%</td>
</tr>
<tr>
<td>Low</td>
<td>75</td>
<td>6</td>
<td>5877.1 (29.9%)</td>
<td>5385.1 (5.8%)</td>
<td>8.4%</td>
</tr>
<tr>
<td>Low*</td>
<td>100</td>
<td>6</td>
<td>8771.4 (62.3%)</td>
<td>6842.8 (11.3%)</td>
<td>22.0%</td>
</tr>
</tbody>
</table>

Table 1: Average results for VMND and B&B algorithms.

These results clearly show that our VMND greatly outperforms the original B&B implementation from [1], especially in bigger instances where it can reduce operational costs up to 22% in average. The two entries marked with an asterisk are cases where the B&B algorithm was not able to find a solution for one of the five instances. The average results shown were computed for the instances where both algorithms yielded a solution, even though VMND was able to find one in every instance we tested it. Further experiments, including a sensitivity analysis will be performed during the following months.

4 Results and conclusion

We have successfully implemented a VMND algorithm to solve the MVRPD, with very positive results, especially on harder instances where it reduces in 22% the best solution found by B&B, and it brings down the optimality gap from 62.3% to 11.3%. The VMND algorithm takes advantage of an exact solution of a problem, by using a replica of this implementation to solve local search problems, which makes it best suited for cases where an exact solution of a problem is available, because it can greatly improve results with a small additional effort.

The next steps on this research involve using VMND to perform different sensitivity analyses. In particular, we will study the solution of big instances of the problem, in order to gain knowledge on the effect of flexibility on the delivery dates.

References


1 Introduction

We study a complex variant of the vehicle routing problem (VRP) arising in the context of a third-party logistic provider, which provides long-haul transportation service for a number of partners. The company operates on a planning horizon and delivers products to various delivery locations as requested by the partners a few days in advance. A strict delivery deadline, in the form of a last possible delivery day, is set for each transportation request. The company has established agreements with each partner specifying, among others, a minimum level of on-time deliveries for its group of requests.

To capture some essential decisions in this situation, we introduce a variant of the VRP, called VRP with service levels (VRP-SL) and formulated as follows. Let $G = (V, E)$ be a complete undirected graph with $|V| = n + 1$ nodes. The node $v_0 \in V$ represents a depot, where a fleet of $m$ identical vehicles is based. Each other node $v_i$ for $i \in \{1, \ldots, n\}$ represents a customer, associated with a demand $q_i$, a profit $p_i$, and a service weight $s_i$ which represents its relative importance in the group service level constraint.

The set of customers is partitioned into $K$ subset. Each subset represents the deliveries of one partner and is associated with a requested service level $\alpha_k$. Any edge $(i, j) \in E$ represents a possible trip between a node $v_i \in V$ and a node $v_j \in V$ with a distance cost $d_{ij}$. The goal of the VRP-SL is to find up to $m$ vehicle routes starting and ending at the depot, such that: (i) each customer is serviced at most one time, (ii) the total demand
quantity of any route does not exceed a vehicle capacity \( Q \), (iii) the service level of each group \( k \) is attained, i.e., the total service weight of the deliveries to this group reaches \( \alpha_k \sum_{v_i \in V_k} s_i \), and (iv) the sum of travel costs and lost profits is minimized.

This problem belongs to the wide class of vehicle routing problems with profits, which also includes the team orienteering problem (TOP), the profitable VRP (VRPP), the VRP with private fleet and common carrier (VRPPFCC) and the capacitated profitable tour problem (CPTP). In addition, the VRP-SL is also a natural extension of the generalized VRP (GVRP). Interestingly, this problem fills a gap in the literature, since most known multi-vehicle problems with customer selections either aim to maximize service levels subject to distance constraints (TOP) or seek a weighted optimization of distance and service levels, through penalties for outsourcing or lost profits (VRPPFCC and CPTP). To this date, only a few studies have addressed multi-vehicle routing optimization subject to a service level (SL) constraint, usually in the context of hot strip mill scheduling for steel production [4, 5]. To address the problem, we introduce a compact formulation based on mixed integer linear programming (MILP) and a branch-and-price algorithm which can solve to optimality small- and medium-scale instances, as well as a hybrid population metaheuristic inspired by the Unified Hybrid Genetic Search (UHGS) framework of [3].

2 Exact methods

The proposed MILP is based on the two-commodity flow formulation of [1], and strengthened by three families of valid inequalities, which still keep the formulation compact. Moreover, we propose a column generation algorithm based on a set partitioning formulation, considering each possible route as a variable. The pricing sub-problem is an elementary shortest path problem with resource constraints, which is relaxed to consider the non-elementary \( ng \)-routes and solved by dynamic programming. The column generation is improved by means of heuristic pricing and dual stabilization, and embedded into a branch-and-bound procedure which branches on the original edge-flow variables.

3 Population-based metaheuristic

We also introduce a dedicated hybrid genetic search with advanced diversity control to solve efficiently medium and large scale instances. The method uses the same resolution strategy as the unified hybrid genetic search (UHGS) of [3]. However, the proposed algorithm also significantly differs from UHGS in the definition of its basic building blocks: solution representation, crossover, local search moves, distance measure between individuals, and penalties allowed.

In the proposed HGS, each individual is represented by two chromosomes: a service level chromosome, which gives the current service level of each group \( k \in \{1, \ldots, K\} \),
and the giant-tour chromosome, which provides a permutation of visits for the serviced customers, without occurrences of the depot. The cost of a solution involves the distance and the total profit associated to the customers which are visited, as well as penalty factors associated to violations of the capacity and service level constraints. The coefficients associated to these penalties are automatically adjusted by the method during the search.

To generate new individuals, we introduce an adapted order crossover (AOX). In a first step, the crossover inherits the service level information from both parents. This is done by crossing the service level chromosomes of both parents using an extended intermediate recombination [2]. Then, in a second step, the giant-tour chromosome of the child C is initialized with the longest size among both parent and inherits – as in the order crossover (OX) – a fragment of a parent. Finally, the giant-tour chromosome of C is completed by sweeping circularly the deliveries of the other parent and inheriting them, starting one index after the end of the fragment from \( P_1 \). Each insertion of a visit \( i \) of a group \( k \) is done under the condition that \( i \) does not already exist in \( C \), and that the target service level \( \alpha^T_k(C) \) has not yet been reached.

Each new individual is educated by a local search. We use the same classical neighborhoods as in [3]: 2-Opt, 2-Opt*, Swap, Relocate as well as generalized Swap and Relocate involving two consecutive nodes, restricted to close services. To also optimize the decision subset related to customer selections in the LS, we include three additional neighborhoods called Remove, Add and Replace.

4 Computational Experiments

The algorithms were coded in C/C++ and run on a single thread. The exact algorithms were run on a 3.07-GHz Intel Xeon CPU, and the metaheuristic was run on a 3.4-GHz Intel Core i7 CPU. CPLEX 12.7 was used for the resolution of the compact formulation and for the linear programs. We tested our methods on new instances of the VRP-SL with up to 199 customers, as well as classical VRPPFCC and CPTP instances.

For the VRP-SL, the B&P algorithm outperformed the compact formulation on most instances, except for some instances with few vehicles and long routes. HGS finds all of the known optimal solutions on all test runs. The method also returns solutions of consistent high quality: it found for 70% of the instances the value of the best known solution (BKS) on all runs of all instances. The percentage gaps between the average and best known solutions are close to zero (0.01% overall), and the instances with fewer groups appear to be generally easier to solve. The gaps to the lower bounds are also small (1.38% in average), and thus the solutions of HGS are guaranteed to be close to the optima. Finally, the average CPU time never exceeded 93 seconds.

For the VRPPFCC, the proposed metaheuristic appears to outperform previous meth-
ods in terms of solution quality, with an average gap of 0.141%, in comparison to 0.445% and 0.345% for the two best existing metaheuristics. During these tests, eight new BKS have been found. The proposed branch-and-price algorithm is the first in the literature to report optimal VRPPFCC solutions for three instances: CE-01, CE-06 and G-17.

For the CPTP, the proposed HGS obtains average solutions of similar or better quality (0.012% gap compared to 0.029% and 0.172% gap) in a fraction of the CPU time of previous algorithms. Three previous BKS were improved, leading to an average gap of −0.002% for the best solution quality of 10 runs. The proposed B&P found similar or better solutions for all instances tested. It improved the bounds for 37 instances, with an average improvement of 0.629%, and proved optimality for 103 instances, including ten new optimality certificates.

5 Acknowledgments

This research is partially supported by CNPq and CAPES in Brazil, and the National Foundation for Science and Technology Development (NAFOSTED), under Grant Number 102.99-2016.21 in Vietnam.

References


Adaptive large neighborhood search for multicommodity VRP

Wenjuan Gu, Diego Cattaruzza, Maxime Ogier, Frédéric Semet
Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL
F-59000 Lille, France
Email: maxime.ogier@centralelille.fr

1 Introduction

In this work we study a vehicle routing problem where customers request multiple commodities. Different strategies to deliver a set of commodities to customers were presented in [1]. Among these strategies, a new one, the commodity-constrained split-delivery mixed routing problem (C-SDVRP) is presented and compared with classical ways to deliver multiple products (allowing to split a commodity or using vehicles dedicated to each commodity). In the C-SDVRP, the vehicles are flexible and can deliver any set of commodities, and a customer who requests multiple commodities can be delivered by different vehicles. When a commodity is delivered to a customer, the entire required amount is handed over. This, if the customer is visited more than once, the different vehicles will deliver different sets of commodities.

This problem arises for example in delivery of fresh fruits and vegetables to catering. In this case, products can easily be mixed into the same vehicle, and splitting the delivery of an individual commodity is not acceptable. This is also very relevant in the multi-depot case, where each each depot has a limited quantity of each commodity. A customer can then be delivered by vehicles coming from different depots.

The C-SDVRP has first been studied in [1]. The authors proposed a branch-and-cut algorithm able to solve 25 out of 64 small instances (15 customers) to optimality within 30 minutes, and a heuristic method. The heuristic consists in (1) making copies of each customer, one for each commodity required by the customer, and (2) using a heuristic for the capacitated VRP. In [2], the authors proposed an extended formulation for the C-SDVRP and developed a branch-price-and-cut algorithm. They solved to optimality instances with up to 40 customers and 3 commodities per customer within 2 hours.

This work proposes an adaptive large neighborhood search to solve the C-SDVRP, with the objective of obtaining very good solutions on small instances, and to be able to efficiently solve large instances.
2 Problem definition

The C-SDVRP can be defined based on a directed graph $G = (V, A)$ in which $V = \{0\} \cup V_C$ is the set of vertices, and $A$ is the set of arcs. More precisely, $V_C = \{1, \ldots, N_C\}$ represents the set of customer vertices, and $0$ is the depot. A cost $c_{ij}$ is associated with each arc $(i, j) \in A$ and represents the non-negative cost of travelling from $i$ to $j$. Let $M$ be the set of commodities that have to be delivered to the customers. Any customer $i \in V_C$ may request any set of commodities. The depot contains a fleet of identical vehicles with capacity $Q$, able to deliver any subset of commodities. The objective is to minimize the total travelling cost.

The problem involves two related decisions: (1) finding a set of vehicle routes serving all customers; (2) selecting commodities delivered to each customer. The constraints are: (1) each route starts and ends at the depot; (2) the total quantity of commodities delivered by each vehicle does not exceed the vehicle capacity $Q$; (3) each commodity requested by each customer must be delivered by a single vehicle; (4) the demands of all customers need to be satisfied.

To solve the C-SDVRP, it is possible to duplicate the node associated with each customer by the number of commodities requested by the customer [1]. To each duplicated node, we then associate the demand of the customer for the corresponding commodity. For the sake of clarity, in the following we will call the duplicated nodes customer commodity.

3 Adaptive large neighborhood search

In order to solve the C-SDVRP for large instances, we propose a heuristic method based on the ALNS framework of [3]. Local search moves are also used in order to improve the solutions. A solution is represented as a set of routes. In order to take into account the specific features of C-SDVRP, a route can be represented: (1) as a sequence of customers, each customer having a set of commodities, or (2) as a sequence of customer commodities. In the first case, removing a customer from a route implies to remove the customer with the set of commodities delivered in this route, while in the second case it is possible to remove only one commodity.

3.1 General framework

ALNS relies on a set of removal and insertion heuristics which iteratively destroy and repair solutions. The probability to select a heuristic at a given iteration is influenced by its performance during past iterations. A sketch of the method is outlined in Algorithm 1.

An initial solution is constructed as follows: (1) give a random sequence of customers commodities to construct a giant tour, (2) apply a split procedure [4] to get a solution, (3) apply local search to improve this solution.
Algorithm 1 Adaptive large neighborhood search

1: generate an initial solution $s \in \{\text{solutions}\}$, $\rho \leftarrow 1$, $s_{\text{best}} \leftarrow s$

2: repeat

3: Roulette wheel: select a removal heuristic $h_{\text{rem}}$ and an insertion heuristic $h_{\text{ins}}$

4: Destroy: $s_{\text{rem}} \leftarrow$ remove $\rho$ customer commodities from $s$ applying $h_{\text{rem}}$

5: Repair: $s_{\text{ins}} \leftarrow$ insert removed customer commodities into $s_{\text{rem}}$ applying $h_{\text{ins}}$

6: Improve: $s' \leftarrow$ improve solution $s_{\text{ins}}$ with local search

7: if $f(s') < f(s_{\text{best}})$ then $s_{\text{best}} \leftarrow s'$ end if

8: if accept($s', s$) then $s \leftarrow s'$, $\rho \leftarrow 1$ else $\rho \leftarrow \rho + 1$, or $\rho_{\text{min}}$ if $\rho = \rho_{\text{max}}$ end if

9: until stopping criterion is met

10: return $s_{\text{best}}$

At each iteration, a simulated annealing criterion is used to determine if the new solution $s'$ is accepted. The number $\rho$ of customer commodities to remove from the current solution $s$ follows the scheme proposed in [5], with the aim of applying small moves when a new solution has just been accepted, while applying large moves when no new solutions has been accepted in the most recent iterations.

3.2 Removal and insertion heuristics

Classical removal and insertion heuristics ([3]) have been implemented. Removal heuristics are: random removal, worst removal and Shaw removal with a relatedness measure based on distance. Insertion heuristics are: greedy insertion and regret insertion. Worst removal and Shaw removal work with customers, while random removal can be applied either with customers or customer commodities. Insertion heuristics work with customer commodities.

3.3 Local search moves

In order to improve a solution, we apply a set of local search moves. Seven different moves are implemented. In the first three moves routes are considered as a set of customers. These moves are: (1) insert customer, (2) swap customers, (3) 2-opt of customers. Then, we propose two other classical moves adapted for customer commodities: (4) insert customer commodity, and (5) swap customer commodities. We also propose two other moves: (6) erase route: from a given route, remove the customer commodities until no other route has capacity to accept another customer commodity; and (7) reassign commodities: for a given customer we propose a Mixed Integer Program to optimally assign all the commodities of this customer to the routes of the solution.

Given a feasible solution, all 7 moves are iteratively applied one after the other, until no one improves the solution. Then, routes of the current solution are concatenated and split algorithm is applied. If this gives a better solution, the procedure is repeated.
4 Preliminary results

The algorithm is implemented in C++ and run on a Intel (R) Core(TM) i7-6500U, 2.50GHz and 8GB of RAM. The instances are the ones proposed in [1], based on the R101 and C101 Solomon instances for the VRP. We consider 3 sets of instances with up to 3 commodities: small instances with 15 customers, mid-sized instances with 20 customers or 80 customers.

Global results are reported in Table 1. We report the average, minimum and maximum gap (gap%) between the value we obtained and the best known solution in literature [2], the number of optimal values we obtained and the average computational time in seconds.

Table 1: Global results.

| $|\mathcal{N}_C|$ | nb instances | av.gap (%) | min gap(%) | max gap(%) | nb optimal | av.time(s) |
|---|---|---|---|---|---|---|
| 15 | 64 | 0.12 | 0.00 | 4.20 | 61 | 58 |
| 20 | 20 | 0.18 | 0.00 | 1.8 | 12 | 97 |
| 80 | 20 | 0.77 | -0.88 | 6.69 | - | 1767 |

From Table 1, we can see that our algorithm can solve to optimality 61 out of 64 small instances in reasonable computing time. On mid-sized instances, we obtain very good quality results on a reasonable amount of time, and we provide 10 new best known values for mi-sized instances with 80 customers.

The prospects are to implement specific removal and insertion heuristics. Moreover, the proposed method can then be extended to other versions of the problems, like multi-depot version with a limited capacity of each commodity at the each depot.

References


Vehicle Routing Problems
with Matches and Conflicts

Alessandro Hill
Business School
Universidad Adolfo Ibáñez, Santiago, Chile, Email: alessandro.hill@uai.cl

Stefan Voß
Institute of Information Systems
University of Hamburg, Hamburg, Germany, Email: stefan.voss@uni-hamburg.de

Roberto Baldacci
Department of Electrical, Electronic, and Information Engineering “Guglielmo Marconi”
University of Bologna, Cesena, Italy, Email: r.baldacci@unibo.it

1 Motivation and Contribution

We consider three practical extensions of the fleet size and mix vehicle routing problem (VRP)\cite{1} that integrate special route requirements for pairs of customers. First, we introduce the concept of customer *matches*. A match is a pair of customers that have to be contained in the same route. Secondly, we consider customer *conflicts*. A conflict is a pair of customers that have to be served by distinct routes. These two restrictions are somewhat complementary, since matches can be interpreted as positive conflicts. Matches find application in city sightseeing tours or, more generally, when planning round trips with route sharing preferences between travelers. Similarly, customer conflicts allow to incorporate incompatibilities that stem from language proficiency of tour guides and tourists. Finally, we also consider the problem that combines both customer match and conflict restrictions for planning optimal sightseeing tours.

In this work, we study the behavior of these rich VRP models under various match and conflict densities. We solve the resulting hard problems with constraint programming (CP) based methods on a test set derived from VRP literature instances. Besides a pure CP model, we use CP to refine solutions within a heuristic framework. Our results give indication for the capability of the CP approaches and the hardness of the problems under various parameter settings.

2 Optimization Models

In the considered classical fleet size and mix capacitated VRP, a set of customer nodes needs to be visited by a heterogeneous vehicle fleet. Each vehicle has a setup cost and is supposed to follow a tour that starts at the depot and ends at the depot, if used. The sum
of given customer demands cannot exceed the vehicle capacity. For each trip between two nodes, we are given a non-negative symmetric travel cost. The objective is to find a set of routes that minimizes the vehicle costs plus the overall travel costs. In our experiments we assume that the vehicle costs are much larger than the routing costs.

**Customer Matches** The VRP with matches (VRP+M) has not been considered in the literature before. These conjunctive side constraints are also called forcing constraints and have been considered for other combinatorial problems before (e.g., [6]). The match restrictions can be represented as a matching graph (see green edges in Fig. 1). Note that a match does not require its two customers to be visited in a specific order on a route, or by a specific vehicle. The match relation is transitive, meaning that if a customer is contained in two distinct matches then a match of the two counterpart customers is induced. We say the *match-closure* for a customer is the matching obtained by adding all its incident matches plus the transitive matches. The latter results in a matching graph that consists of disjoint cliques only. The VRP+M reduces to a multi-bin packing problem with the objective to minimize the global cost of the bins (i.e., vehicles) used when the arc costs are zero. More precisely, each customer corresponds to an item of size equal to its demand, and each vehicle corresponds to a bin of size equal to its capacity in this problem. All items have to be packed into the bins. Once one item is placed into a bin, the corresponding vehicle setup cost has to be paid. Two match items are merged such that they form a new item of aggregated size. We say that a VRP+M is *maximal* if no match can be added without causing infeasibility. A maximal VRP+M decomposes into separate TSPs induced by the cliques in the match-closure. The VRP+M is closely related to the clustered VRP [2] which requires the customers within pre-specified customer cluster sets to be visited by the same vehicle, but without visiting any intermediate non-cluster nodes on the corresponding route segment.

**Customer Conflicts** The VRP with conflicts (VRP+C) has been introduced in [2] and has been motivated by an application in waste collection, when specific material type combinations are not allowed to be in the truck simultaneously. Other combinatorial problems have been studied under similar disjunctive constraints (e.g., [4]). Besides generalizing the classical VRP, the graph coloring problem is a special case when having uniform arc costs and unlimited vehicle capacity. The VRP+C is a special case of a more general class of VRPs with loading constraints in which items, corresponding to conflict customers, cannot fit in a vehicle simultaneously. A VRP+C with complete conflicts, i.e., every pair of customers bears a conflict, is trivial since one vehicle has to be assigned exclusively to each single customer. Customer conflicts can be represented as a conflict graph (see orange edges in Fig. 2). In contrast to matches, the conflict relation is not transitive.

**Customer Matches and Conflicts** In case that we have to deal with matches and conflicts simultaneously (VRP+M+C), we observe that a conflict that affects a customer in a match implies a conflict with the other match customer (and therefore with all customers in the match closure of the former match customer). We assume that the set of matches does not intersect the conflict set to assure feasibility.
3 Solution Methodology

We use a scheduling-based CP model in which travel distances are given as discrete interval variables. A \textit{no\_overlap} constraint is imposed on each sequence variable that corresponds to the potential tours of each vehicle. Also, the \textit{alternative} constraint is used to force unique vehicle choice to visit each customer. In addition, we implement a CP based refinement heuristic to improve the solutions found by CP. Namely, we solve the TSP induced by each route in the incumbent solution and replace routes if they could be improved. The model is implemented using ILOG CP Optimizer 12.71 as CP solver.

4 Preliminary Results

In our experiments, we use a test set which we adapted from the one introduced in [3]. The arc costs are Euclidean and they contain 50 to 200 customers. Match and conflict pairs are randomly drawn from the set of all possible customer pairs. The former is done using a uniform distribution, whereas we prefer customer pairs with low arc connection cost for the conflicts in order to provoke non-cost-efficient routes. Using relative densities of 0, 0.5, 1, 2.5, 5, 10, 15, 20, 25, 30, 40, ..., and 100%, we obtain 238 instances for each model. The percentage of matches and conflicts is with respect to the number of customers and the number of potential pairs, respectively. Figure 1 illustrates solutions computed for a small test instance using an increasing number of matches. Similarly, the restricting effect of conflicts can bee seen in Figure 2.

![Figure 1: VRP+M solutions for increasing match density (Costs: 199, 210, 230, 233).](image)

![Figure 2: VRP+C solutions for increasing conflict density (Costs/vehicles: 199/3, 223/3, 235/3, 317/8).](image)

We limited the computation time to 600 seconds for CP and to 20 seconds for each refinement, using an Intel i7-7600U@2.8GHz (4 cores) machine. CP is not able to prove optimality for the instances. In Figure 3 (left), the average relative improvement is depicted for instances of augmenting size. We observe an improvement up to 16%.
for larger instances the refinement heuristic is useful. The chart (right) shows the relative rise of the routing costs for increasing constraint density. It can be seen that conflicts have a much higher impact on the latter than matches for our instances.

![Figure 3: Cost improvement after refinement (left), and routing cost increase (right).](image)

5 Outlook

In order to better understand the hardness of the problems with respect to match and conflict density, we are currently developing mathematical programming approaches in order to compute lower bounds on the achievable costs and compare their performance to CP regarding primal solutions. We are also working on further refinement strategies since they have been proven useful in our preliminary tests.

References

Matheuristic for the Consistent Inventory Routing Problem with Time Windows

Emilio Jose Alarcon Ortega, Michael Schilde, Karl F. Doerner
Department of Business Administration, University of Vienna, Vienna, Austria
Email: emilio.jose.alarcon.ortega@univie.ac.at

1 Introduction

In this abstract we introduce a class of problems which integrate vehicle routing problems with time windows (VRPTW) with inventory management as well as a consistency. This variant of the problem could be seen in some real-world applications like the delivery of beer or tobacco where customers prefer deliveries to be at the same time of the day. We consider that a customer has consistent deliveries if the arrival times of all deliveries are the same or similar. Furthermore, as another characteristic of the problem, and due to temporary high demands that can exceed the capacity of a single vehicle, we allow split deliveries in order to satisfy those customers’ demands. It is remarkable that, in the literature, there are still few solution methods that are able to solve Inventory Routing Problems (IRPs) with time windows or other problem characteristics that are often found in real world applications. We denote our problem as Consistent Inventory Routing Problem with Time-Windows and Split Deliveries (CIRPTWSD).

Previous work about the IRP and variants was summarized in [1]. However, there are only a few articles in the literature related to the IRP with consistency and time windows within the time periods. A similar problem was presented in [2] where the authors assumed that the inventory levels are monitored by the company. Azi et al. [3] present algorithms for the vehicle routing problem with time windows which are treated as hard constraints. In [4], the authors dealt with the arrival-time consistency aspect of the problem by proposing template-based routes. Consistency and its variants were also reported in [5].

2 Problem description

The CIRPTWSP can be formulated as a mixed integer program. Given a directed graph \( G = (A, V) \), where \( V = 0, ..., n \) is the set of nodes with 0 being the depot and \( A \) is the set of arcs given by all pairs of nodes \( i \) and \( j \). The planning horizon covers \( P \) periods and,
at the same time, these periods are divided into $R$ subperiods. Subperiods are introduced to state a maximum driving time and also to deal with the different working shifts that beer companies have. Due to the complexity of this problem we assume that demands are known for each period and subperiod and consumption rates are continuous.

The decision variables in our model are: the binary sequence indicators $x_{ij}^{kpr}$ from customer $i$ to customer $j$ for each vehicle $k$, period $p$, and subperiod $r$, the binary visit indicators for customer $i$, $y_{i}^{kpr}$, the amounts delivered $q_{i}^{kpr}$, the arrival times $t_{i}^{kpr}$, the final inventories at the end of each subperiod $I_{i}^{pr}$, the earliest and latest arrival times at each customer $t_{i}^{max}$ and $t_{i}^{min}$, and the amounts of demand lost at every customer due to stock-out situations $o_{i}^{pr}$.

The objective of the model is to minimize the costs related to the routing, inventory holding, and consistency decisions over the planning horizon as well as to minimize the cost of possible stock-outs.

Apart from the different costs in this problem, we have different groups of constraints. Some of these constraints are well known and widely commented in the literature. Routing, time, and inventory balance flow constrains are considered in the mathematical model as well as constraints related to the use of a maximum number of capacitated vehicles. Furthermore, apart from these constraints we include some others to calculate shortfall quantities and to forbid overstock situations at the retailers by taking into account the possibility of split deliveries and the time continuous consumption of commodity at every customer location. Finally, we consider a group of constraints used to satisfy the order up to level policy while taking into account the split delivery characteristic of the problem. This way we ensure that at the moment of the last delivery to a customer in a subperiod, the amount delivered by the vehicle satisfies the order up to level policy.

### 3 Solution approach

To solve the CIRPTWSD, we develop a matheuristic solution approach based on the concept of Adaptive Large Neighborhood Search (ALNS). The initial solution is generated using an adaptation of the cheapest insertion heuristic combined with a local search. After applying the ALNS we solve a reduced problem based on the problem formulation. Here we repair the obtained solution by improving possible inconsistent deliveries, stock-outs and excessive inventory holding costs.

To construct an initial solution for each period and subperiod we create a list of customers that must be served in each subperiod. This is the case if they run out of stock in the current subperiod or they run out of stock before the end of the next possible delivery time window. We insert those customers while possible by applying a cheapest insertion algorithm using the order up to level distribution policy. Then, if there are still
customers that need a delivery but can not satisfy the order up to level policy, we insert them with a lower amount. If even this is not possible, we split the remaining customers’ deliveries to two vehicles. After finishing this procedure, we apply an improve operator to the obtained solution to destroy single-customer routes.

After we obtain an initial solution, we apply the ALNS procedure in order to improve the quality of the obtained solution. In the proposed ALNS we use operators to destroy and repair the current solution and a procedure to update the inventory after applying an operator. We propose six destroy operators and three repair operators that are selected using a roulette-wheel selection operator. The ”Remove worst” operator deletes the $p$ worst customers (with respect to the detour a customer causes between its preceding and succeeding customer visit along the route). The ”Remove random” operator: deletes $p$ random customers. The ”Remove vehicle” operator: removes all customers in all routes of a randomly selected vehicle. The ”Remove subperiod” operator removes all routes in a randomly selected subperiod. The ”Remove least consistent” operator deletes $p$ customers whose arrival times are very inconsistent. The ”Remove customer” operator removes all deliveries done to $p$ random customers. After we apply a destroy operator, we update the inventory levels of the removed customers and we create a list of customers that must be repaired because of stock-out situations. We then apply a randomly selected repair operator. The ”Repair best before stock-out day” operator creates a list of possible insertions in the day a stock-out occurs and the preceding days for customers that must be repaired. The possible insertions are sorted by the total distance of the detour this insertion causes. It then randomly selects one of the three best possible insertion positions and inserts the customer into the route. If, after evaluating the new inventory flow of the customer, there still are stock-out situations in the succeeding days, we repeat the process between the new stock-out day and the last insertion day. The ”Repair random before stock-out day” operator creates a list of possible insertions as in the previous operator and then randomly selects one of them from the entire list. We repeat the process until a feasible solution is reached. The ”Repair consistency” operator inserts customer such that the difference between earliest and the latest arrival times to this customer is minimized.

If the new solution is not more than 50% worse than the best known solution we solve a reduced variant of the mathematical problem formulation. The reduced variant is based on the mathematical formulation where visits to customers and route sequences are fix and, therefore, it does not include any binary or integer variable and solving times are low. This aims to minimize consistency costs by introducing waiting times, as well as to improve the performance of the algorithm with respect to the amounts delivered and final inventory levels. If the new solution improves the current best solution, we update the best known solution as well as the weights of the destroy and repair operators that have been used in this iteration. The stopping criterion for the ALNS is an overall time limit.


<table>
<thead>
<tr>
<th>Customers</th>
<th>CPLEX</th>
<th>Algorithm</th>
<th>Max Gap</th>
<th>Average Gap</th>
<th>Min Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>5*</td>
<td>1023.23</td>
<td>1030.32</td>
<td>4%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>10*</td>
<td>1371.44</td>
<td>1399.34</td>
<td>8%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td>15*</td>
<td>1774.83</td>
<td>1861.09</td>
<td>14%</td>
<td>5%</td>
<td>0%</td>
</tr>
<tr>
<td>20**</td>
<td>2363.69</td>
<td>2737.30</td>
<td>31%</td>
<td>16%</td>
<td>1%</td>
</tr>
<tr>
<td>48</td>
<td>5746.20</td>
<td>5183.68</td>
<td>10%</td>
<td>-9%</td>
<td>-24%</td>
</tr>
</tbody>
</table>

Table 1: Results for instances with 4 periods and 1 subperiod

4 Computational Experiments

Computational tests using the described algorithm implemented in C++ have been done using a benchmark set for the periodic vehicle routing problem with time windows. These instances were adapted to include the inventory information needed for our problem. In Table 1 we summarize the results obtained by solving 60 instances of different sizes. We run the algorithm 10 times for each instance with a maximum computing time of 30 minutes. Results are compared to the best solution found using CPLEX with a maximum computing time of 10 hours. In instances marked with * we obtain the optimal solution using CPLEX while ** means that CPLEX is not able to obtain a feasible solution in 2 instances of that size. Some additional experiments with 96 customers have been done. For these instances CPLEX can not find any feasible solution within the time limit.

References


A Variable MIP Neighborhood Search for the Multi-attribute Inventory Routing Problem

Leandro C. Coelho
CIRRELT and Canada Research Chair in Integrated Logistics, Laval University, Quebec City, Canada,

Demetrio Laganà
Department of Mechanical, Energy and Management Engineering University of Calabria, Arcavacata di Rende, Italy

Ammarita De Maio
Department of Mechanical, Energy and Management Engineering, University of Calabria, Arcavacata di Rende, Italy
Email: annarita.demaio@unical.it

1 Introduction

In this work a rich variant of the Inventory Routing Problem (IRP) is introduced, the so-called Multi-attribute Inventory Routing Problem (MAIRP). The classical IRP combines inventory management policies and vehicle routing operational plans within a unique and coordinated scheme. The actual real applications present the necessity to consider IRP variants with additional features and constraints with the aim to model a higher level of system detail, including but not limited to: variants in the system structure (several depots, different commodities and vehicles fleet), vehicle requirements (route or time restriction), customer requirement (time windows). In our work we introduce the case of a MAIRP, in which a vendor has to deliver a group of products from different origins or depots to a set of customers with a heterogeneous fleet of vehicles, while limiting the duration of the routes; the aim is to minimize the total inventory costs at the depots and customers and the transportation cost, while satisfying the customers’ demands and avoiding stock-out. The duration of a route represents a limitation on the travel time of the drivers during the working day. In fact, since we make the hypothesis that a truck can be led by only one driver, it is necessary to consider a limited time during which the truck can make deliveries. Since the MAIRP is a special case of the IRP, it is an NP-hard problem. An extensive overview of the IRP and its industrial applications can be found in [1] and [2]. We develop a hybrid algorithm based on the Variable Neighborhood Search
method, in which the neighborhoods’ exploration is performed through a branch–and–cut algorithm. The expected impact is to accelerate the resolution process with respect to other approaches using only the MIP. The methodology was successfully applied to the inventory and distribution of cash for automated teller machines problem by [3]. The idea to solve the local search using a MIP is not new, it was well explored in the literature (see [4], [5] and [6]). In the following sections the problem description, the algorithm and the conclusion are presented.

2 Problem description and formulation

In this section, the MAIRP is described and the corresponding mathematical formulation based on binary edge-variables is presented. We consider a complete undirected graph $G = (V, E)$. We introduce the following notation: let $D = \{1, \ldots, m\}$ be the set of depots and $I = \{m+1, m+2, \ldots, n\}$ be the set of customers, with $V = D \cup I$; let $P = \{1, \ldots, p\}$ be the set of different products to be delivered, $T = \{1, \ldots, H\}$ be the set of the discrete time periods and $K = \{1, \ldots, k\}$ the set of vehicles, with different capacities $Q_k$ and with a variable cost per distance unit $\alpha_k$. For each customer we define a starting inventory level $I_{ip0}$, a maximum inventory level $U_i$, a demand $d_{ipt}$ and an inventory holding cost $h_{ip}$. For each depot we define a starting inventory level $I_{id0}$, an availability of freight $r_{dpt}$ and an inventory holding cost $h_{id}$. We assume no storage capacity at the depots. A non-negative cost $c_{ij}$ that satisfies the triangular inequality and a travelling time $\tau_{ij}$ are associated with edge $(i, j) \in E$. The service time is denoted by $\omega$, while the maximum duration time of each route is $M$. Given $S \subseteq I$ (a proper and non–empty subset of vertices), $E(S)$ denotes the set of edges $(i, j)$ such that $i, j \in S$. Variables $I_{ipt}$ and $I_{dpt}$ denote the inventory level at the end of period $t \in T$ for the product $p \in P$ for each customer $i$ and each depot $d$, respectively. Variable $y_{ipktd}$ indicates the quantity to deliver to customer $i \in I$ of product $p \in P$ in period $t \in T$ by vehicle $k \in K$ starting the tour from depot $d \in D$. Binary variable $x_{ijktd}$ is equal to 1 if vehicle $k \in K$ travels directly from vertex $i \in V$ to vertex $j \in V$ in period $t \in T$ starting the tour from depot $d \in D$; binary variable $z_{iktd}$ is equal to 1 if vehicle $k \in K$ visits customer $i \in I$ in period $t \in T$ starting the tour from depot $d \in D$. Finally, binary variable $z_{dktd}$ is equal to 1 if vehicle $k \in K$, located in depot $d \in D$, is used in period $t \in T$. The mathematical formulation is described by (1)–(19).

\[
\begin{align*}
\min & \sum_{t \in T} \sum_{p \in P} \sum_{i \in I} k_{ip} I_{ipt} + \sum_{t \in T} \sum_{p \in P} b_{dp} I_{dpt} + \sum_{(i,j) \in E} \sum_{k \in K} \sum_{d \in D} \alpha_k c_{ij} x_{ijktd} \\
\text{s.t.} & \quad I_{dpt,t+1} = I_{dpt} - \sum_{k \in K} \sum_{i \in I} y_{ipktd} + r_{dpt} \quad \forall t \in T, \forall p \in P, \forall d \in D \\
& \quad I_{ipt,t+1} = I_{ipt} + \sum_{k \in K} \sum_{d \in D} y_{ipktd} - d_{ipt} \quad \forall t \in T, \forall p \in P, \forall i \in I \\
& \quad \sum_{p \in P} \sum_{k \in K} \sum_{d \in D} y_{ipktd} + \sum_{p \in P} I_{ipt} \leq U_i \quad \forall t \in T, \forall i \in I
\end{align*}
\]
\[
\sum_{i \in I} \sum_{p \in P} y_{ipktd} \leq Q_k z_{dktd} \quad \forall t \in T, \forall d \in D, \forall k \in K
\] (5)

\[
y_{ipktd} \leq Q_k z_{iktd} \quad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K, \forall d \in D
\] (6)

\[
\sum_{i \in I} \sum_{p \in P} y_{ipktd} \geq z_{dktd} \quad \forall t \in T, \forall d \in D, \forall k \in K
\] (7)

\[
z_{ktd} = 0 \quad \forall t \in T, \forall d \in D, \forall k \in K, \forall b \in D (p \neq b)
\] (8)

\[
\sum_{d \in D} z_{dktd} \leq 1 \quad \forall t \in T, \forall k \in K, \forall i \in I
\] (9)

\[
\sum_{d \in D} z_{iktd} \leq 1 \quad \forall t \in T, \forall d \in D
\] (10)

\[
\sum_{k \in K} z_{dktd} \leq 1 \quad \forall t \in T, \forall d \in D
\] (11)

\[
\sum_{(i,j) \in E} \tau_{ij} x_{ijktd} + \sum_{i \in I} \omega_i z_{iktd} \leq M 
\quad \forall k \in K, \forall t \in T, \forall d \in D
\] (12)

\[
\sum_{j \in I, j < i} x_{ijktd} + \sum_{j \in I, j > i} x_{ijktd} = 2z_{iktd} \quad \forall i \in V, \forall t \in T, \forall d \in D, \forall k \in K
\] (13)

\[
\sum_{(i,j) \in E(S)} z_{ijktd} - z_{iktd} \leq 2z_{iktd} - z_{uktd} \quad \forall S \subseteq V, |S| \geq 2, \forall t \in T, \forall k \in K, \forall d \in D, \forall u \in S
\] (14)

\[
y_{ipktd} \geq 0 \quad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K
\] (15)

\[
l_{ipt} \geq 0 \quad \forall i \in I, \forall t \in T, \forall p \in P
\] (16)

\[
l_{dpt} \geq 0 \quad \forall d \in D, \forall t \in T, \forall p \in P
\] (17)

\[
y_{ipktd} \geq 0 \quad \forall i \in I, \forall t \in T, \forall p \in P, \forall k \in K, \forall d \in D
\] (18)

\[
z_{iktd} \cdot z_{dktd} \in \{0, 1\} \quad \forall i \in I, \forall t \in T, \forall d \in D, \forall k \in K
\] (19)

The objective function (1) minimizes the total inventory and routing costs. Constraints (2)-(4) define the inventory balancing at the depots and customers. Constraints (5)-(6) are vehicle capacity constraints. Constraints (7)-(11) characterize the multi-depot and multi-product case and model split delivery conditions. Constraints (12) impose that maximum duration of the route. Constraints (13) and (14) control the degree of the vertices and prohibit subtours. Constraints (15)–(19) define the integrality and non-negative conditions for the variables.

### 3 The Variable MIP Neighborhood Search

The VMNS is a matheuristic algorithm that uses a MIP to explore neighborhoods. In our algorithm, the MIPs are based on the mathematical formulation (1)-(19), in which some variables are fixed and new constraints are added. The algorithm is based on two different phases called branch-and-cut routine (BCR) and local search routine (LSR), that are executed alternately. We introduce the following notation: \( Y \) is the solution vector for the MIP problem, \( N = \{1, \ldots, h\} \) is the set of the neighborhoods, that are obtained by applying an operator to the current solution \( Y \): applying an operator consists of setting a subset of variables to the value they take in the current solution and leaving the other free. In such a way, a restricted MIP corresponding the neighbourhood is solved. The routines are described in the following:
BCR: in this phase the MAIRP is solved. At the beginning, a feasible solution is obtained through a relaxation of the main MIP and it is passed as initial solution to a basic branch-and-cut algorithm that runs until an improvement is found or a time limit is reached. At this point the LSR is runned. The LSR returns a new initial solution for the branch-and-cut that is the best improvement found in the local search phase.

LSR: in this phase different neighborhoods are explored around a given solution $Y$ (the best solution found in the BCR, when $n = 1$). If a new improvement is found in the current neighborhood $n$, the procedure is re-started from the beginning and the neighborhoods of the new solution are explored from $n = 1$. If no improvement occurs, the exploration is performed according to the sequence of the neighborhoods. The procedure ends if the time limit is reached or if all the neighborhoods are explored without improvements.

We propose five different neighborhoods for the VMNS, that alternatively sets the variables related to a given feature of the problem. They are divided in two main categories: simple neighborhoods, that consider the variables related to only one feature (depot, period, vehicle), and complex neighborhoods, that set the variables related to different features (depot-period-vehicle, depot-customer).

4 Conclusion

The problem is very challenging from the computational point of view and the complexity depends much from the neighborhoods size, as well as it is very interesting for real-case applications. If the size of neighborhoods is very small it is possible to have no improvement of the objective function, while if it is large a long time is required to find a good feasible solution that improves the incumbent. The aim is to find a good trade-off, while some steps for improving the approach are planned: a sensitive analysis on the effectiveness of the neighborhoods can be performed and a diversification procedure is needed.

References


A Benders Decomposition for the Inventory Routing Problem

Fabián Penaranda*, Marcus Poggi*, Rafael Martinelli**
*Departamento de Informática, **Departamento de Engenharia Industrial
Pontifícia Universidade Católica do Rio de Janeiro, Brazil
Email: {fpenaranda, poggi}@inf.puc-rio.br, martinelli@puc-rio.br

1 Introduction

Critical goods inventory is, most often, managed by the supplier under a service level agreement with its clients. This is the case for many chemicals industries, hospitals and other activities where the consequences of having a stock falling short is very expensive, if not unmeasurable. Under these conditions and when customers have with suppliers a joint managing of its stocks, the total cost of the logistic activity of maintaining a good available at all customers can be managed by a centralized entity. This entity, most likely the supplier, will then be responsible for determining when and how to deliver this good in such away that it fulfills the customer demands at all times while keeping the stocks within the desired ranges, at all customers and at its own warehouse. The inventory-routing problem (IRP) captures then the planning decisions when this entity also controls the routing of the delivery. The aim is therefore the minimimization of total operating cost, inventory and routing costs. The IRP is sensitive to the stock replenishment policy. Two policies are tackled in the literature. The OU policy where deliveries must always fill the customer demands at all times while keeping the stocks within the desired ranges, at all customers and at its own warehouse. The inventory-routing problem (IRP) captures then the planning decisions when this entity also controls the routing of the delivery. The aim is therefore the minimization of total operating cost, inventory and routing costs. The IRP is sensitive to the stock replenishment policy. Two policies are tackled in the literature. The OU policy where deliveries must always fill the inventory to its maximum capacity; and the ML policy where any quantity can be delivered. We consider the ML policy in this work, for which, currently, the best results are obtained by [3], improving the results in [2].

In the IRP, a single supplier, denoted 0, produces a known quantity of a single commodity at each period \( t \) of a finite planning horizon \( H \). Using a homogeneous fleet of vehicles \( K \), each with capacity \( C_k \), the supplier serves a set \( N \) of customers where each customer \( i \) demands a known quantity \( q_{it} \) in each period \( t \in H \). Given \( V = N \cup \{0\} \), each element \( i \in V \) has an inventory capacity \( C_i \), an initial inventory \( I_i^0 \leq C_i \), and a unit holding cost \( h_i \). The IRP consists of building feasible vehicle routes in each period such that no stockout occurs, while respecting inventory capacities and the load satisfies vehicle capacities. Furthermore, each customer can be serviced at most once per period.
2 A Benders Decomposition

We propose a Benders Decomposition for the IRP based on the MIP formulation of [2] on which we keep the inventory management in the first level, moving the vehicle routing decisions to the second level, solved for each period separately. The formulations on both levels of the decomposition are integer programs. With this structure we are not limited to use the resulting routing formulation from the decomposition, any known vehicle routing formulation can be used. Clearly, the tighter the formulation, the stronger are the bounds we expect to obtain. We choose to use the Two-Index Formulation [5]. Regular Benders cuts obtained from solving the linear relaxation of the second level may be separated during the execution of the algorithm. However to reach the integer optimal solution of the first level, we must separate Benders cuts from an integer solution of the second level. This is done by tailored Benders combinatorial cuts (or integrality cuts as in [4]).

Benders Master Problem. The first level of the decomposition contains the inventory management decisions. The routing contribution to the objective function is represented by variable $\alpha$ as in (1), and its value is coupled to variables $\alpha_t$ in (3) which in turn are associated to benders cut added for each period $t \in H$.

$$\min \sum_{i \in V} \sum_{t \in H} h_i.I_i^t + \alpha$$

(inventory constraints)

$$\alpha - \sum_{t \in H} \alpha_t \geq 0$$

Benders Optimality Cuts. Benders optimality cuts (4) are separated by solving the linear relaxation of the second level VRP associated to each master demand solution. Integer variables $w_s^t$ are associated to the right-hand side of the rounded capacity cuts in the sub-problem and constraints (5) state that their minimum value in the master problem corresponds to the number of vehicles needed to serve demands of all customers in $S \subset V$. Dual values $\beta^t$ and $\pi_i^t$ are associated to the degree constraints related to the depot and customers respectively, and dual values $\theta_s^t$ to the capacity constraints.

$$\alpha_t \geq \sum_{i \in V} 2.\pi_i^t.y_i^t + \sum_{s \in S} 2.\theta_s^t.w_s^t + 2.K.\beta^t$$

$\forall t \in H$ (4)

$$C_k.w_s^t \geq \sum_{i \in s} Q_i^t$$

$\forall t \in H, \forall S \subset V$ (5)

$$w_s^t \in \mathbb{Z}^+$$

$\forall t \in H, \forall S \subset V$ (6)

Benders Feasibility Cuts. Benders feasibility cuts (7) are separated by solving the linear relaxation of a Bin Packing sub-problem, which is used only to check if a demand
obtained from the master problem would have a feasible routing solution, i.e. if it fits the vehicles (bin) capacities without splitting.

\[
\sum_{i \in V} \pi_t^i Q_t^i + \sum_{i \in V} \phi_t^i y_t^i + \sum_{k \in K} \beta_t^k \leq 0 \quad \forall t \in H
\]  

(7)

**Combinatorial Benders Optimality Cuts.** Combinatorial Benders optimality cuts (8)-(10) are used to close the integrality gap in the master problem and are separated by solving the integer VRP subproblem. We call them route based cuts, because the exact routing cost for a given demand \( r \) imposes a valid bound on the routing cost of any other demand \( r' \) if all feasible routes of \( r' \) are a subset of the feasible routes of \( r \). The set of infeasible sets \( IS_r \) contains minimal customer subsets whose demands do not fit a single vehicle. Each of these subsets, along with zero demand customers, represent the unfeasible routes associated with demand \( r \). Binary variables \( \alpha_{t-IS_r} \) have an associated cost of \( Z_{VRP_r}^* \) that corresponds to the optimal routing cost of \( r \). Constraints (9) forces the value of these variables to 1 for each demand \( r' \) that meets the condition above, in which case, the \( Z_{VRP_r}^* \) routing bound is imposed to \( r' \) by constraints (8). Binary variables \( m_{s-t}^{r,t} \) are set to 1 by constraints (10) if each infeasible customer subset \( s \subset IS_r \) is still infeasible for demand \( r' \).

\[
\alpha_t \geq Z_{VRP(r)}^* \alpha_{t-IS_r} \quad \forall t \in H
\]  

(8)

\[
\alpha_{t-IS_r} \geq 1 - |IS_r| - n + \sum_{s \in IS_r} m_{s-t}^{r,t} + \sum_{i \in V_1} y_t^i + \sum_{i \in V_2} (1 - y_t^i) \quad \forall t \in H
\]  

(9)

\[
\sum_{i \in s} Q_t^i \leq K.C_k.m_{s-t}^{r,t} + C_k \quad \forall t \in H, \forall s \in IS
\]  

(10)

3 Computational Experiments

We tested the methodology on some of the classical instances from the IRP literature [1], considering multiple vehicles in the same manner as [3]. Our approach was developed in Julia and the experiments were conducted on an Intel Core i7 6700HQ, with 32GB of RAM. For each test, we allowed a time limit of 3600 seconds and we report the results obtained for three vehicles in Table 1. Column **Instance** presents the name of the instance, Column **Iter** presents the executed iterations, Columns **LB** and **UB** present the lower and upper bounds, respectively, Column **T(s)** the time in seconds and Columns **RBC** and **CBC** the number of regular and combinatorial Benders Cuts, respectively.

Experiments show optimal solution obtained only for instances up to 10 customers for 3 periods and up to 5 for 6 periods. The bottleneck of the current approach appears to be the resolution of the master problem. The integer extra variables to deal with the rounding of the Benders optimality cuts seem turn the MIP problem hard. Handling a
<table>
<thead>
<tr>
<th>Instance</th>
<th>Iter</th>
<th>LB</th>
<th>UB</th>
<th>RBC</th>
<th>CBC</th>
<th>T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>absH3high_1n10</td>
<td>6</td>
<td>5506.09</td>
<td>5506.09</td>
<td>27</td>
<td>27</td>
<td>20.21</td>
</tr>
<tr>
<td>absH3high_2n10</td>
<td>15</td>
<td>5743.25</td>
<td>5743.25</td>
<td>66</td>
<td>66</td>
<td>62.87</td>
</tr>
<tr>
<td>absH3high_3n10</td>
<td>29</td>
<td>4808.06</td>
<td>4808.06</td>
<td>123</td>
<td>123</td>
<td>103.27</td>
</tr>
<tr>
<td>absH3high_4n10</td>
<td>57</td>
<td>5335.29</td>
<td>5335.29</td>
<td>72</td>
<td>195</td>
<td>1373.1</td>
</tr>
<tr>
<td>absH3high_5n10</td>
<td>19</td>
<td>5224.45</td>
<td>5224.45</td>
<td>93</td>
<td>93</td>
<td>43.91</td>
</tr>
<tr>
<td>absH3low_1n10</td>
<td>7</td>
<td>2732.61</td>
<td>2732.61</td>
<td>33</td>
<td>33</td>
<td>19.37</td>
</tr>
<tr>
<td>absH3low_2n10</td>
<td>24</td>
<td>3470.15</td>
<td>3470.15</td>
<td>114</td>
<td>114</td>
<td>58.79</td>
</tr>
<tr>
<td>absH3low_3n10</td>
<td>37</td>
<td>2649.26</td>
<td>2649.26</td>
<td>171</td>
<td>171</td>
<td>94.68</td>
</tr>
<tr>
<td>absH3low_4n10</td>
<td>32</td>
<td>3183.36</td>
<td>3183.36</td>
<td>93</td>
<td>135</td>
<td>174.95</td>
</tr>
<tr>
<td>absH3low_5n10</td>
<td>28</td>
<td>2461.12</td>
<td>2461.11</td>
<td>129</td>
<td>129</td>
<td>75.27</td>
</tr>
<tr>
<td>absH6high_1n5</td>
<td>29</td>
<td>7259.05</td>
<td>7259.05</td>
<td>294</td>
<td>294</td>
<td>479.79</td>
</tr>
<tr>
<td>absH6high_2n5</td>
<td>26</td>
<td>6530.35</td>
<td>6530.35</td>
<td>306</td>
<td>306</td>
<td>841.55</td>
</tr>
<tr>
<td>absH6high_3n5</td>
<td>5</td>
<td>9862.89</td>
<td>9862.89</td>
<td>72</td>
<td>72</td>
<td>56.89</td>
</tr>
<tr>
<td>absH6high_4n5</td>
<td>10</td>
<td>6405.74</td>
<td>6405.74</td>
<td>150</td>
<td>150</td>
<td>429.64</td>
</tr>
<tr>
<td>absH6high_5n5</td>
<td>20</td>
<td>5972.78</td>
<td>5972.78</td>
<td>234</td>
<td>234</td>
<td>808.44</td>
</tr>
<tr>
<td>absH6low_1n5</td>
<td>29</td>
<td>4657.15</td>
<td>4657.15</td>
<td>252</td>
<td>252</td>
<td>453.99</td>
</tr>
<tr>
<td>absH6low_2n5</td>
<td>29</td>
<td>4221.00</td>
<td>4221.00</td>
<td>336</td>
<td>336</td>
<td>1252.38</td>
</tr>
<tr>
<td>absH6low_3n5</td>
<td>6</td>
<td>7703.53</td>
<td>7703.53</td>
<td>78</td>
<td>78</td>
<td>114.51</td>
</tr>
<tr>
<td>absH6low_4n5</td>
<td>10</td>
<td>4495.31</td>
<td>4495.31</td>
<td>150</td>
<td>150</td>
<td>398.34</td>
</tr>
<tr>
<td>absH6low_5n5</td>
<td>8</td>
<td>3879.09</td>
<td>3879.09</td>
<td>102</td>
<td>102</td>
<td>397.61</td>
</tr>
</tbody>
</table>

Table 1: Results for selected IRP instances. K=3

A large number of combinatorial cuts represents also a challenge for the MIP solvers tested. Current research is now focused on solving the master efficiently.

References


A branch and cut algorithm for single and multiple vehicle IRP with pickups and deliveries

C. Archetti and M.G. Speranza
Department of Industrial Economics and Management,
University of Brescia, Brescia, Italy

M. Boccia, A. Sforza and C. Sterle
Department of Electrical Engineering and Information Technology,
University Federico II of Naples, Naples, Italy
Email: maurizio.boccia@unina.it

1 Introduction

The Inventory Routing Problem (IRP) consists in the distribution of one or more products from a supplier to a set of customers over a discrete planning horizon [3]. IRP and its variants have been widely tackled in the last 30 years by the operations research community, mainly motivated by real applications arising in different contexts and in particular in supply chain management [5]. In this work we focus on a particular class of IRP where pickup and delivery operations integrate vehicle routing and inventory management decision problems. This problem, referred in the following as IRP-PD, has been introduced in its basic version in [1], where the authors tackle the single-commodity, single-vehicle land-based inventory routing problem with pickups and deliveries (1-1-IRP-PD). IRP-PD has been scarcely treated in literature and, to the best of authors’ knowledge, contributions dealing with similar problems concern the automated teller machine replenishment [6] and maritime transportation [4]. In this context, the contribution of this work can be summarized as follows:
- We extend the 1-1-IRP-PD formulation of [1] to the single commodity, multiple vehicle variant of the IRP-PD, in the following referred to as 1-M-IRP-PD. In particular, we consider a flow based and a vehicle index formulation (Sec. 2).
- We reformulate the 1-1-IRP-PD and the 1-M-IRP-PD using the lot-sizing inequalities introduced in [2] for the delivery nodes to the pickup nodes (Sec. 3).
- We propose a branch-and-cut algorithm for the 1-1-IRP-PD and the 1-M-IRP-PD which integrates/adapts valid inequalities by [1] and lot-sizing and cut inequalities by [2].
algorithm is experienced and validated on 1-I-IRP-PD benchmark instances coming from
the literature and on new original 1-M-IRP-PD instances (Sec. 3).

2 Single-commodity, multi-vehicle IRP-PD formulation

The 1-M-IRP-PD is defined in a discrete time horizon \( T = 1, \ldots, T \) on a network \( \mathcal{G} = (\mathcal{N}', \mathcal{A}) \),
where \( \mathcal{N}' \) and \( \mathcal{A} \) are the node and the arc sets. \( \mathcal{N}' = \{0 \cup \mathcal{N}\} \), where 0 is the depot, and \( \mathcal{N} = \{\mathcal{N}_P \cup \mathcal{N}_D\} \) is the union of the pickup (\( \mathcal{N}_P \)) and delivery node (\( \mathcal{N}_D \)) sets. Each node \( i, i \in \mathcal{N}' \) is associated with five parameters, \( d_{it}, h_i, I_{i0}, L_i \) and \( U_i \), where: \( d_{it} \) is the quantity made available (demanded) at pickup (delivery) node \( i, i \in \mathcal{N} \) in time period \( t, t \in T \); \( h_i \) is the unitary holding cost at node \( i \); \( I_{i0}, L_i \) and \( U_i \) are the initial inventory level, lower and upper inventory limits, respectively, at node \( i \). For the depot we consider an initial inventory level \( I_{00} \), but no upper and lower limits are defined. Finally, a cost \( c_{ij} \) is associated to each arc \((i, j) \in \mathcal{A}\) and a fleet of \( m \) vehicles with capacity \( Q \) is available. We assume that each vehicle can perform just one route during each period composing the time horizon. Using this setting, for each time period \( t, t \in T \), the following variables are defined:

1. Node variables \( q_{it} \) (continuous), \( y_{it} \) (binary); \( y_{0t} \) (integer) and \( I_{it} \) (continuous): \( q_{it} \) identifies the quantity picked up (delivered) at node \( i, i \in \mathcal{N}_P \) (\( i \in \mathcal{N}_D \)); \( y_{it} \) assumes value 1 if the node \( i, i \in \mathcal{N} \) is visited, 0 otherwise; \( y_{0t} \) counts the number of vehicles used (routes performed); \( I_{it} \) is related to the inventory level at node \( i, i \in \mathcal{N} \).

2. Arc variables \( x_{ijt} \) (binary) and \( l_{ijt} \) (continuous): \( x_{ijt} \) assumes values 1 if arc \((i, j) \in \mathcal{A}\) is used, 0 otherwise; \( l_{ijt} \) gives the quantity traversing the arc \((i, j) \in \mathcal{A}\).

On this basis, the flow formulation for the 1-M-IRP-PD is the following one:

\[
\min z = \sum_{(i,j) \in \mathcal{A}} \sum_{t \in T} c_{ij}x_{ijt} + \sum_{i \in \mathcal{N}'} \sum_{t \in T} h_i I_{it}.
\]

\[
\sum_{j \in \mathcal{N}'} x_{ijt} - \sum_{j \in \mathcal{N}'} x_{j,it} = 0, \quad \text{for } j \in \mathcal{N}', t \in T, (2)
\]

\[
\sum_{j \in \mathcal{N}'} x_{ijt} - y_{it} = 0, \quad \text{for } j \in \mathcal{N}', t \in T, (3)
\]

\[
\sum_{i \in S} \sum_{j \in S} x_{ijt} \leq \sum_{i \in S} y_{it} - y_{mt}, \quad \text{for } S \subseteq \mathcal{N}, m \in S, t \in T, (4)
\]

\[
I_{it} = I_{i0} - I_{it-1} - d_{it} + q_{it} = 0, \quad \text{for } i \in \mathcal{N}_P, t \in T, (5)
\]

\[
I_{it} = I_{i0} - I_{it-1} + d_{it} - q_{it} = 0, \quad \text{for } i \in \mathcal{N}_D, t \in T, (6)
\]

\[
I_{it} = I_{0t} + \sum_{i \in \mathcal{N}_D} q_{it} - \sum_{i \in \mathcal{N}_P} q_{it} = 0, \quad \text{for } t \in T, (7)
\]

\[
I_{0i} = I_{i0}, \quad \text{for } i \in \mathcal{N}', (8)
\]

\[
I_{i,t-1} + q_{it} \leq U_i, \quad \text{for } i \in \mathcal{N}_D, t \in T, (9)
\]

\[
I_{it} \geq L_i, \quad \text{for } i \in \mathcal{N}_D, t \in T, (10)
\]

\[
I_{it} \leq U_i, \quad \text{for } i \in \mathcal{N}_P, t \in T, (11)
\]
Objective function (1) minimizes the sum of inventory and routing costs. Constraints (2) and (3) are related to route construction. Constraints (4) are subtour elimination constraints to be dynamically added to the formulation. Constraints (5), (6) and (7) are balance constraints for pickup nodes, delivery nodes and the depot, respectively. Constraints (8) impose the initial inventory conditions. Constraints (9) and (10), and constraints (11) and (12), define lower and upper inventory limit conditions for delivery and pickup nodes, respectively. It is important to note that they are coherent with the following sequence of operations: first the node is served, then the demand is satisfied, and finally the inventory level is calculated. Constraints (13) impose the limits on the quantity that can be picked up or delivered in a node. Constraints (14) and (18) are consistency constraints. Constraints (15) impose that the number of routes in each time period does not exceed the number of available vehicles \(m\). Constraints (16) and (17) are load balance constraints, while constraints (19) limit the quantity distributed from the depot with respect to the available inventory level. Finally, constraints (20), (21) and (22) are the requirements for the variables. This formulation can be obviously used also for the 1-1-IRP-PD imposing \(m = 1\). Instead, if we explicitly define the set of vehicles \(K\), we can modify this formulation to obtain a vehicle index formulation specifying each variable and constraint for a generic vehicle \(k\). For the sake of the brevity, we omit the needed operations.

3 IRP-PD reformulation, branch-and-cut and results

The formulations proposed for the 1-M-IRP-PD and 1-M-IRP-PD can be strengthened by additional inequalities which exploit the lot-sizing substructure of the IRPs. As discussed in [2], when dealing with IRP, each delivery node is associated to a lot-sizing problem which can be reformulated as a unit flow problem. When all the demands \(d_i\) are constant overall the time horizon and the delivery node inventory upper limit \(U_i\) is a multiple of the demands, i.e. \(U_i = V \ast D_i, V \in [2,3]\), we can strengthen the IRP formulation by several
valid inequalities which partially describe the unit flow problem polyhedron. For the sake of the brevity we do not report here these inequalities and we address the interested reader to [2]. In this work we generalize the previous inequalities for the delivery nodes to the pickup nodes. This can be done just substituting $I_t$ by $U - I_t$ in all the inequalities. This substitution is intuitive and it can be explained considering that in delivery nodes we fill-up related stock until the maximum level, whereas in pickup nodes, we empty the related stock, taking a quantity which cannot be higher than the maximum stock level. All these inequalities, together with the ones by [1], are valid for both 1-1-IRP-PD and 1-M-IRP-PD, regardless the used formulation, and, being polynomial in the number, can be all added to the previous optimization models. Moreover, they have been integrated in an original branch-and-cut algorithm which dynamically adds subtour elimination constraints and exploits also the adaptation of cut inequalities derived by [2]. The proposed algorithm has been tested on the 1-1-IRP-PD instances proposed in [1], succeeding in optimally solving most of the unsolved instances with a lower computation time. Then, it has been experienced on original 1-M-IRP-PD instances, obtained adapting the previous ones to the multiple vehicle variant. The obtained results are satisfactory both in terms of quality of solution and computation time, since most of the instances have been solved to optimality or near optimality within one hour computation time.

References


Auction-based mechanisms in carrier collaborations: challenges and limitations

Margaretha Gansterer Richard F. Hartl Rudolf Vetschera
Department of Business Administration, University of Vienna, Vienna, Austria
Email: margaretha.gansterer@univie.ac.at

1 Introduction

Collaborative relationships have been recently identified by [1] to be one of the big trends in transportation. Collaboration among carriers may reduce inefficiencies in carrier operations and generate economic and ecologic benefits for the participants involved as well as for the society.

In horizontal logistics collaboration, several companies pool their transportation requests in order to execute them more efficiently. The results of these initiatives are impressive: double-digit efficiency improvements of up to 30% have been reported [2]. Such coalitions are commonly realized by using auction-based exchange systems. In combinatorial auctions requests are not traded individually but are combined into packages. The main reason for this is that because of synergies a specific bundle might have a different value to the partners than the sum of the individual requests. An example for such an exchange among pickup and delivery carriers is illustrated in Figure 1.

A combinatorial transportation auction would typically follow a 5-phase procedure [3]. First, carriers select request that they want to auction-off, then the auctioneer generates bundles of requests and offers them to the participants. They bid on offered bundles by calculating the marginal profit of each bundle. Finally, requests are allocated to carriers according to their bids, so that the total profit for the coalition is maximized. Gained profits are distributed among the carriers. Each of these stages has been intensively researched in the literature, e.g [4], [5], [6]. However, exchange mechanisms rely on information about costs provided by carriers and thus have to ensure that this information is given correctly. The resulting problems of incentive compatibility have only rarely been addressed in literature.
2 Proposed mechanisms

Exchange mechanisms should fulfill several properties such as: (i) incentive compatibility (carriers should have incentives to provide correct information), (ii) efficiency (requests should be allocated in a way that minimizes overall costs), (iii) budget balance (the mechanism should not result in a deficit for the auctioneer), and (iv) individual rationality (no carrier should be worse off after participating in the exchange). However, these requirements cannot all be fulfilled at the same time. Since we focus on carriers’ strategic behavior, we design an incentive compatible pricing mechanism that induces reporting true costs. It is based on the well-known Vickrey-Clarke-Groves (VCG) mechanism, which uses the concept of second price auctions. We show that the interpretation of a second price is not straightforward. Carriers have to bid on bundles that potentially include their own requests, since they want to exchange requests, and therefore have to give buying and selling prices, simultaneously.

Since requests are unique and each request is provided by only one carrier, dropping a carrier (as it is usually done in VCG mechanisms) leads to a "chain reaction" of modifications in request allocation. This overestimates the impact of one single carrier and leads to inflated "second prices". We therefore propose a modification of the VCG mechanism, that takes into account that such effects cannot uniquely be identified with one bidder but an entire set of bidders, which we call a "super bidder". We follow the logic of the VCG mechanisms, which requires that bidders should not be able to influence the price they are paying via their bids. The proposed mechanism is therefore incentive compatible.
Table 1: Comparison of the proposed mechanisms (IR: individual rationality is violated; AL: auctioneer’s loss in % of the total initial profit; IR_fee: IR violation if participation fee is paid).

<table>
<thead>
<tr>
<th>Instance</th>
<th>VCG</th>
<th>Super Bidder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IR</td>
<td>AL</td>
</tr>
<tr>
<td>O1</td>
<td>0%</td>
<td>1.5%</td>
</tr>
<tr>
<td>O2</td>
<td>0%</td>
<td>4.9%</td>
</tr>
<tr>
<td>O3</td>
<td>0%</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

3 Computational study and discussion

We numerically analyze the proposed mechanisms by applying them to available test instances. We consider carriers having less than truckload requests. The instances (O1, O2, O3) cover different scenarios in terms of (i) degree of customer overlaps and (ii) distance of requests to the carriers’ depots [4].

3.1 Results

Since both mechanisms can be proven to be incentive compatible, and efficient, we focus on the properties of budget balance and individual rationality. While VCG is individual rational, this property does not hold for the proposed supper bidder mechanism. Thus, we first quantify how often individual rationality is violated. None of the mechanisms is budget balanced, and VCG will per definition lead to a higher loss for the auctioneer. This loss can be compensated by a participation fee paid by the carriers. We assume that the auctioneer’s loss is distributed equally to the carriers. Obviously, such a fee might lead to payments that violate individual rationality. We therefore evaluate, whether individual rationality holds, if carriers are charged with a participation fee. Results are summarized in Table 1.

While VCG is 100% individual rational if carriers do not have to pay a participation fee, individual rationality is violated if the auctioneer’s loss has to be compensated. When distributing the loss equally among participants, no individual rational solution can be reached in 52.5% (O1), 42.5% (O2), and 35.0% (O3) of the instances, respectively. However, the participation fee has no major impact on the individual rationality in the super bidder mechanism.

Our computational results show that both of the proposed mechanisms come with some limitations with respect to the four desired properties. VCG is efficient, and incentive compatible. But it is not budget balanced, and not individual rational as soon as carriers have to compensate auctioneers’ losses. The super bidder mechanism is less costly for
the auctioneer, but individual rationality cannot be guaranteed. Even if carriers do not pay for the loss of the auctioneer, there might be participants who by participating are worse-off.

3.2 Conclusion

Our study shows that strategic behavior in auction-based carrier collaborations can be avoided, but only at the price of violating individual rationality of participants. It cannot be guaranteed that all carriers benefit from participation. Vice versa, if carriers insist on proven individual rationality, untruthful behavior might become the dominant strategy.

However, the identification of a profitable untruthful bidding strategy is not straightforward. Due to the synergies among transportation requests, and the fact that carriers sell and buy requests simultaneously, the outcome of strategic bidding is hard to predict. We emphasize this by additional computational results, where we observe that a natural approach of untruthful bidding is not the dominant strategy.

References


1 Introduction

Recent literature surveys by [1] and [2] reveal a considerable growth in the number of articles and applications of collaborative transportation. This literature positions collaborative transportation as an important mechanism to reduce cost, increase profits and service levels, and reduce negative environmental effects. In parallel, the literature on location-routing has also grown considerably. The central problem in this literature is to locate facilities while simultaneously finding routes to serve customers from these facilities. Recent reviews by [3] and [4] give account of a broad range of applications as well as a considerable progress in solution methods for location-routing.

Despite the growing interest in both collaborative transportation and location-routing, the integration of these two areas remains fairly unattended. The only exceptions, to our knowledge, are [5] and [6], which present promising studies on the benefits of collaboration in location-routing. Besides the academic interest, the intersection of these two areas motivates from practical situations, such as the installation of urban consolidation centres for city logistics and the formation of strategic alliances. For example, [7] recounts the case of the Bristol-Bath freight consolidation centre from the perspective of its users and points to pricing and cost coverage as important factors for success or failure. If a consolidation centre is located in the periphery of a city to serve users in the city centre by shared routes, the users will naturally be concerned about the cost of the service and how this compares to the non-shared solution. The location of the centre may, therefore, play an important role in the willingness of users to adopt the shared solution. Another example
in practice is given by the alliance between Colgate-Palmolive, GlaxoSmithKline, Henkel and Sara Lee in France [8]. The alliance started in 2005 with cooperation on routing only, but subsequently led to a decrease in the number of facilities of the alliance from four to only one.

In this article, we formally introduce the collaborative location-routing problem and study its properties by combining tools from integer programming and cooperative game theory. Our results provide first a theoretical characterization of this problem and, second, insights derived from intensive computational experiments.

2 Problem definition and modelling framework

Given a set of potential facility locations, a set of customers and their corresponding demands, the location-routing problem aims at finding locations and routes that minimize the total travelling costs, costs of using vehicles and opening costs of facilities while assuring that all customers are visited and their demand is satisfied. Other constraints also ensure that each vehicle is used only once and departs from open facilities where it also returns after visiting the customers. This problem and many of its variants can be modelled by integer linear programming formulations, extensively discussed in [3] and [4]. Our interest focuses in a collaborative version of the location-routing problem, that is, where the customers belong to different companies and the potential location of facilities may also be associated to different companies. When companies collaborate, the overall problem opens opportunities to combine their customers within a same route and to serve their demands from shared facilities. To model this situation, we use a cooperative game theory framework. Let $N = \{1, \ldots, n\}$ denote the set of all companies, $K$ the set of all subsets of $N$ and $c: K \rightarrow R$ a function assigning to each coalition $k \in K$ its optimal cost (by convention, $c(\emptyset) = 0$). The optimal cost for each $k \in K \setminus \{\emptyset\}$ can be found by solving the corresponding location-routing problem. The pair $(N, c)$ defines a transferable utility game which we call the standard location-routing game. In addition, we define and study two variants. The first one introduces an upper bound in the maximum number of depots that can be opened. The second one introduces an upper bound in the supply available at the depots.

3 Results

We provide theoretical results and numerical results. The theoretical results aim at characterizing the game and the numerical results aim at gaining some insights on the behaviour of the game and cost allocation methods.
3.1 Theoretical results

First, the standard location-routing game is subadditive. This means that $c(S \cup T) \leq c(S) + c(T) \forall S, T \subseteq N; S \cap T = \emptyset$. A stronger property is convexity, which means that $c(S \cup T) + c(S \cap T) \leq c(S) + c(T) \forall S, T \subseteq N$. We prove that this property is, in general, not satisfied by the location-routing game.

Studies of transferable utility games naturally lead to studies of cost allocations which prescribe the costs to be paid by particular companies within the cooperation. An important concept of stability is the core of the game. This is a set of cost allocations, represented as vectors $(x_1, \ldots, x_n)$, that satisfy efficiency ($\sum_{j \in N} x_j = c(N)$) and rationality ($\sum_{j \in S} x_j \leq c(S) \forall S \subseteq N$) conditions. An allocation in the core guarantees that no set of companies comes out better off deviating and forming an independent coalition. This has been recognized as one of the fundamental features of stability in the literature and also as an essential condition to sustain the cooperation in practice, thus it turns interesting to study whether a game admits or not allocations in the core. We prove that, in general, the core of the location-routing games is not necessarily non-empty.

3.2 Numerical results

Since the game might be non-convex and its core empty, it turns interesting to investigate how often this happens, how different allocation methods behave when the core is non-empty, and which factors affect this behaviour. For this purpose, we perform a numerical experiment in 1,000 instances including 3 companies and 17 nodes. These instances are solved using AMPL/Gurobi. The core of the standard location-routing game is non-empty in 99.9% of the instances, whereas in the limited-supply case, this occurs in 92.5%. As for the cost savings, these range from 7% to 52%. Interestingly, collaboration in location-routing does not necessarily imply a decrease in the total travelling distance. In fact, we observe 10.3% of instances where the total travelling cost increases in comparison to the non-collaborative solution. The intuition for this is that, since collaboration aims at reducing the overall costs (which include travelling, facility and vehicle costs), in some situations the selection of a reduced number of locations might offset the increase in costs due to larger travelled distances.

Among the instances with non-empty core, core-guaranteed methods such as the nucleolus and the equal profit method dominate in terms of stability, as by definition they provide allocations in the core. For the Shapley value, we observe a reasonably good performance, as it provides stable allocation for 98.2% of the instances, whereas proportional methods only in 51 – 56% of the instances. When the instances do not provide stable allocations, the violation of rationality constraints, as measured by the epsilon-core concept, is rather small (in the range 0.01 – 5.1%).
4 Conclusion

This work is, to our knowledge, the first one studying collaboration in location-routing. Using a cooperative game theory framework, we have derived characteristics of this problem in terms of subadditivity, convexity, core-stability, and non-emptiness, as well as performed numerical experiments. The potential for savings is rather high and comparable to related problems. This, together with important applications in practice, motivates further research in this problem. Interesting directions for future research are finding structural conditions to guarantee stability and investigating the effect of approximate solution approaches for large-dimension problems in comparison to the exact version of the game.

References


Shared Capacity Routing Problem with Transfer Points

Joydeep Paul, Niels Agatz
Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University, The Netherlands

Martin Savelsbergh
Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA

Corresponding author: paul@rsm.nl

1 Introduction

An increasingly popular omni-channel fulfillment model is one in which customers can pick up goods ordered online at an in-store pickup point (PUP). The PUPs are typically not supplied from the store inventory but by a dedicated e-fulfillment warehouse. This often means that the stores are visited by different vehicles to replenish the store inventory and to supply the PUPs. Motivated by a collaboration with the leading omni-channel grocery retailer in the Netherlands, we develop a strategy to consolidate the transportation flows to the stores without completely centralizing the planning but by sharing the capacity of vehicles across different channels.

This works as follows. Consider two carriers that deliver to a set of shared customers from their own warehouses. The sequence of stops (i.e., the customers visited) for the delivery routes of the fixed-route carrier are fixed in advance (i.e., store replenishment), whereas the flexible-route carrier (i.e., PUP supply) determines the delivery routes based on the actual demand. If there is capacity available, the flexible-route carrier may piggy-back on the delivery routes of the fixed-route carrier. This can reduce the system-wide travel distance and the number of customer visits.

To outsource one or more customers to the fixed-route carrier, the flexible-route carrier can either move the associated demand to the fixed-route carrier’s warehouse, or to one of the shared customer locations. In both cases, the demand is subsequently delivered to one or more customers by the fixed-route carrier on the remaining route after the transfer point. In [1], we considered a setting in which transfers could only take place at the fixed-route carrier’s warehouse by using dedicated transfer trips. In this paper, we
also consider the shared customer locations (i.e., stores) as possible transfer points, which increases the potential for capacity sharing. Only allowing a transfer of loads at the fixed-route carrier’s warehouse limits the consolidation opportunities as the fixed-route carrier’s vehicles typically do not have much spare capacity when they depart from the warehouse. Later in a fixed-route carrier’s delivery route, after delivering to some customers, more capacity is available for transfers (and thus consolidation). Another advantage is that transfers may be accomplished with shorter detours for the flexible-route carrier.

We introduce the Shared Capacity Routing Problem with Transfer Points (SCRPT) which aims to minimize the total cost of the flexible-route carrier to deliver to all its customers either directly or by outsourcing to the fixed-route carrier via the transfer points.

2 Problem Description

We model the SCRPT on a complete directed graph $G = (V, A)$. Here, $V = \{o, d\} \cup N$, where $o$ and $d$ are the warehouses of the flexible-route carrier and fixed-route carrier respectively, and $N$ is the set of customer locations the flexible-route carrier has to visit. Each customer $i \in N$ has a demand $q_i \geq 0$, which has to be fulfilled from the warehouse $o$.

Demand of each customer $i \in N$ can be fulfilled directly by the flexible-route carrier, or by outsourcing it to the fixed-route carrier. We do not allow splitting of demand while serving a customer, which means that a customer is visited exactly once by the flexible-route carrier or its demand is fully outsourced.

To fulfill demand directly, the flexible-route carrier has a sufficient number of vehicles available, each with capacity $Q$. For simplicity, we assume $Q \geq q_i, \forall i \in N$. Every vehicle drives a route, which is a simple cycle in $G$ starting and ending at the warehouse, and fulfills demand of each customer that is visited along the route. A route is considered feasible if the total demand of the customers that are visited do not exceed the capacity $Q$. Furthermore, $c_{ij}$ is the cost of traversing an arc $(i, j) \in A$. We assume that $c_{ij}$ satisfies the triangle inequality.

To fulfill demand by outsourcing, consider the set of shared customers $S \subseteq N$ that is visited by both the carriers. Only the demand of these customers can be outsourced. Let $R$ be the set of all fixed-route carrier routes, which we refer as fixed routes. For each fixed route $r \in R$, $S_r \subseteq S$ is the set of shared customers visited in the fixed route $r$. We denote the set of transfer points for each fixed route $r$ as $T_r = S_r \cup \{d\}$. Let $e^*_r$ be the spare capacity available in the fixed route $r$ departing at location $i$ and $e_i$ be the maximum temporary storage capacity at location $i$ to accommodate transfers. Let $\sigma^*_i$ be the time of arrival of the vehicle of the fixed-route carrier at $i$.

Any location $i \in T_r$ can be used as a transfer point by the flexible-route carrier to
outsource the demands of customers selected from the set $O^r_i \subseteq S_r$, which is the set of shared customers visited after $i$ in the fixed route $r$. For a feasible transfer at location $i \in T_r$, the flexible-route carrier needs to visit the transfer point $i$ before $\sigma^r_i$ and the demand of the customers outsourced at $i$ must be less than equal to $\min(e^r_i, e^r_i)$. Note that a flexible-route carrier can visit a location $i \in N \cup \{d\}$ to deliver the demand $q_i$ only or to transfer the demand of outsourced customers only or both. There is a fixed handling cost $\tau$ per unit of outsourced demand at the transfer point.

The objective of the SCRPT is to determine the location the transfer points with the associated outsourced customers and corresponding routes to make the transfers and deliver to the non-outsourced customers so that the total costs are minimized.

3 Methodology

As the SCRPT reduces to the vehicle routing problem when there is no spare capacity in the fixed routes, the SCRPT is NP-hard. Hence, we focus on developing heuristics to solve the problem for large instances. We develop a two-stage heuristic in which the transfer points and the associated outsourced customers are chosen in the first stage and in the second stage we solve the routing problem for the transfer points and the non-outsourced customers.

In Stage 1, we choose the transfer points to maximize the number of outsourced customers. As a result, there will be lesser number of customer locations to visit, which typically leads to reduction in routing costs. The maximization of number of outsourced customers can be modeled as a multiple knapsack problem where the spare capacity at each transfer point constrains the number of customers that can be outsourced.

The output of Stage 1 provides the set of transfer points and the associated customers outsourced. The choice of the transfer points also sets the latest time of arrival for the flexible-route carrier at the transfer points. In Stage 2, we solve the routing problem for the transfer points and the non-outsourced customers using the ALNS heuristic as proposed by Pisinger and Ropke [2]. Note that for solving the routing problem, the demand of the transfer point has to adjusted to include the demand of the outsourced customers at that transfer point.

4 Results & Discussion

In this section, we present some initial results to show the savings obtained by consolidating customers using transfer points. We generate the instances based on the VRP instances from the VRP-lib repository. For the flexible-route carrier we use the customer demand and vehicle capacities as given in these instances.
We assume the fixed-route carrier visits the same set of customers, i.e., all customers are shared. We use the demand distribution given in the VRP-lib instance to generate five different demand scenarios for the customers of the fixed-route carrier and accordingly build the fixed routes for each scenario. We assume that the fixed carrier does not pick-up any load from its customers, so the spare capacity is always non-decreasing as it traverses its route. The values in Table 1 corresponding to each instance are averages over five different realizations of fixed routes.

<table>
<thead>
<tr>
<th>Instance</th>
<th>No. of Transfer Points</th>
<th>No. of Outsourced Customers</th>
<th>Savings Transport Costs (%)</th>
<th>Savings Customer Visits (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-n32-k5</td>
<td>10</td>
<td>18</td>
<td>9.8</td>
<td>59.4</td>
</tr>
<tr>
<td>A-n48-k7</td>
<td>16</td>
<td>27</td>
<td>8.0</td>
<td>57.9</td>
</tr>
<tr>
<td>A-n64-k9</td>
<td>22</td>
<td>35</td>
<td>8.6</td>
<td>54.9</td>
</tr>
<tr>
<td>A-n80-k5</td>
<td>25</td>
<td>49</td>
<td>10.0</td>
<td>62.5</td>
</tr>
</tbody>
</table>

In Table 1 we show the number of locations used as transfer points and also the total number of customers outsourced to the fixed carrier. We see that the average savings in transport costs across all instances are around 9% while the for customer visits the average savings are around 59%. This difference can be attributed to the choice of transfer point locations. As we maximize the number of outsourced customers in the Knapsack problem of Stage 1, we do not take into consideration the location of the transfer point. We plan to experiment with different objectives in the Knapsack problem to account for both distance and number of outsourced customers.

The initial results of our experiments show potential savings due to consolidation of customer demands by sharing vehicle capacity. We plan to improve our heuristic and benchmark it against optimal solutions. We will conduct an extensive computational study on both real-life instances and artificial instances to understand the effect of consolidation under different settings.

References


1 Introduction

Companies in need of transportation services may choose to join forces by combining their logistical networks and, as a result, benefit from reduced transportation costs and CO\textsubscript{2} emissions. Case studies as presented by [1] and [2] show that logistics service providers are able to achieve significant savings through collaboration. Unfortunately, splitting the costs among the cooperating companies proves to be a difficult task. Many companies do not want to exchange any information or allow their competitors to benefit from cooperative transportation. To stimulate the adoption, it is crucial to give companies insight into the way costs are allocated, and to create procedures that suit their needs.

The problem of assigning a fair cost allocation for cases in which each logistics service provider has exactly one customer has been studied extensively in the literature. This problem is often approached through game theory and referred to as the Vehicle Routing Game. However, the Vehicle Routing Game may not be suitable to model collaborations between multiple large companies. To that end, we introduce a new game and relax the assumption that each player has exactly one location.

We consider the problem of assigning a fair cost allocation among multiple logistics service providers, each with multiple customers, should they choose to collaborate. In the following we introduce a new game for the problem and propose an algorithm based on cutting-plane techniques.
Game. The Joint Network Vehicle Routing Game is a generalisation of the Vehicle Routing Game.

Let the set of all players be denoted as $N$. A subset of the players, also known as a coalition, is denoted as $S \subseteq N$. The cost of a coalition of players $c(S)$ is given by the optimal routing costs to service all customers of coalition $S$. The goal of the game is to find a suitable cost allocation such that no player has any incentive to stop the cooperation. It can occur that a coalition can cut more costs by collaborating with a selection of all players. This incentivises players to only collaborate with some players rather than all other players $N$. Such behaviour is unwanted as the goal is to stimulate all players to cooperate and to minimise the total costs. A solution to stimulate all rational-entities into collaborating is said to be in the core. The core can be defined as all feasible solutions to the following linear programming problem:

$$\min \ 0$$

Subject to:

$$\sum_{i \in S} y_i \leq c(S) \ \forall \ S \subseteq N \quad (1)$$

$$\sum_{i \in N} y_i = c(N) \quad (2)$$

$$y_i \geq 0 \ \forall i \in N \quad (3)$$

where $y_i$ represents the costs allocated to player $i$. The aim of the problem is to find a solution which stimulates all of the players to cooperate. Dependent on the objective, a specific solution in the core can be generated, for the general case we set the objective to 0. One should note that an exponential amount of constraints in the amount of players, given by (1), may have to be evaluated. For each of these constraints a vehicle routing problem over all customers of the players in the coalition $S$ has to be solved. As such, we propose an efficient method to generate these constraints for the Joint Network Vehicle Routing Game.

3 Solution method

In the following we propose a cutting-planes method to solve the problem of finding a cost allocation in the core. Cutting planes have also successfully been applied by [3] and [4] suggesting that a similar methodology could be effective for the Joint Network Vehicle Routing Game. To this end, we will generate constraints (1) by means of a separation problem.

3.1 Separation problem

The separation problem consists of a prize-collecting vehicle routing problem. The goal of the separation problem is to determine a coalition that violates a core constraint. Let $V_S$ denote the customers belonging to players $S \subseteq N$. The prize-collecting vehicle routing problem is a special case of the vehicle routing problem in which a prize $y_i^*$ is collected if all customers $V_{\{i\}}$ belonging to player $i$ are visited. In our case the collected prize equals the allocated cost $y_i^*$ to
player $i$, as determined by the linear program given by a subset of constraints (1) and constraints (2)-(3). Let the decision variable $z_i$ indicate whether all customers of player $i$ have been visited. Furthermore, let $K$ be the set of all feasible routes $k$ such that a route $k$ visits a subset of the customers in $V_N$ such that the sum of the total demand of the visited customers does not exceed the capacity $Q$. The cost of a route $k$ is denoted by $c_k$ and $a^v_k$ is defined as the parameter indicating whether location $v$ is on route $k$. The decision variable $x_k$ indicates whether route $k$ is included in the optimal solution. This gives rise to the following set partitioning problem:

$$\max \sum_{i \in N} y_i^* z_i - \sum_{k \in K} c_k x_k$$

Subject to:

$$\sum_{k \in K} a^v_k x_k = z_i \quad \forall v \in V_N, i : v \in V_i \quad (4)$$

$$x_k \in \{0,1\} \quad \forall k \in K \quad (5)$$

$$z_i \in \{0,1\} \quad \forall i \in N \quad (6)$$

Constraints (4) ensure that each player is visited exactly once or not at all. Constraints (5) ensure that each route is used exactly once or not at all in the final solution. Constraints (6) ensure that the indicator $z_i$ is binary. If, for an optimal solution, the collected rewards are greater than the routing costs, a core constraint has been violated. The violated core constraint belongs to the coalition $S = \{i : z_i = 1\}$.

After determining the most violated constraint, the corresponding optimal costs of that coalition $S$ have been determined as well. These optimal costs are given by:

$$c(S) = \sum_{k \in K} c_k x_k. \quad (7)$$

where the $x_k$ are chosen as in the optimal solution to the separation problem. This means that the constraint can immediately be added to the linear programming formulation of the core.

However, the separation problem does not have to be solved to optimality to indicate any violated constraints. By relaxing constraints (5) the problem becomes easier to solve. Now, if the optimal solution value is positive the coalition potentially violates its corresponding constraint. As the set-partitioning formulation is known for its tight bounds, it is expected that most potentially violating coalitions are indeed violating the constraint. In order to determine whether the constraint is violated, a Vehicle Routing Problem over all customers in $S$ has to be solved. Finally, the constraint is added to the problem along with a cut in order to prevent the coalition from being generated again.

### 3.2 Implementation

Both the Vehicle Routing Problems and the separation problem are solved using a set partitioning formulation in a branch-and-price framework. The problem of assigning a cost allocation in the core is always initialised with the single player constraints. The other constraints are
generated using the separation problem. The pricing problem is solved using a combination of local neighbourhood searches, heuristics and exact labelling algorithms. Furthermore, several cuts for the vehicle routing game are adapted to the new separation problem.

4 Results and conclusions

The results of the proposed methods will be presented and are compared to a basic algorithm in which the routing costs are explicitly solved for each coalition. Randomly generated instances have been used to obtain preliminary results. The maximum number of players considered is 12, whereas the maximum number of customers considered is 25. Preliminary results show that the algorithm which incorporates the exact separation problem outperforms the basic method by up to 80 times if the amount of players is high and the number of customers per player is low. However, if the amount of customers per player increases, the algorithm is able to solve less instances, due to a set time limit, albeit being on average 2.6 times faster for the instances which were solved.

The fact that the constraint generating method was able to solve less instances within the time limit can be accounted for by an relatively large integrality gap between the relaxed and integer versions of the separation problem. This gap causes the branch-and-price procedure to create lots of nodes and slow down as a result. We expect the implementation of cuts for the separation problem to partially alleviate this issue. Nonetheless, the constraint generating algorithms show great promise provided that a significant performance gain has already been achieved. We believe that by tailoring our algorithm more to the problem the method can become viable for solving the Joint Network Vehicle Routing Game.

Furthermore, the method and separation problem can easily be extended to other well-known allocation methods such as the Nucleolus, Lorenz method and the Equal Profit Method.

References


The time dependent pickup and delivery problem with time windows

Peng Sun
Kuhne Logistics University, Hamburg, Germany

Lucas P. Veelenturf, Tom Van Woensel
School of Industrial Engineering,
Eindhoven University of Technology, Eindhoven, The Netherlands

Mike Hewitt
Quinlan School of Business,
Loyola University Chicago, Chicago, United States

Email: l.p.veelenturf@tue.nl

1 Introduction

Following the rapid development of E-commerce and just-in-time logistics (including same day delivery), we observe an increased need for demand-responsive freight transportation systems, which provide a flexible point-to-point service based on booking requests. The pickup and delivery problem with time windows (PDPTW) is an important building block in these types of transportation systems.

In this study, we focus on four variants of the time-dependent pickup and delivery problem with time windows. In the classical PDPTW, a set of point-to-point transportation requests with predefined customer demands must be collected from pickup nodes and transported to delivery locations. These requests come with time windows and are served by a fleet of vehicles with limited capacity. The objective of the problem is to design minimum-cost routes such that all requests are served. The majority of the relevant literature assume that the travel cost between two nodes is a constant value such as a distance metric or an average travel time estimate. While, in reality, due to traffic congestion, the travel time between two nodes does not only depend on the traveled distance, but also on the departure time at the first node. As congestion delays are in many cases highly predictable, considering time-dependent travel times when dealing with the PDPTW gives a lot of value in terms of feasibility and applicability.

Within this time-dependent setting, we study four variants, obtained based on the interplay of flexibility into two important dimensions: departure time and request ful-
fillment. First, in a time-dependent environment, travel times for vehicles are not fixed but vary according to the time of departure. Therefore, the departure time at the depot becomes an important decision variable. Under different labor regulations or operational situations, in some cases flexible departure times are allowed, while in other cases fixed departure times are required. Secondly, either requests need to be all served, leading to pickup and delivery problem variants, or requests can be selected based upon a certain profitability criterion, leading to orienteering problem variants (see e.g. [7]).

2 Methodology

In our study we have 2 major methodologies: An exact branch-cut-and-price algorithm and a heuristic ALNS solution approach.

2.1 Exact Method

We first propose a branch-cut-and-price algorithm for the problem. In this algorithm columns are generated via a tailored labeling algorithm proposed by [5]. Various improvement techniques are used to evaluate the performance of the methodology. For example in order to derive better upper bounds and smaller branch trees, limited-memory subset-row cuts (see [2]) are dynamically added. In the end two methods are specified. One dedicated to the problem variants where all customers need to be served and one for the variants where it is allowed to reject certain customers.

2.2 Heuristic Method

Our heuristic approach is based on an adaptive large neighborhood search (ALNS) heuristic, as ALNS has already successfully tackled a variety of related routing problems, such as the pickup and delivery problem with time windows (PDPTW) [3], the pickup and delivery problem with time windows, profits, and reserved requests (PDPTWPR) [1] and the time-dependent vehicle routing problem with soft time windows and stochastic travel times [6]. The ALNS heuristic applies multiple removal and insertion operators. In each iteration, a roulette wheel mechanism is used to choose one removal operator to partially deconstructs the current solution and one insertion operator to repair it in a different way. In the end of each iteration, a neighborhood of the current solution is obtained. For our specific problem and to efficiently handle the time-dependent travel time in the ALNS, novel mechanisms for route feasibility checks and route evaluations are proposed. As the departure time is flexible and the triangle inequality does not hold, these feasibility checks are time consuming. This makes the insertion operators expensive in terms of computation time. Furthermore, to maintain scalability on large instances and to improve the speed of the proposed ALNS, neighborhood reductions are introduced.
3 Results

In order to assess the efficiency of the proposed methods, we test them on our data sets constructed from the instances of [3]. For the exact method, we set a maximum of ten hours computation time. For the ALNS method, all the instances were solved 10 times. In Table 1 a very limited set of our results is presented for the most flexible variant of the problem. In this variant the departure time is flexible and not all requests need to be served. For each instance, the objective function value of the optimal solution (collected profits minus travel time) and the computation time of the exact method are provided. Furthermore, the computation time of the 10 ALNS runs is provided together with the percent gap of the best solution compared to the optimal solution. The number in the instance name represents the number of requests.

As can be seen, the ALNS performs well in terms of solution quality (i.e. a gap of at maximum 0.59%). For small instances (with up to 20 requests), the exact method is faster, while for large instances, the proposed ALNS is able to derive a compatibly good solution with much less processing time.

Table 1: Results of the profitable time dependent pickup and delivery problem with time windows and flexible departure times

<table>
<thead>
<tr>
<th>Instance</th>
<th>Optimal Time(Exact)</th>
<th>Time(ALNS)</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_AA20</td>
<td>706.36</td>
<td>2.70</td>
<td>115.85</td>
</tr>
<tr>
<td>M_BB20</td>
<td>843.94</td>
<td>1.90</td>
<td>119.03</td>
</tr>
<tr>
<td>M_CC20</td>
<td>964.99</td>
<td>46.80</td>
<td>175.27</td>
</tr>
<tr>
<td>M-DD20</td>
<td>942.38</td>
<td>7.40</td>
<td>193.31</td>
</tr>
<tr>
<td>M_AA25</td>
<td>1051.88</td>
<td>245.80</td>
<td>170.56</td>
</tr>
<tr>
<td>M_BB25</td>
<td>1184.67</td>
<td>441.10</td>
<td>287.27</td>
</tr>
<tr>
<td>M_CC25</td>
<td>1355.72</td>
<td>192.40</td>
<td>254.71</td>
</tr>
<tr>
<td>M-DD25</td>
<td>1303.20</td>
<td>481.10</td>
<td>170.56</td>
</tr>
<tr>
<td>M_AA30</td>
<td>1506.89</td>
<td>370.00</td>
<td>209.06</td>
</tr>
<tr>
<td>M_BB30</td>
<td>1689.39</td>
<td>13552.50</td>
<td>245.18</td>
</tr>
<tr>
<td>M-DD30</td>
<td>1627.76</td>
<td>14533.00</td>
<td>371.50</td>
</tr>
</tbody>
</table>

4 Final remarks

We will provide full insights in the results of all variants of the Time Dependent Pickup and Delivery Problem with Time Windows. This include insights in the performance of both methods, insights between the results of different variants of the problem (e.g. What is the effect of flexible departure times? What is the effect of having the possibility to reject customers?) and insights in the effect of taking time dependent travel times into account.
account. In Figure 1 a first impression on the value of having flexibility is provided, where *Fix* versus *Flex* refers to fixed versus flexible departure times, and *All* versus *Prof* refers to handling all versus only the profitable requests. Variant *Flex, Prof* is the most flexible variant. It can be seen that the more request, the lower the gains of this flexibility are.

![Figure 1: Value of flexibility](image)

References


Heuristic Solution of the Consistent Vehicle-Routing Problem

Dominik Goeke
School of Business and Economics, RWTH Aachen University

Roberto Roberti
Department of Information, Logistics and Innovation, Vrije Universiteit Amsterdam

Michael Schneider
School of Business and Economics, RWTH Aachen University
Email: goeke@dpo.rwth-aachen.de

1 Introduction

Vehicle-routing problems (VRPs) with consistency considerations have received substantial interest because of the practical importance of consistent service in many industries, like, e.g., small package shipping, health care, or vendor-managed inventory systems. To boost customer satisfaction, customers should be served at roughly the same time (arrival time consistency, ATC) by the same driver (driver consistency, DC), or at least a small set of familiar drivers, each time they require service. Taking the driver’s perspective, serving the same customers repeatedly makes the driver familiar with the geographic region and the characteristics of the customer, and thus more efficient in fulfilling his tasks.

The most prominent VRP with consistency considerations is the consistent VRP (Con-VRP), introduced by [1]. The ConVRP is a multi-day VRP requiring that, in addition to the traditional constraints on vehicle capacity and route duration, the same driver serves the same customers at approximately the same time on each day that these customers require service, given by a maximum allowed difference between the arrival times on the different days. Originally, the problem is motivated from the delivery and collection operations at United Parcel Services, where strong emphasis is put on customer and employee satisfaction. In the academic literature, the ConVRP has received adequate attention from the heuristic side [1, 4, 2, 3], but no exact approach has been proposed yet.

We present a large neighborhood search featuring suitable penalty mechanisms to deal with infeasible solutions and a repair procedure specifically designed to improve the ATC of solutions. The LNS is able to significantly improve the solution quality on benchmark instances from the literature compared to state-of-the-art heuristics.
2 Problem Definition

In the ConVRP, a set of customers require delivery of a single commodity over a set of days. A homogeneous fleet of vehicles based at a single depot is available to satisfy all customer requests. Capacity and maximum route duration of vehicles are restricted. The travel time between two locations is assumed to be deterministic and symmetric. Each customer is associated with service time and demand that depend on the day.

To respect DC, each customer must be served by the same vehicle on every day of the planning horizon on which service is required. ATC is expressed by requiring that the service must take place roughly at the same time, so the difference between the latest and the earliest arrival time at each customer over the planning horizon cannot exceed the maximum allowed time difference. As defined in [1], we assume that vehicles are not allowed to wait at a customer or the depot to meet ATC. The objective of the ConVRP is to find a set of routes for the vehicle fleet that minimizes the total vehicle operating time $z$ over the planning horizon.

3 The Proposed Large Neighborhood Search

LNS is a metaheuristic principle that aims at iteratively improving an initial solution by first removing a larger part of the solution (using a set of so-called removal operators) and then reinserting the removed solution components (using so-called insertion operators).

We propose a LNS for the ConVRP that is enhanced by several components: (i) suitable penalty mechanisms to deal with infeasible solutions, (ii) a repair procedure that is applied to improve the ATC, and (iii) regularly solving a set-partitioning problem using a cluster previously found by the search to improve the solution quality. To this end, we save all clusters, i.e., a subset of customers served by one vehicle and the associated routes, in a set $\Omega_{LNS}$.

An overview of the algorithm, which we call LNS with set partitioning (LNS-SP), is given in Figure 1. First, LNS-SP generates a feasible initial solution $\mathcal{S}_c$ with a multi-day savings algorithm that respects the consistency requirements of the ConVRP. Then, the initial solution is improved in $\eta_{total}$ iterations of LNS. Removal and insertion create a tentative solution $\mathcal{S}_t$ that respects DC. Subsequently, we try to improve the ATC of $\mathcal{S}_t$ with a two-stage procedure: First, we invert a subset of the routes to generate similar orders of the customer visits on all days. Then, we relocate customers in order to reduce ATC violations. The resulting solution $\mathcal{S}_t$ may be infeasible and is therefore evaluated with a generalized objective function that penalizes violation of capacity, maximum route duration and ATC. The decision whether to accept $\mathcal{S}_t$ or to keep the current solution $\mathcal{S}_c$ is based on a simulated annealing criterion. Finally, every 250 iterations without improvement of $\mathcal{S}_{best}$, we reset $\mathcal{S}_c$ to $\mathcal{S}_{best}$. 

496
\( \eta \leftarrow 1 \) \{set iteration counter\}
\( S_c \leftarrow \text{generateInitialSolution} \)

**while** \( \eta \leq \eta_{\text{total}} \) **do**

\{Large neighborhood search\}
\( \delta \leftarrow \text{drawNumberOfCustomersToRemove} \)
\( S_t \leftarrow \text{insertCustomers(\text{removeCustomers}(S_c, \delta))} \)
\( S_t \leftarrow \text{applyATCImprovement}(S_t) \)
updatePenalties(\( S_t \))

**if** acceptSA(\( S_c, S_t \)) **then**

\( S_c \leftarrow S_t \)

**end if**

**if** \( S_t \) improves \( S_{\text{best}} \) **then**

\( S_{\text{best}} \leftarrow S_t \)

**end if**

\{Set partitioning\}
\( \Omega^{\text{LNS}} \leftarrow \text{addClusters}(S_t) \)

**if** 5000 iterations have passed since last set partitioning **then**

\( S_{\text{best}} \leftarrow \text{solveSetPartitioning}(S_{\text{best}}, \Omega^{\text{LNS}}) \)

**end if**

**if** solution has not improved for 250 iterations **then**

\( S_c \leftarrow S_{\text{best}} \)

**end if**

\( \eta \leftarrow \eta + 1 \)

**end while**

**return** \( S_{\text{best}} \)

Figure 1: Overview of the LNS-SP algorithm.

## 4 Computational Results

We compare LNS-SP to the approaches from the literature; for the sake of conciseness, we limit the comparison to the two best-performing approaches, i.e., the template-based ALNS of [2] (denoted as KPH) and the LNS method of [3] (denoted as KGHP).

Table 1 shows the results on benchmark instances from the literature. Two different versions of LNS-SP are studied: LNS-SP–25k using a total of \( \eta_{\text{total}} = 25000 \) iterations, and LNS-SP–5k using a reduced number of iterations \( \eta_{\text{total}} = 5000 \). For each instance, we report the name and the previous best-known solution (BKS). For each solution method, we report the percentage gap of the best solution found in 10 runs to the BKS \( (\Delta z_b) \), the gap of the average solution value of the 10 runs to the BKS \( (\Delta z_a) \), and the average computation time in seconds \( (t) \). In addition, the best solution that we found during the overall testing of our method and its gap to the BKS are reported in columns LNS-SP. For each instance, the best solution found by any of the tested methods is marked in bold.
Table 1: Comparison of LNS-SP to the best-performing approaches from the literature.

<table>
<thead>
<tr>
<th>Inst.</th>
<th>BKS</th>
<th>KPH</th>
<th>KGHP</th>
<th>LNS-SP–25k</th>
<th>LNS-SP–5k</th>
<th>LNS-SP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$\Delta z_b$</td>
<td>$\Delta z_a$</td>
<td>$t$</td>
<td>$\Delta z_b$</td>
<td>$\Delta z_a$</td>
</tr>
<tr>
<td>1_50_0.7</td>
<td>2124.21</td>
<td>0.0</td>
<td>3.3</td>
<td>5.5</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>2_75_0.7</td>
<td>3540.80</td>
<td>1.7</td>
<td>1.8</td>
<td>14.7</td>
<td>0.0</td>
<td>1.4</td>
</tr>
<tr>
<td>3_100_0.7</td>
<td>3280.47</td>
<td>1.4</td>
<td>1.8</td>
<td>25.6</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td>4_149_0.7</td>
<td>4473.31</td>
<td>1.9</td>
<td>2.8</td>
<td>84.3</td>
<td>0.0</td>
<td>1.9</td>
</tr>
<tr>
<td>5_199_0.7</td>
<td>5632.22</td>
<td>0.6</td>
<td>0.9</td>
<td>122.2</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>6_49_0.7</td>
<td>4051.48</td>
<td>0.0</td>
<td>0.0</td>
<td>6.6</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>7_75_0.7</td>
<td>6673.61</td>
<td>1.5</td>
<td>2.0</td>
<td>18.3</td>
<td>0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>8_100_0.7</td>
<td>7126.29</td>
<td>0.0</td>
<td>0.9</td>
<td>32.2</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>9_150_0.7</td>
<td>10381.90</td>
<td>0.0</td>
<td>0.7</td>
<td>97.4</td>
<td>0.1</td>
<td>0.6</td>
</tr>
<tr>
<td>10_198_0.7</td>
<td>12955.10</td>
<td>1.1</td>
<td>2.2</td>
<td>146.3</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>11_119_0.7</td>
<td>4471.22</td>
<td>0.3</td>
<td>0.3</td>
<td>36.0</td>
<td>0.0</td>
<td>2.6</td>
</tr>
<tr>
<td>12_100_0.7</td>
<td>3497.93</td>
<td>0.0</td>
<td>0.0</td>
<td>25.6</td>
<td>0.7</td>
<td>2.5</td>
</tr>
<tr>
<td>Avg.</td>
<td>0.7</td>
<td>1.4</td>
<td>51.2</td>
<td>0.1</td>
<td>1.2</td>
<td>44.1</td>
</tr>
</tbody>
</table>

5 Conclusions

We present an LNS that is able to find high-quality solutions in short run-times. The method embeds (i) a suitable penalty mechanisms to deal with infeasible solutions, (ii) a repair procedure to improve the ATC, and (iii) the solution of a set-partitioning problem to enhance solution quality. The computational experiments show that our LNS is able to clearly improve the solution quality compared to previously published heuristics on benchmark instances from the literature.

References


498
The SDVRP with Time Windows and Customer Inconvenience Constraints

Michael Drexl
Faculty of Applied Natural Sciences and Industrial Engineering,
Deggendorf Institute of Technology, D-94469 Deggendorf, Germany

Stefan Irnich
Chair of Logistics Management, Gutenberg School of Management and Economics,
Johannes Gutenberg University, Jakob-Welder-Weg 9, D-55128 Mainz, Germany

Nicola Bianchessi
Chair of Logistics Management, Gutenberg School of Management and Economics,
Johannes Gutenberg University, Jakob-Welder-Weg 9, D-55128 Mainz, Germany
Email: nbianche@uni-mainz.de

1 Introduction

In classical routing problems concerning the delivery of goods, each customer is visited exactly once. By contrast, when allowing split deliveries, customers may be served by means of multiple visits. This potentially results in substantial savings in routing costs and fleet size, as in the Split Delivery Vehicle Routing Problem (SDVRP), the relaxation of the Vehicle Routing Problem in which split deliveries are possible (see [1] for the most recent survey on the topic). Allowing split deliveries is clearly beneficial to the transport company. On the customer side, though, several visits cause inconvenience, as at each visit, the customer has to interrupt his primary activities to handle the goods receipt.

In this work, with the aim to control the degree of inconvenience caused by split deliveries and to balance overall distribution costs and customer satisfaction, we introduce a generalization of the SDVRP with Time Windows (SDVRPTW) we named the SDVRP with Time Windows and Customer Inconvenience Constraints (SDVRPTW-IC). We embed into the SDVRPTW-IC two measures for limiting customer inconvenience: (i) Maximum number of visits: this is the obvious and most direct way to limit customer inconvenience. (ii) Temporal synchronization of deliveries: it is required that all deliveries to the same customer arrive within a pre-defined time span. While the first measure has been already considered in routing problems with split delivery, the second is new. Solving the SDVRPTW-IC enables, to some extent, win-win situations for transport companies
and their customers. To our knowledge, one of the most effective exact algorithms for the solution of the SDVRPTW is the branch-and-cut algorithm proposed in [2]. We extend this branch-and-cut algorithm to address the SDVRPTW-IC.

The contribution of the work is not only innovative from a methodological point of view. Even more important, we shed light on complex interdependencies between VRPTW, SDVRPTW, and SDVRPTW-IC special cases. On the basis of the experimental analysis carried out, we show that straightforward comparisons based on variable routing costs (the standard SDVRPTW objective, see [3]) are insufficient, i.e., more general objective functions are needed to analyze savings that result from split deliveries. Moreover, we draw useful insights for logistics managers who want to find good trade-offs between customer inconvenience and cost savings.

2 The SDVRPTW with Customer Inconvenience Constraints

The SDVRPTW-IC is the generalization of the SDVRPTW, defined in [3], taking into account upper bounds on the number of visits and synchronization constraints for split deliveries occurring to the same customer. More formally, the following parameters become part of the problem definition: (i) Maximum number of visits: \( n_{i}^{\text{max}} \) and \( n_{\text{max}} \) limit the number of visits to customer \( i \in N \) and the overall number of visits respectively. (ii) Temporal synchronization of deliveries: \( \Delta_{i} \) limits the length of the time interval in which all deliveries to \( i \in N \) must take place. Moreover, the impact of these customer inconvenience constraints on the following types of distribution costs is taken into account in the SDVRPTW-IC objective function: (i) Variable routing costs: These are given for each arc as in the SDVRPTW. (ii) Costs related to route durations: We denote by \( \gamma \) the time-to-cost ratio that, multiplied by the duration of a route, yields the duration-related costs. (iii) Fixed vehicle costs: The fixed costs for using a vehicle are denoted by \( C \).

3 A Branch-and-Cut Algorithm

In order to address the SDVRPTW-IC, we extend the branch-and-cut algorithm proposed in [2] for the SDVRPTW. The algorithm is based on a relaxed compact formulation for the problem. This means that some integer solutions to the formulation are infeasible for the SDVRPTW-IC. Valid inequalities are used in order to strengthen the relaxed compact formulation and possibly cut off solutions that are infeasible for the SDVRPTW-IC. However, even with the valid inequalities, integer solutions to the new compact formulation remain to be tested for feasibility. The positive arc flow values in any given integer solution to the relaxed formulation induce a subnetwork of the original instance. All time-window feasible routes on this subnetwork are enumerated. An extended set-covering problem is then solved in order to decide on the selection of routes, their schedules, the quantities
to deliver to the visited customers, and, hence, overall feasibility. All solutions proved infeasible are cut off from the feasible region of the relaxed problem.

4 Experimental Results

We found that the standard benchmark for SDVRPTW, which is based on the benchmark defined by Solomon (1987), see [4], lacks generality, because instances do not exhibit different demand distributions. Therefore, we created 560 new test instances again derived from the well-known VRPTW instances defined in [4]. The vehicle capacity $Q$ is always set to 100. Five scenarios are then considered w.r.t. the customer demands: $D_1: [10; 70]$, $D_2: [10; 50]$, $D_3: [30; 70]$, $D_4: [30; 50]$, $D_5: [50; 70]$. In each scenario $[a, b]$, the demand $d_i$ of customers $i \in N$ is drawn from a discrete uniform distribution in $[\frac{a}{100}Q, \frac{b}{100}Q]$.

We considered the eight distribution policies described in Table 1. The extreme policies are those leading to the VRPTW (no splitting at all) and the SDVRPTW (arbitrary splits allowed), while the introduction of the inconvenience measures creates variants of the SDVRPTW-IC. The VRPTW served as baseline against which the other distribution policies were compared. We consider synchronization and limiting the number of visits as alternative measures for controlling inconvenience and therefore analyzed them separately.

We performed three sets of experiments using different objectives, as defined in Table 2. An instance was used for the analyses of the results only when it had been solved to optimality for all policies and all objective functions. This was the case for 115 instances.

5 Conclusions

In this work, we investigated the possibilities and limitations of split deliveries with the aim of creating a win-win situation for carriers and customers in goods distribution systems. Based on the analyses of the results obtained, we can make the following final recommendations to logistics managers: (i) In general, split deliveries pay off; they should be considered independent of the objective. (ii) When variable routing costs and costs related to route durations are relevant (Objective II), split deliveries are less beneficial than for other objectives, but still an alternative worth considering. (iii) A limit on the number of visits to individual customers is not an effective measure to mitigate customer inconvenience resulting from split deliveries, as it hardly changes the number of visits w.r.t. the SDVRPTW, i.e., it does not improve the quality of service to the customers. (iv) According to the average percentage of split customers, a moderate limit on the total number of visits seems to be a valid measure to reduce customer inconvenience. (v) Nevertheless, the synchronization of visits allows in general to find better results. Visit synchronization, if properly implemented in practice, causes only very minor increases

501
in any of the three components of logistics costs (w.r.t. the SDVRPTW) and therefore appears to be the most sensible and useful distribution policy.

Table 1: The different distribution policies considered in the computational experiments

<table>
<thead>
<tr>
<th>Policy</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRPTW</td>
<td>Standard VRP with time windows.</td>
</tr>
<tr>
<td>SDVRPTW</td>
<td>Split delivery VRP with time windows.</td>
</tr>
<tr>
<td>SΔ, for Δ = 0</td>
<td>SDVRPTW with temporal synchronization of deliveries/visits. Δ = 0 means exact temporal synchronization.</td>
</tr>
<tr>
<td>NVν, for ν = 2</td>
<td>SDVRPTW with at most (n_i^{\text{max}}=\nu) visits for each customer (i \in N).</td>
</tr>
<tr>
<td>TNV(x), for (x = 25, 50, 75)</td>
<td>SDVRPTW with a limit on the total number of visits, (n^{\text{max}}). For an instance with (n) customers and (\xi) visits in the optimal SDVRPTW solution, (n^{\text{max}} = n + \lceil \frac{x}{100} \cdot (\xi - n) \rceil).</td>
</tr>
</tbody>
</table>

Table 2: The different objective functions used in the computational experiments

<table>
<thead>
<tr>
<th>Objective function components</th>
<th>Objective function</th>
<th>Variable routing costs</th>
<th>Costs related to route durations</th>
<th>Fixed vehicle costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>yes</td>
<td>yes: (\gamma = 1)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>yes</td>
<td>yes: (\gamma = 1)</td>
<td>yes: (C = 1,000,000)</td>
<td></td>
</tr>
</tbody>
</table>

References


A strongly polynomial Contraction-Expansion algorithm for network flow problems

Jean Bertrand Gauthier
Chair of Logistics Management,
Gutenberg School of Management and Economics, Mainz, Germany

Jacques Desrosiers
Département de sciences de la décision,
GERAD & HEC Montréal, Canada
Email: jacques.desrosiers@hec.ca

Marco E. Lübbecke
Chair of Operations Research,
RWTH Aachen University, Aachen, Germany

1 Introduction

Degeneracy can greatly affect the resolution process of the primal simplex algorithm (PS) going as far as jeopardizing the convergence of the latter. It is in fact the only phenomenon which questions the efficiency of this algorithm. The presence of degeneracy across a broad range of linear programs makes this phenomenon worth studying. This is especially true for network flow problems which are well known for very high level of degenerate pivots. It has drawn attention since the fifties alongside the birth of linear programming. As it stands today, the possibility of eradicating the drawbacks attributed to degeneracy is still an open question.

PS pursues optimality using pivoting operations until a sufficient stopping criterion is met. When facing degeneracy, this iterative process suffers from null step operations. The robustness of the pivot operation is compromised and, by extension, the efficacy of the resolution tool. It is not surprising that a plethora of pivot rules have fed the content of several papers [1, 2, 3]. Unfortunately, while practical concerns have sometimes been met, none of these rules have provided theoretical answers. The same is true for the work of [4] which introduces right-hand side perturbations to modify the way the polyhedron’s hyperplanes intersect.

All of these tricks are intended to be used on the global problem and are unable to tackle the problems unceasingly increasing in size. The more successful methods propose
to decompose the original problem in order to better guide the resolution. The Improved Primal Simplex algorithm (IPS) is one such method [5, 6, 7]. While the latter aims to capitalize on degeneracy, several theoretical questions remain unanswered and lay the work ahead.

2 Cycles, pricing, and pivots

Within the realm of linear programming, iterative algorithms that maintain feasibility throughout the solution process all identify a direction and then move along the latter with some nonnegative step-size. We call the oracle used to identify a direction the pricing problem, regardless of the algorithm version retained. Since this oracle maintains its form across the various algorithms, it is a common denominator whose canonical form is first observed in the minimum mean cycle-canceling algorithm (MMCC) [8, 9], the average cost of a cycle being taken over the number of arcs. In this respect, the network flow nomenclature is heavily borrowed thus contributing to the intuitive understanding of the pricing problem.

It is well known that all cycles necessary to reach an optimal minimum cost flow solution can be observed on the residual network [10]. That is, all directed cycles. Furthermore, each of these cycle can individually accommodate some strictly positive flow. In optimization terms, each of these cycles, or combination of, forms a direction. A degenerate pivot is therefore induced when the selected cycle does not actually exist on the residual network. [11] transfers the concept of paths and cycles along with some network flow properties to linear programs. Alternative necessary and sufficient optimality conditions expressed on the so-called residual problem are obtained in the process.

3 Properties

The easiest way to describe the Contraction-Expansion algorithm [12] is with respect to the residual network. In MMCC, every arc for which the current solution displays a flow that is not at its bound is doubled. Our proposal hides some of the arcs from the residual network. In doing so, independent trees are identified and replaced by single nodes in an alternative network. This mechanic is called contraction. The resulting so-called contracted network is much more dense than the original one, it contains less nodes and drastically less arcs. This loss of information does not compromise the existence of negative cycles nor their unit cost. It does however modify the average evaluation of this cost since some arcs are no longer accounted for. The order in which the negative cycles are selected is thus modified when one compares it with MMCC.

The tools provided by the complexity analysis of MMCC allow us to state certain properties regarding this contraction gymnastic. The most important behavior noted
with MMCC is that of phases as seen in the Cancel-and-Tighten solution strategy. In fact, the paper also introduces a so-called Type 3 cycle which insists on the measurable jump aspect we wish for between two successive phases. It can also be observed in this framework when using partial contraction to mimic the behavior of MMCC thus ensuring a strongly polynomial algorithm. This partial contraction is obtained by modifying the choice of the hidden arcs as the algorithm progresses. The selection is made in such a way that it actually corresponds to an expansion of the contracted network. Whether this expansion is required to achieve strongly polynomial properties is still an open question. Although, it is interesting to note that strongly polynomial time complexity is indeed verified without partial contraction for special applications, namely those in binary format (assignment, shortest path, maximum flow with unit capacities). The computational study compares the behavior of the Contraction-Expansion algorithm with those of MMCC and Cancel-and-Tighten.

References


The Directed Network Design Problem with Relays

Markus Leitner
Department of Statistics and Operations Research, University of Vienna, Austria

Ivana Ljubić
ESSEC Business School of Paris, Cergy-Pontoise, France
Email: ljubic@essec.edu

Martin Riedler
Institute of Computer Graphics and Algorithms, TU Wien, Vienna, Austria

Mario Ruthmair
Department of Statistics and Operations Research, University of Vienna, Austria

1 Introduction

The Directed Network Design Problem with Relays (DNDPR) is used to model many applications in the design of transportation and telecommunication networks. Besides the usual installation of links in the network, additional decisions have to be made on where to place the relay points so as to make sure that connections between the given origin-destination pairs can be established. More formally, the problem is defined as follows:

We are given a digraph $G = (V, A, c, w, d)$ with the set of nodes $V$, the set of arcs $A$, with relay costs $c : V \rightarrow \mathbb{Z}_{\geq 0}$, arc costs $w : A \rightarrow \mathbb{Z}_{\geq 0}$, and arc distances $d : A \rightarrow \mathbb{Z}_{\geq 0}$. Moreover, we are given a distance bound $\lambda_{\text{max}}$ and a set of origin-destination pairs $\mathcal{K}$ that we will refer to as commodities. For $(u, v) \in \mathcal{K}$ we call $u$ the source of the commodity and $v$ its target. The goal of the directed network design problem with relays (DNDPR) is to find a cheapest way to place relays on a subset of the nodes and to select a subset of the arcs s.t. in the resulting network, for each $(u, v) \in \mathcal{K}$ there exists a directed path from $u$ to $v$ not exceeding the limit $\lambda_{\text{max}}$ between $u$ and the first relay, any two consecutive relays, and the last relay and $v$.

There are two main variants of this problem that need to be distinguished. The first one requires that the commodity pairs are connected using simple paths only, whereas the other version also permits trails. In general, cycles allow cheaper solutions, see Figure 1 for an illustration.
Figure 1: An example of a symmetric instance with arc distances provided next to the arcs, and relay and edge costs given in parentheses. There is a single commodity $\mathcal{K} = \{(0, 3)\}$ and $\lambda_{\text{max}} = 4$. The figure in the middle shows the acyclic optimal solution, and the most right figure shows the non-restricted general solution. Installed relays are colored black.

1.1 Related Literature

The problem has been studied in [Li et al.(2012)] where the authors have proposed two mixed integer programming (MIP) formulations: the first one is the compact node-arc formulation, the second one is the arc-path formulation with an exponential number of variables. For the latter model, the authors have implemented an exact branch-and-price based solver. In their computational study, the authors have demonstrated that using the branch-and-price procedure, provably optimal solutions can be found for small and sparse instances with 5-10 commodities and up to 60 nodes, whereas larger or denser graphs remain out of reach for their solver.

The undirected version of the problem has been studied in the literature as well, see e.g., [Cabral et al.(2007), Kulturel-Konak and Konak(2008), Konak(2012)]. In this problem, known simply as the Network Design Problem with Relays (NDPR), cycles are allowed, whereas in the DNDPR literature they are prohibited. Clearly, this makes a significant difference between the modeling approaches used for the two problems, and in particular, most of the MIP formulations known for the NDPR cannot be straightforwardly translated into valid models for the DNDPR.

In this work, we will focus on the DNDPR and its acyclic variant.

2 Formulations on a Layered Graph

The models we use in our study exploit the so-called layered graphs. Layered graphs are a powerful tool to implicitly encode distances in their structure, so that only paths that are feasible w.r.t. the distance bound $d_{\text{max}}$ are allowed. Some recent examples where layered graphs have been successfully used to beat state-of-the-art exact approaches can be found in [Ljubić and Gollowitzer(2013), Gouveia et al.(2011), Ruthmair(2012), Gouveia et al.(2014)].
2.1 Layered Graphs

Given a digraph \( G = (V, A, d) \) with distances \( d: A \rightarrow \mathbb{Z}_{\geq 0} \) we obtain a layered digraph \( G_L = (V_L, A_L) \) as follows. We use sets \( V_l^0 \) containing all node copies on layers smaller than or equal to \( l \). We define these sets recursively:

\[
V_L^0 = \{v_0 \mid v \in V\}
\]

\[
V_L^l = \{v_l \mid u_m \in V_L^{l-1}, v \in \delta^+(u), m + d(u, v) = l\} \cup V_L^{l-1}
\]

Layer zero (also called base layer) contains all nodes \( v \in V \) from the original graph. Set \( V_L^l \) contains all nodes that can be reached with total distance at most \( l \) starting at vertices on lower layers. To avoid exceeding the distance bound only node copies on layers smaller than or equal to that bound are considered. Thus, we define \( V_L = V_L^{\lambda_{\text{max}}} \).

The arc set \( A_L \) connects all nodes that have been connected in the original graph. Additionally, we add arcs from all nodes \( v_l \in V_L \), \( l > 0 \) on higher layers to the base layer. Considering a path in the layered graph using such an arc corresponds to utilizing a relay. Thus, we call these arcs relay arc. Arc set \( A_L \) consists of the relay arcs \( A_L^r \) and the arcs \( A_L^a \) derived from the base graph:

\[
A_L^r = \{(i_l, i_0) \mid i_l \in V_L, l > 0\}
\]

\[
A_L^a = \{(i_l, j_m) \mid i_l \in V_L, j_m \in V_L, (i, j) \in A, d(i, j) = m - l\}
\]

\[
A_L = A_L^a \cup A_L^r
\]

Figure 2 shows the layered graph corresponding to the instance given in Figure 1. The relay arcs \( A_L^r \) are depicted in dashed lines and the remaining arcs in solid lines.

Figure 2: Layered graph \( G_L = (V_L, A_L) \) corresponding to the instance in Figure 1 for \( \lambda_{\text{max}} = 4 \)
<table>
<thead>
<tr>
<th>Instance</th>
<th>B&amp;P 2</th>
<th>Our B&amp;C on LG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Lin et al.)</td>
<td>asym.</td>
</tr>
<tr>
<td>04A05B70L05K</td>
<td>8.4</td>
<td>4.3</td>
</tr>
<tr>
<td>04A05B70L10K</td>
<td>55.9</td>
<td>32.1</td>
</tr>
<tr>
<td>05A05B70L05K</td>
<td>20.8</td>
<td>13.2</td>
</tr>
<tr>
<td>05A05B70L10K</td>
<td>107.3</td>
<td>71.2</td>
</tr>
<tr>
<td>06A05B70L05K</td>
<td>7.1</td>
<td>29.7</td>
</tr>
<tr>
<td>06A05B70L10K</td>
<td>61.1</td>
<td>193.9</td>
</tr>
<tr>
<td>07A05B70L05K</td>
<td>31.8</td>
<td>27.1</td>
</tr>
<tr>
<td>07A05B70L10K</td>
<td>34.6</td>
<td>258.2</td>
</tr>
<tr>
<td>08A05B70L05K</td>
<td>9.3</td>
<td>139.5</td>
</tr>
<tr>
<td>08A05B70L10K</td>
<td>92.0</td>
<td>523.4</td>
</tr>
<tr>
<td>09A05B70L05K</td>
<td>9.9</td>
<td>107.0</td>
</tr>
<tr>
<td>09A05B70L10K</td>
<td>40.5</td>
<td>760.0</td>
</tr>
<tr>
<td>10A05B70L05K</td>
<td>40.9</td>
<td>246.0</td>
</tr>
<tr>
<td>10A05B70L10K</td>
<td>33.6</td>
<td>778.7</td>
</tr>
<tr>
<td>11A05B70L05K</td>
<td>25.1</td>
<td>228.4</td>
</tr>
<tr>
<td>11A05B70L10K</td>
<td>45.4</td>
<td>1152.8</td>
</tr>
<tr>
<td>12A05B70L05K</td>
<td>5.2</td>
<td>442.9</td>
</tr>
<tr>
<td>12A05B70L10K</td>
<td>110.1</td>
<td>1294.6</td>
</tr>
</tbody>
</table>

Table 1: Speedup ratio with respect to the node-arc model from [Li et al.(2012)].

On such obtained layered graph, we study several MIP formulations, based on compact flow models, and a cut-set based one (exponential in size) for which we develop a branch-and-cut (B&C) procedure.

Due to the lack of space, the models are skipped in this report. We summarize the obtained computational results in Table 1, using the set of benchmark instances from the literature. Two approaches are compared: the previous state-of-the-art branch-and-price solver by [Li et al.(2012)] (denoted by B&P 2) and our new branch-and-cut approach on the layered graph (denoted by Our B&C on LG). To allow for a fair comparison, instead of comparing the running times on different machines, we report the speed-up ratios of both approaches with respect to the basic node-arc formulation given in [Li et al.(2012)]. As the obtained results indicate, in most of the cases our new B&C solver outperforms the B&P by [Li et al.(2012)] by orders of magnitude. Further details of our model and its properties will be shown at the conference.
References


Leveraging network structure in covering path problems: An application to school bus routing

Karen Smilowitz and Liwei Zeng
Department of Industrial Engineering and Management Sciences
Northwestern University, IL USA

Sunil Chopra
Kellogg School of Management
Northwestern University, IL USA

Email: ksmilowitz@northwestern.edu

1. Introduction

In the face of budget cuts, many public school districts in the United States are looking to reduce transportation expenditures. In this talk, we will provide a brief overview of a partnership with a public school district north of Chicago and present our work on the School Bus Routing Problem (SBRP). The SBRP has been studied by the operations research community for over fifty years, identifying creative routing and scheduling approaches for school districts (see review in [1]). The SBRP itself is a composite of five decision sub-problems: data preparation, bus stop selection, bus route generation, school bell time adjustment, and route scheduling. We take a new approach to the joint problem of bus stop selection and bus route generation, leveraging the underlying grid-like structure of the road network to obtain robust, easy-to-implement solutions.

In our first step towards a comprehensive solution approach, we introduce the covering path problem on a grid (CPPG) which finds a cost-minimizing path connecting a subset of points in a grid such that each point is within a predetermined distance of a point from the chosen subset. With a grid structure, we are able to develop efficient methods to find feasible, high quality solution paths for the CPPG. Our results for the stylized setting establish key building blocks for more general settings, including multiple, capacitated vehicles.

2. Preliminaries

Given a weighted graph $G = (V, E)$, a coverage region $D$, and a coverage radius $k > 0$, the Covering Path Problem (CPP) finds the least cost path connecting a subset $V_1 \subseteq V$ such that for
every point \( x \in D \), there exists \( v_1 \in V_1 \) such that the distance between \( x \) and \( v_1 \) is at most \( k \). We develop efficient solution methods for the CPP when the problem is restricted to a unit grid graph.

Motivated by the SBRP in which students are located along streets and bus stop locations are selected from intersections, we define the following problem setting. We use the \( l_1 \) norm to measure the distance between any two points. Given a positive integer pair \( (m, n) \), let \( D = \{(x, y) \mid 0 \leq x \leq n, 0 \leq y \leq m\} \) be a rectangular region in \( \mathbb{R}^2 \). \( D_{\text{int}} \) corresponds to the integer points in \( D \) and \( D_{\text{edge}} \) corresponds to points along the edges of the grid graph defined on \( D_{\text{int}} \). Point \( A \) is said to cover point \( B \) if and only if the distance between \( A \) and \( B \) is no more than \( k \).

**Definition 1 (CPP on grid (CPPG)).** Given a positive integer pair \( (m, n) \) and coverage radius \( k > 0 \), find a minimum cost path connecting a subset of integer points \( D_1 \subseteq D_{\text{int}} \) such that each point in \( D_{\text{edge}} \) is covered by at least one point in \( D_1 \).

We consider two costs: a fixed stop cost for each point in \( D_1 \), and a path cost, measured by the length of the path connecting the points in \( D_1 \). We focus on the following optimization version of the CPPG and a related decision version.

**Definition 2 (Optimization version of the CPPG).** Given cost function \( C(\cdot, \cdot) \), find a covering path that minimizes \( C(L, T) \), where \( L \) is the path length and \( T \) is the number of stops.

One can not minimize \( L \) and \( T \) simultaneously. To minimize stop count, the overlap in coverage region for stops is minimized; to minimize path length, traversals across the region are minimized which increases coverage overlap.

To solve the CPPG, we show that one can reduce the CPPG with any coverage radius \( k > 0 \) to one of two cases: \( k \) as integer or half-integer. When \( k \) is an integer, covering \( D_{\text{edge}} \) is equivalent to covering the rectangle \( D \) with coverage radius \( k \) and when \( k \) is a half-integer, covering \( D_{\text{edge}} \) is equivalent to covering integer points \( D_{\text{int}} \) with coverage radius \( k - \frac{1}{2} \). The transformation leads to two CPPG variations: the continuous CPPG in which we cover all points in the rectangular grid \( D \) and the discrete CPPG in which we cover all integer points in \( D_{\text{int}} \). The continuous CPPG falls into a stream of continuous facility location and routing models which have been shown to offer computational simplicity compared to their discrete counterparts; see recent work in [2]. We show that insights from the continuous CPPG can be used for the discrete CPPG. To develop these insights, we define a relaxation of the C-CPPG (called RC-CPPG) where the set of stops \( D_1 \) can be selected from all points in \( D \).

3. Main results

For each CPPG variant, we employ the following approach to obtain feasible paths and bound the performance of the feasible paths. We introduce a trade-off constraint which quantifies the trade-off between the path length \( L \) and number of stops \( T \) and provides lower bounds for the set of feasible solutions to the optimization problem. Our analysis of the boundary in the form of a trade-off between path length and stop count uses two functions based on \( L \) and \( T \):
• the average distance between consecutive stops on a path, \( d = \frac{L}{T-1} \).

• \( f(d) = d(2k - \frac{d}{2}) \) represents the maximum area of the region covered by a stop that is not covered by the previous stop.

The function \( f(\cdot) \) is an approximate measure of the unique coverage region for consecutive stops. To minimize \( T \), one strives to maximize this area. We show the geometric interpretation of \( f(\cdot) \) in Figure 1. The coverage region of a stop is a diamond with the \( l_1 \) norm. If two consecutive stops \( F_i \) and \( F_{i+1} \) are separated by distance \( d \), \( f(d) \) is an upperbound of the area of the region covered by \( F_{i+1} \) but not \( F_i \) (shaded region in Figure 1). If \( d > 2k \), then \( F_i \) and \( F_{i+1} \) serve disjoint regions and \( f(d) = 2k^2 \).

**Theorem 1 (Trade-off constraint for the RC-CPPG).**

If \((L, T)\) is a feasible pair for the RC-CPPG with \( T > 1 \),

\[
(T - 1)f\left(\frac{L}{T-1}\right) \geq N - 2k^2,
\]

(trade-off constraint)

where \( f(\cdot) \) is a function of the average distance between consecutive stops, \( d \), defined as:

\[
f(d) = \begin{cases} 
  d(2k - \frac{d}{2}) & \text{if } d \in (0, 2k] \\
  2k^2 & \text{if } d \in (2k, \infty),
\end{cases}
\]

(1)

and \( N = mn \) is the area of rectangle \( D \).

Moreover, when \( \frac{L}{T-1} \leq 2k \), the trade-off constraint is equivalent to

\[
2kL - \frac{L^2}{2(T-1)} \geq N - 2k^2.
\]

(2)

Theorem 1 provides a lower bound on the feasible region for \( L \) and \( T \) which we use to show the feasible paths are near-optimal. We define a group of feasible paths called “up-and-down paths”. We use the term “traversal” to represent the vertical line connecting points \((s, 0)\) and \((s, m)\). An up-and-down path connects a set of traversals and the separation of subsequent traversals is a function of \( d \) as defined in Definition 3.

**Definition 3 (Type-\(d\) up-and-down path).** For \( d \in (0, 2k] \), define \( r = 2k - \frac{d}{2} \). In the RC-CPPG, a type-\(d\) up-and-down path connects the following points sequentially (as shown in Figure 2):

\[
(0, 0) \rightarrow (0, m) \rightarrow (r, m) \rightarrow (r, 0) \rightarrow (2r, 0) \rightarrow (2r, m) \rightarrow (3r, m) \rightarrow (3r, 0) \rightarrow \cdots .
\]
For an odd traversal connecting \((2ir, 0)\) to \((2ir, m)\) for \(i = 0, 1, \cdots (i \leq \frac{n}{2r})\), we establish stops at \((2ir, jd)\) for \(j = 0, 1, \cdots \) for \(jd \leq m\). For an even traversal connecting \(((2i + 1)r, m)\) to \(((2i + 1)r, 0)\) for \(i = 0, 1, \cdots (i \leq \frac{n}{2r})\), we establish stops at \(((2i + 1)r, jd + \frac{d}{2})\) for \(j = 0, 1, \cdots \) for \(jd + \frac{d}{2} \leq m\). Finally, we establish a stop at point \((\cdot, m)\) for each traversal to ensure coverage.

The optimal solution to the RC-CPPG can be found quickly through a simple algorithm based on a convex relaxation of the optimization problem. This approach is extended in our work to solve the C-CPPG and D-CPPG, thus fully solving the CPPG.

4. Conclusion

We study the covering path problem on a grid with cost for stops and travel along the path. We show that the CPPG can be solved with either C-CPPG or D-CPPG depending upon the coverage radius. For each setting, we provide a feasible path following an up-and-down pattern and determine the parameters of the path by solving a convex relaxation of the original optimization problem. The objective value of the constructive solution can be bounded relative to the optimal value. These results in a stylized grid setting are important building blocks for solutions on more general grids. The trade-off constraints for different CPPG variants are valid for general grid graphs since they do not rely on the structure of a complete rectangular grid. As long as the general grid can be constructed as the union of a few rectangular grids, our constructive approach can be used to find a solution that is unlikely to be too far from optimal. In ongoing work, we are exploring ways to generalize the insights and results of this paper to address the many additional complications of school bus routing.

References


A Memetic Algorithm for Optimal Charging and Repositioning of Electric Vehicles in a Free-Floating Carsharing System

Henrik Andersson, Carl Axel Folkestad
Nora Hansen, Kjetil Fagerholt

Department of Industrial Economics and Technology Management,
Norwegian University of Science and Technology, Trondheim, Norway

Giovanni Pantuso
Department of Electronics, Information, and Bioengineering,
Polytechnic of Milan, Milan, Italy
Email: henrik.andersson@ntnu.no

1 Introduction

Carsharing is generally defined as short-term vehicle access among a group of members who share the use of a vehicle fleet that is owned, maintained, managed, and insured by a Carsharing Organization (CSO). Carsharing services can be divided into two categories; free-floating systems and station-based systems. Free-floating systems enable users to pick up available cars and deliver them anywhere within a specified business area. In a station-based system, the cars are allocated to dedicated stations. A station-based system is either a two-way or a one-way system. In a two-way system, the user must pick up and return the car at the same station, while in a one-way system the user can pick up and return the cars at different stations.

Among the most important challenges in electric vehicle free-floating systems are recharging and repositioning of rental cars. When the battery level falls below a given threshold, a vehicle must be charged before it can be made available for customers. Furthermore, demand imbalances for vehicles in the business area may result in rental cars accumulating in certain areas. Therefore, repositioning is done to redistribute rental cars to other areas to improve the operator’s ability to meet customer demand. These problems are typically solved by having dedicated staff moving the cars from their current position to charging stations or to other areas.
2 Problem Description

We define the Free-Floating Electric Carsharing Charging and Repositioning Problem (FFECCRP) as the problem of repositioning electric rental cars in a free-floating carsharing system to charging stations when their battery level falls below a predefined threshold. When a rental car needs charging, an operator handles the rental car from its original position to a charging station. The duration each rental car can travel is limited based on its current battery level. The location to charge a car is chosen taking both the range of the car and the current distribution of cars in the business area into account. Each charging station has a given number of available charging slots and must have free capacity if a car is repositioned to that station.

The operators are transported to rental cars by service vehicles. Once an operator has repositioned a car to a charging station, a service vehicle will pick him/her up and transport him/her to his/her next rental car. Operators are not necessarily picked up by the same service vehicle that dropped them off, but service vehicles are the only available means of transport. Each service vehicle has a fixed capacity to carry operators and a given number of operators and service vehicles are available at the depot.

To quantify the distribution of rental cars in the system, the concept of states is introduced for each charging station and its surrounding area. The initial state describes the number of rental cars available for customers at the charging station when the planning period starts, i.e. all rental cars at the charging station or in the surrounding area with a battery level above a given threshold. The ideal state, assumed known from demand analyses, is the ideal number of rental cars at the charging station with a battery level above the threshold. After solving the problem the final state is reached. The final state equals the initial state at a charging station plus the number of rental cars handled to the station. As rental cars moved to a charging station become available for customers after a given time, they are counted in the final state.

The costs a CSO incur are due to the handling of rental cars, the postponement of handling, and imbalances in the distribution of cars. Particularly, the cost of handling cars is the cost of transporting operators with service vehicles to the rental cars in need of charging, and the cost of transporting operators from charging stations back to the depot. If handling is postponed the rental car is unavailable for customers until the next planning period. Hence, a penalty cost incurs. The CSO incurs deviation costs when there is a deviation between the ideal and final state at a charging station at the end of the planning period. Finally, a fixed cost is added for each service vehicle and operator used to handle the fleet of rental cars.

The objective is to minimize the costs of handling, repositioning, postponement, and deviations. Central to the problem is the trade off between the cost of not meeting demand due to a disadvantageous distribution of cars in the system and the cost of repositioning.
3 Hybrid Genetic Search with Adaptive Diversity Control

The implementation of the heuristic draws on the Hybrid Genetic Search with Adaptive Diversity Control (HGSADC), but it has been modified and extended significantly to fit the FF ECCRP. The algorithm evolves a population of individuals, where an individual represents a solution to the FF ECCRP. Solutions are allowed to be infeasible with respect to the maximum duration and the number of service vehicles used.

3.1 Individual Representation

An individual describes the routes of all operators and service vehicles. The operator routes include assignment of operators to handle each rental car, postponement of handling or assignment of rental cars to charging stations, and the handling order of each operator. The routes of the service vehicles include assignment of transport requests by operators and the visit sequence of each service vehicle.

Each individual is represented by five chromosomes. The first chromosome determines the charging station to move a rental car to, or if the handling of the car is postponed. The second chromosome defines for each rental car which operator that is going to perform the handling. The third chromosome defines for each operator the order to handle the rental cars assigned to the operator. Taking the first three chromosomes as given, transport requests for the operators that need to be taken care of by the service vehicles are formulated. The transport requests are used to define the fourth chromosome which assigns each transport request to a service vehicle. Finally, the last chromosome describes the route of each service vehicle.

3.2 The Algorithm

One of the main ideas behind the HGSADC is the following: if the rental car destination, the operator, and the handling sequence are given, the remaining problem, i.e. to determine the transport request assignment and the service vehicle routes, is similar to a dial-a-ride problem (DARP). The first three chromosomes determine all rental cars and charging stations each operator has to visit, including the visit order, and can therefore be used to formulate transport requests. Each transport request is associated with the operator requesting transport and by specifying time windows for the formulated transport requests, solution methods used for DARP can be used to construct the transportation request assignment chromosome and the route chromosome.

An initial solution is created by a four-step construction procedure. In the first step, each rental car is assigned a destination or the handling of it is postponed. Step two assigns an operator to handle each rental car. The third step of the algorithm sets the handling order of the cars assigned to each operator. In the last step, transport requests
are defined and given time windows, and a sweep heuristic for the DARP is used to create routes for all service vehicles.

The offspring generation scheme of the HGSADC selects two parent individuals using a binary tournament. A four-stepped crossover operator is then used. In the first step, the genes to inherit from each parent are decided. Step two inherits data from the first parent. In step three, data is inherited from the second parent. Finally, in step four, transport requests and service vehicle routes are constructed using the last step from the construction procedure. Due to the design of the crossover operator the offspring is feasible except in the time constraints and number of service vehicles used.

The education phase improves the handling sequence, transport request assignment, and routes. As different rental car destination and operator chromosomes are evaluated as a part of the overall HGSADC, these are not altered in the education module. Simple improvement operators are used in order to run a large number of improvement iterations with little computational effort. The education module also includes a repair procedure to make infeasible individuals feasible.

Three population management schemes are employed to improve the performance of the genetic search algorithm. Survivor selection is performed to increase the quality of the population by removing the worst quality individuals based on the biased fitness.

Penalty parameter adjustment updates the penalty parameters occasionally with the goal of attaining a given target ratio of feasible individuals. If the proportion of feasible individuals is below a threshold less than the target ratio, the penalty parameter is adjusted up. Similarly, if the target ratio is above the threshold more than the target ratio the penalty parameter is adjusted down.

Diversification is executed to prevent the algorithm from converging to a local optima.

4 Results

The HGSADC is capable of solving instances with up to 200 rental cars in need of handling yielding seemingly high quality solutions in an average computational time of less than 2400 seconds. When comparing solutions from the HGSADC with solutions produced by the algorithm when repositioning not is considered, the number of postponed rental cars and the number of deviations decrease by 9.4 percent and 7.1 percent, respectively. A reduction in postponed cars implies that more rental cars are handled when repositioning is considered due to the added positive effect of handling. The reduction in deviations imply that the rental cars handled are handled to more favourable destinations considering the distribution of rental cars in the system when repositioning is considered. Hence, we conclude that combining handling with repositioning is beneficial for the CSO.
Branch-and-price for charging station placement in free-floating electric car sharing systems

Georg Brandstätter
Department of Statistics and Operations Research,
University of Vienna, Vienna, Austria
Email: georg.brandstaetter@univie.ac.at

Markus Leitner
Department of Statistics and Operations Research,
University of Vienna, Vienna, Austria

Mario Ruthmair
Department of Statistics and Operations Research,
University of Vienna, Vienna, Austria

1 Introduction

Free-floating urban car sharing is a new mode of transportation that offers its customers much the same flexibility as owning a car without the associated ownership costs. Using electric vehicles in these systems offers both economic and ecological benefits, since (a) they operate very efficiently in urban settings, (b) they do not produce any local tailpipe emissions, and (c) they can be recharged from renewable sources, which reduces the system’s overall environmental impact. Their limited range is less of a concern in these settings, as trips are rather short.

However, electric cars must still be recharged regularly, which takes non-negligible time. A network of charging stations at which cars can be recharged between trips must therefore be built throughout the system’s operational area. As such stations are expensive to build and maintain, placing them effectively is paramount to the system’s economic viability.

So far, this station placement problem has only been considered for station-based systems where cars must return to a station at the end of each trip (see, e.g., [1, 2]). Since free-floating systems offer both increased flexibility to the users and potentially improved demand coverage to the operator, we want to investigate this problem in their context.
2 Problem definition

The free-floating charging station location problem (FF-CSLP) extends the charging station location problem (CSLP) \cite{2} insofar as cars no longer need to return to a station after each trip. While cars can now be parked anywhere, we assume that customers will only park either at their desired destination or at a nearby charging station.

The street network in the operational area is represented by an undirected graph $G = (\mathcal{V}, \mathcal{E})$ where each edge $e \in \mathcal{E}$ has an associated length $\ell_e \geq 0$. Some of its vertices $S \subseteq \mathcal{V}$ are potential charging station locations, each with associated opening cost $F_i \geq 0$, per-charger cost $Q_i \geq 0$ and maximum charger capacity $C_i \in \mathbb{N}$.

Expected customer demand is represented by a set of trips $K$, where each trip $k$ has associated origin and destination $o_k, d_k \in \mathcal{V}$, start and end time $s_k, e_k \in T = \{0, \ldots, T_{\text{max}}\}$, battery consumption $b_k$ and expected profit $p_k$. From a trip’s origin and destination, we can find its start locations $N(o_k)$ and end locations $N(d_k)$ as follows: a trip can start at any station within walking distance of its origin, as well as at the destination of any completed trip that is similarly close-by. Likewise, it can end at its own destination $d_k$ or at any station that is within walking distance of it.

Up to $H \in \mathbb{N}$ electric cars can be purchased, each with the same acquisition cost $F_c \geq 0$, battery capacity $B_{\text{max}} \geq 0$ and recharge rate $\rho \geq 0$.

Our objective is to choose which stations to open, how many chargers to build at each of them, and how many cars to buy in order to maximize the profit of those trip requests that can then be accepted, while staying within the available budget $W$.

A solution for the FF-CSLP therefore consists of a selection of stations from $S$ that are opened, along with the number of chargers built at each of them, and a selection of up to $H$ car routes from the set of all feasible routes $\Omega$. One such route consists of a sequence of temporally non-overlapping trips in order of their start time where each trip is assigned a start and end location from $N(o_k)$ and $N(d_k)$, respectively. It is feasible if (a) the assigned start station of each trip is the same as the previous trip’s assigned end station, and (b) the car’s battery level never goes below zero.

If a customer returns a car with low battery (i.e., below $\beta_0$) to a station instead of parking at the destination, he receives a monetary incentive. We model this with a reduced profit $p_k' < p_k$ that we collect instead of the full one.

3 Integer linear programming formulation

We use binary variables $y_i$ to denote whether station $i \in S$ has been opened and integer variables $z_i \in \{0, \ldots, C_i\}$ to denote the number of chargers built at each of them. Binary variables $x_k$ determine whether trip request $k \in K$ is accepted or not.

Binary variables $\mu_\omega$ indicate whether route $\omega \in \Omega$ is performed by one of the purchased
cars. For each route $\omega$, coefficient $\xi_{\omega}^i t$ is one if that route parks at location $i$ at time $t$ (either by arriving there at the end of a trip, or by remaining parked there from the previous time period $t - 1$), and zero otherwise. Similarly, coefficient $\chi_{\omega}^k$ is one if the route contains trip $k$ and zero otherwise. The route’s total profit $p_{\omega}$ is simply the sum of all its contained trip’s profits (full or reduced, depending on battery state and end location).

The FF-CSLP can now be formulated as the following integer linear program. We denote a constraint’s corresponding dual variable in parentheses where it is relevant to the pricing problem.

\[
\max \sum_{\omega \in \Omega} p_{\omega} \mu_{\omega} \quad (1)
\]
\[
\text{s.t. } \sum_{i \in S} (F_i y_i + Q_i z_i) + F_c \sum_{\omega \in \Omega} \mu_{\omega} \leq W \quad (\psi)
\]
\[
\sum_{\omega \in \Omega} \mu_{\omega} \leq H \quad (\vartheta)
\]
\[
z_i \leq C_i y_i \quad \forall i \in S \quad (4)
\]
\[
\sum_{\omega \in \Omega} \xi_{\omega}^i t \mu_{\omega} \leq z_i \quad (\pi_{it}) \quad \forall i \in S, t \in T \quad (5)
\]
\[
\sum_{\omega \in \Omega} \chi_{\omega}^k \mu_{\omega} = x_k \quad (\nu_k) \quad \forall k \in K \quad (6)
\]

The objective function (1) maximizes the profit of the selected routes. The budget constraint (2) limits expenditures for opening stations, building chargers and purchasing vehicles to the available budget $W$. Constraint (3) further guarantees that no more vehicles can be purchased than are available.

Inequalities (4) ensure that no chargers are built at closed stations. Constraints (5) then ensure that the number of cars parked at each station never exceeds the number of chargers built at it. Finally, set packing constraints (6) require that for each selected trip, one route containing it is also selected.

Due to the large number of variables $\mu_{\omega}$, we solve this model with branch-and-price.

**Pricing** In order to identify new routes that should be added to the model, we search for resource-constrained shortest paths (RCSP) in a directed acyclic time-space network that we call the location graph $G = (V, A)$. These paths represent the movement of a car between locations (stations and trip destinations) throughout the planning period $T$ and can be found with a dynamic programming labeling algorithm.

**Branching** We first branch on any fractional variables $y_i$, $x_k$ and $z_i$. If any fractional $\mu_{\omega}$ remain, we branch on pairs of trips $(k_1, k_2)$: one branch requires the selection of a route that contains both trips, while the other forbids the selection of any such route (cf. Ryan and Foster [3]). In case some $\mu_{\omega}$ variables are still fractional, we branch on their start and end locations, either requiring or forbidding a specific combination of them.
3.1 User behavior modeling

The aforementioned model can sometimes find overly optimistic solutions, since it gives the operator a lot of freedom regarding which trip requests to accept or decline and which start and end location to assign to each accepted trip. In practice, such decisions would be made by the system’s users, with little chance for the operator to influence their behavior.

Adding additional constraints to the model allows us to consider these behavioral aspects, ensuring that (a) trip requests cannot be declined if they would be feasible, (b) customers always choose the closest sufficiently charged car as their start location, and (c) they always return a car as close to their destination as possible.

4 Computational results

We evaluated the algorithm on a set of benchmark instances based on real-world data from Vienna. Potential stations were placed at supermarket parking spaces and near subway stations, while taxi trip data served as an estimation of car sharing demand.

Preliminary results show that the profit attainable in free-floating systems can be almost twice that of station-based systems, though computational effort often increases. Incorporating user behavior mitigates this somewhat and also reduces attainable profit.

Acknowledgements

This work is supported by the Joint Programme Initiative Urban Europe under the grant 847350 and by the Vienna Science and Technology Fund (WWTF) through project ICT15-014. Georg Brandstätter is additionally supported by a Dissertation Completion Fellowship of the University of Vienna.

References


Considering Spatial and Temporal Flexibility in Optimizing One-Way Electric Carsharing Systems

Burak Boyaci*, Konstantinos G. Zografos
Centre for Transport and Logistics,
Lancaster University Management School, United Kingdom
b.boyaci@lancaster.ac.uk, k.zografos@lancaster.ac.uk

1 Introduction

Carsharing is a shared-use vehicle model that allows users to rent cars for short periods of times. There are different types of systems according to their operational properties. In round-trip carsharing systems users are expected to return vehicles to their pickup locations. One-way systems relax this restriction and allows users to return cars to different drop-off locations. In station-based systems, there are designated parking locations to which vehicles should be returned. Free-floating systems relax this restriction and allow users to park vehicles to any legal parking locations within a designated area [1].

In this research we are dealing with operational planning decisions in station-based one-way electric carsharing systems with dynamic relocations. Different than the previous work in literature [2, 3, 4, 5, 6], we introduce spatial and temporal flexibility to the system by considering multiple pick-up and drop-off times and locations at different prices to increase total profit of the system. In other words, we will assume that the operator could offer to users discounted prices for pick-up time, and pick-up and drop-off locations that differ from the originally requested. The rational of the proposed approach is to accommodate to the maximum possible extend the requests of the users while at the same time ensures the maximization of the total profit resulting from the operation of the system.

2 Modelling Framework

The modelling framework used in this work includes modules for optimization and simulation. Optimization module introduces 3 separate mathematical models: (i) The station clustering, (ii) the operations optimization and (iii) the personnel flow models. The flow chart used for the solution framework can be seen in Figure 1. The proposed framework

*corresponding author
is an extension of the framework presented in [2]. The added value in this new framework is the proposed clustering model and introduction of the spatial and temporal flexibility.

The proposed operations optimization model is an advanced network flow model. Each node represents a unique station-time interval pair. There are separate networks for all vehicles and different personnel shifts. Every rental request and relocation are represented with flows on one or multiple networks. For instance, a rental is a flow on the network of vehicles; a relocation with a vehicle is a flow in the networks of both vehicles and related personnel shift; a relocation without a vehicle is a flow in the network of related personnel shift only. Every option of the rental request is generated as a potential flow in the network of vehicles and at most one of these options is allowed to be realised [2].

The designated system handles relocations throughout the day if there are available personnel. This means that, there can be as many relocations as the number of time intervals times the number of stations squared. To decrease this number, we introduced virtual nodes to the system and re-route relocation arcs through these nodes. The stations are clustered and assumed that the relocations are happening in three virtual steps: From (i) origin node to its cluster, (ii) between origin and destination nodes’ clusters and (iii) from destination node’s cluster to destination node. With this change, the number of variables and consequently the solution time are decreased dramatically. However, this simplification also introduces an error in the determining relocations. The cluster structures affects the magnitude of the error. To keep the error emerging from the restructuring of the network low, we develop and solve a mathematical model. This model takes travel times between stations as input and returns the set of clusters that minimises the error.

The station clustering model is used to create sets of stations for relocation flows. The mathematical model not only decides on the clusters of the stations, but also calculates the travel time between clusters and from/to stations to/from their clusters. Unfortunately, using clustering in the operations optimization model slightly overestimates the relocation durations. An objective function which minimises the weighted sum of the time difference between real and clustered relocation durations is used to minimise this effect. Since the relocation with and without vehicle speeds are assumed to be different, the model is executed twice at the very beginning of the solution process to create separate clusters of stations for both relocations with and without vehicles.

The relocations produced by the operations optimization model is an input to the
personnel flow model. Since the clusters are introduced, the solution of the operations optimization model represents relocations between clusters. The personnel flow model takes these output and creates feasible schedules for every relocation personnel. These individual relocation schedules are passed to the simulator for a feasibility check with the other parameters from the operations optimization model.

In the operations optimization model, vehicles’ movements are modelled as unit flows in a time-space network in order to keep model simple and efficient. However, this approach also prevents to track charging levels of the vehicles explicitly. The model regards every vehicle as identical flows. However, we can still check if the solution gives a schedule in which every vehicle has feasible charging levels or not. Furthermore, if the suggested schedule is infeasible, we can add more constraints to the model which forces these vehicles which were rented with infeasible charging levels to stay at their stations and to be charged. To detect infeasible charging levels of the vehicles, we developed a discrete event simulator which replicates an electric carsharing system. This simulator is used with the operations optimization and personnel flow models iteratively in the solution process. The output of these two models are tested in the simulator. If there are infeasibilities in the charging levels of the vehicles, the simulator informs the operations optimization model. Then the existing model is updated with additional constraints. This loop continues until the simulator does not detect infeasibilities of the car charging levels.

3 Preliminary Results and Conclusions

In the experiments, we used the locations and demand from the electric carsharing system operating in France [2]. In this system, we assumed that there are 3 parking places and one fully-charged available car at each station at the beginning of the day. Time intervals are set to 15min. There are four 4-hour working shifts to cover the entire 16 hours (6am-10pm) of operations. Three levels of demand (300, 400 and 500 rentals/day) are tested with 5 different levels of temporal flexibility in pick-up and drop-off times (0, ±15, ±30, ±45 and ±60min) and 3 different levels of spatial flexibility (0, 0.25 and 0.5km). Temporal flexibility is the maximum time difference between the assigned and requested pick-up (and drop-off) time. Spatial flexibility is the maximum accessing (and egressing) distance difference between the assigned and requested origin (and destination) stations. We discounted the price of the trip if the user is not served with what he requested at first place. In other words, the operator earns less when it cannot serve what the users exactly requested. The entire model including the simulator and mathematical models were implemented in C# .NET environment. IBM ILOG Cplex Version 12.71 with Concert Technology was used for solving the mathematical models. Each setting is run with 10 different demand sets. We assumed each rental cost €18 per hour. Customers also receive €8 per km for the
distance between their requested and offered stations during their pick-up and drop-off. Similarly, customers receive €9 per hour discount for early or late pick-up and drop-off.

Preliminary experimental results showed that both the spatial and temporal flexibility increases the profit of the operator. The results can be seen in Table 1. Bold figure in the first column and first row of each demand level of the table shows the average profit of the operator for the case without flexibility. The rest of the cells show the increase in profit from the base cases. As expected the profit of the operator increases with increased flexibility. However the highest marginal profit increase if observed when the temporal flexibility is 15min and the spatial flexibility is increased from 0.25km to 0.5km. The effect of temporal flexibility in particular is quite interesting. Implementing even a 15min flexibility to pick-up and drop-off times makes the operator profit over 4%. On the other hand, gain from spatial flexibility is also promising. The increase in the accessibility distance gives the operator to utilise its stations better. When the accessibility distance is increased to 0.5km, the number of stations that can be used for a pick-up (or drop-off) becomes 1.85 on average. This value is just 1.05 when the distance is 250 metres. Interestingly, the number of stations that can be used for pick-up when the accessibility distances are 0.75 and 1 km are 3.72 and 5.87 on average. Although we have excluded these cases in our preliminary analyses (since customers are expected to not willing to walk more than 500 metres), we will include in our future experiments to see how much the operator gains if they are implemented.

As another direction in our project, we want to check 3 different run types in details. In the run type we observed here, the operator receives all requests and decides which should be served. In the second run type, the operator evaluates the requests one at a time as they arrive chronologically in the system and decides to serve or not to serve the demand without knowing the future demand. The third run type is very similar to the second one. In the third run type, the operator not only decides if a request will be served or not but also informs the users about the pick-up and drop-off times as well as the origin and

<table>
<thead>
<tr>
<th># of requests</th>
<th>spatial flexibility (km)</th>
<th>temporal flexibility (minutes)</th>
<th>0</th>
<th>15</th>
<th>30</th>
<th>45</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0</td>
<td>3117</td>
<td>4.21%</td>
<td>4.43%</td>
<td>4.56%</td>
<td>4.56%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td></td>
<td>0.35%</td>
<td>4.71%</td>
<td>4.79%</td>
<td>4.87%</td>
<td>4.87%</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td></td>
<td>6.35%</td>
<td>8.62%</td>
<td>8.79%</td>
<td>8.79%</td>
<td>8.79%</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
<td>4002</td>
<td>4.60%</td>
<td>4.93%</td>
<td>4.93%</td>
<td>4.93%</td>
<td>4.93%</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td></td>
<td>0.34%</td>
<td>4.8%</td>
<td>5.27%</td>
<td>5.34%</td>
<td>5.34%</td>
</tr>
<tr>
<td></td>
<td>0.50</td>
<td></td>
<td>5.66%</td>
<td>10.18%</td>
<td>10.25%</td>
<td>10.33%</td>
<td>10.33%</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>4804</td>
<td>4.16%</td>
<td>4.81%</td>
<td>4.81%</td>
<td>4.81%</td>
<td>4.81%</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td></td>
<td>0.46%</td>
<td>4.78%</td>
<td>4.95%</td>
<td>4.99%</td>
<td>5.03%</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td></td>
<td>5.65%</td>
<td>10.02%</td>
<td>10.07%</td>
<td>10.12%</td>
<td>10.56%</td>
</tr>
</tbody>
</table>

Table 1: Preliminary experimental results
destination stations for their accepted requests. Although the first option could give more profitable schedules for the operator, users would prefer the third option more since the full information about their requests are received immediately. With extensive analyses, we want to see if different run types create substantial differences in profits or not.

References


Dynamic travelling salesman problem with uncertain release dates

Claudia Archetti  
Department of Economics and Management,  
University of Brescia, Brescia, Italy

Dominique Feillet  
Ecole des Mines de Saint-Etienne and LIMOS UMR CNRS 6158,  
CMP Georges Charpak, Gardanne

Andrea Mor  
Department of Economics and Management,  
University of Brescia, Brescia, Italy

Email: andrea.mor@unibs.it

M. Grazia Speranza  
Department of Economics and Management,  
University of Brescia, Brescia, Italy

1 Introduction

A common trait of the classical Vehicle Routing Problems (VRP) is the assumption that the goods to be distributed are available at the depot when the distribution starts. To consider a more dynamic organization of the distribution, [3] introduces the arrival time of the goods at the depot as part of the optimization problem. This poses the additional problem of whether it is better to wait for additional goods to arrive and have a better loaded vehicle, or to start a route of the vehicle with the currently available goods. The arrival time at the depot of the goods to be delivered to a customer is called its release date. The study of this problem is motivated by the growing interest in topics such as city logistics, where goods are consolidated at a distribution center and distributed to customers typically with smaller and lighter vehicles. In this case, goods are delivered to the distribution center throughout the day, when the final distribution to customers is already taking place. Another application arises in same day delivery problems related to e-commerce logistics. The TSP-rd is introduced in [2], where the complexity of the problem is analyzed for special topologies of the graph representing the distribution network. In [1] a heuristic approach based on a large neighborhood search and a local search is presented
for the variant of the problem where the minimization of the sum of the traveling time and waiting time at the depot is sought. In this work we consider an extension of the problem where future release dates are defined as uncertain and allowed to vary over time. Various reoptimization policies and solution strategies are devised for the problem and compared.

2 Definition and notation

The dynamic vehicle routing problem with uncertain release dates (DTSP-urd) is defined as follows. Let $G = (V, A)$ be a complete graph. A traveling time is associated with each arc $(i, j) \in A$, denoted by $d_{ij}$. It is assumed that the triangle inequality is satisfied. The set of vertices $V$ is composed by vertex 0, which identifies the depot, and the set $N$ of customers, with $|N| = n$. Each customer is characterized by the arrival time of its parcel at the depot. We call this value its release date. Based on the type of information available on the vehicle delivering the parcel to the depot we define three types of customers. Customers of the first type are those whose release date is known, as the vehicle has already arrived at the depot. We call $C \subseteq N$ the set of such customers. If the truck has not yet arrived, the release date is uncertain. The information about the uncertain release date can either be static or updated over time. The former is the case of non GPS-equipped vehicles, where no new information is available after the vehicle is off to its route until it reaches the depot, i.e., the release date becomes certain. We call $S \subseteq N$ the set of such customers. The latter is the case of GPS-equipped trucks where the information regarding the arrival time is updated over time. We call $D \subseteq N$ the set of such customers. The vehicles delivering to the depot are not affected by the arrival of other vehicles, i.e., the release dates are assumed to be independent. A single vehicle is allowed to perform a sequence of routes during the time horizon to deliver the goods to the customers. Capacity constraints are not considered. The objective is to minimize the expected completion time, that is, the total travel time plus the expected waiting time at the depot. The waiting time is due to the fact that the vehicle has to wait at the depot until the latest release date of a customer that it is going to serve. Given a solution to the problem, we call route a tour starting and ending at the depot and not visiting the depot in between. We call $K = \{1, \ldots, |K|\}$ the set of routes. Time is assumed to be discrete, i.e., $t \in \mathbb{N}$. All customers are known at $t = 0$, the beginning of the distribution, as either the goods have already been delivered to the depot or because the truck is en route to the depot. The release date of customer $i \in N$ at time $t$ is denoted by $\tilde{r}^t_i$. Based on the type of customer, $\tilde{r}^t_i$ is defined according to the following:

- if $i \in C$, $\tilde{r}^t_i$ is the known release date, $\tilde{r}^t_i = r_i$, where $r_i$ is the observed arrival time of truck $i$;
- if $i \in S$, $\tilde{r}^t_i$ is a random variable describing the release date with the information
available at the beginning of the distribution, \( \hat{r}_t^i = \hat{r}_0^i \), \( \forall t > 0 \);

- if \( i \in D \), \( \hat{r}_t^i \) is a random variable describing the release date at time \( t \).

In the latter two cases, the random variables are assumed to be bounded. The Gaussian distribution is taken as probability density function for the release dates.

3 Reoptimization policies

Because of the nature of the problem, a dynamic policy must be defined to identify the reoptimization epochs. Different strategies with an increasing number of reoptimizations are considered. The first and simplest policy solves the problem at the beginning of the distribution and each time the vehicle returns to the depot. The second policy reoptimizes at every return and every time the vehicle should depart from the depot to serve some customers. The third policy reoptimizes at arrival and expected departure and every given fraction of time the vehicle is waiting at the depot.

4 Solution strategies

The decision center may decide to solve the distribution problem in a deterministic way, deriving a point estimation of the release date from the random variable, or in a stochastic way, by handling the release dates as random variables. In either case, the evolution of the information is observed by means of the arrival of the trucks and the updates of the information provided by the GPS-equipped trucks.

4.1 Deterministic model

When a point estimation is considered for each release date, the solution of the deterministic model is achieved through a modified version of the heuristic presented in [1]. The deterministic information to be used in the model is obtained from the stochastic release dates as the conditional expected value of the release date. The original algorithm consists of an Iterated Local Search where the perturbation is performed by means of a destroy-and-repair (DR) procedure. The two components of the algorithm have been designed as follows. The DR is performed by removing a number of nodes from the input solution and adding them back in a randomly chosen route, different from the one they originally were included in. The Local Search (LS) considers a set of four neighborhoods: the relocation of a node to a different route, the merging of two consecutive routes, the split of a route in two separate ones, and the anticipation or postponement of a depot visit. The nodes are randomly sorted and considered sequentially: if the node is not the depot the node relocation move is considered, otherwise the remaining three moves are evaluated. The
first improving move is implemented and the process is repeated until no improving move is found. The ending conditions for the heuristic are the maximum time allowed for each run and the maximum number of iterations without improvement. The original algorithm has been modified to reflect the dynamic setting. In particular, the LS has been revised to be a best improvement local search where solutions will null improvement over the current best solution are implemented on the condition that the structure of the solution is improved. This condition is measured by evaluating the sum over all routes of the difference between the expected starting time and the maximum release date.

4.2 Stochastic model

If the information on the release dates is assumed to be uncertain, the problem is formulated as a stochastic model. A solution for the model is obtained by adopting the same heuristic scheme presented for the deterministic model, modified to consider the new objective function taking into account the stochastic component of the waiting times.

5 Computational experiments

Computational experiments have been designed to assess the performance of the different reoptimization policies in combination with the two solution strategies. Instances have been generated starting from the instances of the TSP-rd(time). The randomness and the dynamism of the release dates is generated by simulating the arrival of the trucks delivering the goods to the depot. The aim is to understand whether in the presented dynamic and uncertain setting it is best to consider a point estimation or the stochastic information of the release dates and if increasing the reoptimization frequency improves the quality of the solution. Preliminary results show that all solution methods, i.e., model-policy combination, provide a large improvement over a “myopic” algorithm serving customers as soon as possible.

References


A Dynamic Multi-Period General Routing Problem

Demetrio Laganà
Department of Mechanical, Energy and Management Engineering,
University of Calabria, Arcavacata di Rende (CS), Italy

Gilbert Laporte
CIRRELT and Canada Research Chair in Distribution Management,
HEC Montréal, Montréal, Canada

Francesca Vocaturo
Department of Economics, Statistics and Finance “G. Anania”
University of Calabria, Arcavacata di Rende (CS), Italy
Email: vocaturo@unical.it

1 Introduction

Arc routing problems (ARPs) and general routing problems (GRPs) arise in several applications related to garbage collection, mail delivery, and school bus routing. In the ARP, the elements requiring a visit correspond to links (i.e., arcs or edges) of the graph representing the street network of interest. In the GRP, these elements correspond to both links and vertices of the graph. The GRP is more general than the ARP. In particular, in real-world contexts, depending on their service demand and dispersion, the elements requiring a visit may be modelled as vertices or groups of elements as street segments.

We introduce and study a dynamic multi-period GRP (DMPGRP) with a mail delivery application in mind. Some items (e.g., letter post) must be delivered to some recipients. These items dynamically reveal themselves over a planning horizon consisting of several days. Then, delivery routes must be planned day by day with the aim of minimizing the total cost over the planning horizon. We remark that the decisions must be made without the knowledge of future items. This feature makes our problem dynamic and adjective “dynamic multi-period” differentiates it from a periodic routing problem where all information is available at the beginning of the planning horizon. The problem closest to the DMPGRP is the dynamic multi-period vehicle routing problem. Wen et al. [1] propose an interesting study on this problem by referring to the case of Lantmännen, a large Swedish distributor within the food, energy and agricultural industries.
2 Problem Description and Solution Strategy

In real postal delivery systems, the items seem to be delivered with a speed related to their nature. We imagine that the items are classified as: urgent (e.g., 24-hour express postal services), prominent (e.g., electricity bills), and unimportant (e.g., postcards). Therefore, we imagine that everyday new items arrive in the postal system and are classified through a label as urgent, prominent or unimportant. The delivery of prominent or unimportant items can be postponed. The labels of the undelivered items opportune change over the time. After some days, e.g., the undelivered items classified as prominent become urgent; similarly, the unimportant items become firstly prominent and secondly urgent. The undelivered items constitute the so-called “outstanding work” and are reconsidered the next day.

We focus on day $h$ of the planning horizon. Note that, even if the problem is dynamic, the delivery problem at the beginning of each day can be viewed as static since the routes for that day are planned based on the items arrived so far to the postal delivery system and the routes are fixed before their execution. An important step in the route design process is defining the service demand for day $h$. Specifically, for each recipient or group of recipients on a same street, the demand corresponds to the sum, in volume or weight, associated with all items addressed to it. Then, a label is associated with each demand. Note that common recipients (e.g., private households) on a same street are always handled as a unit, even when just one of them must be serviced. In other words, we always consider a link, instead of a vertex.

Three different typologies of customers are considered in the route design process for the $h$-th day: required customers, potential customers of level I, and potential customers of level II. The set of required customers, that must be serviced by the planned routes concerning day $h$, includes single recipients (vertices) or groups of recipients (links) with an urgent demand. The set of potential customers of level I (respectively, II), that can be serviced by the planned routes concerning day $h$, includes recipients/groups with a prominent (respectively, unimportant) demand.

2.1 Daily Delivery Model

For each day $h$ of the planning horizon, a mathematical model must be solved to select some (or all) potential customers and determine the vehicle routes. Formally, let $G = (V, A, E)$ be a mixed graph defined by a set of vertices $V$, a set of arcs $A$ and a set of edges $E$. Vertex $1 \in V$ represents the depot at which a set $K$ of identical vehicles of capacity $Q$ are based. Some subsets of arcs and edges, denoted respectively by $A_R \subseteq A$ and $E_R \subseteq E$, include required customers (more specifically, required links). Some subsets of arcs and edges, denoted respectively by $A_P \subseteq A$ and $E_P \subseteq E$ include potential customers of level I.
includes other required customers (more specifically, required vertices).

is serviced by vehicle \( x_{ij} \) and in a travel time \( t_{ij} \). Let \( \hat{A}_R \) be subset \( A_R \cup A_P \cup A_T \), and \( \tilde{E}_R \) subset \( E_R \cup E_P \cup E_T \). Every link \( (i,j) \in \hat{A}_R \cup \tilde{E}_R \) has a service demand \( d_{ij} \) and an additional service time \( s_{ij} \). Similarly, subset \( V_R \subseteq V \setminus \{1\} \) includes other required customers (more specifically, required vertices). Subsets \( V_P \subseteq V \setminus \{1\} \) and \( V_T \subseteq V \setminus \{1\} \) include other potential customers of level I and II (more specifically, potential vertices of level I and II), respectively. Let \( \hat{V}_R \) be subset \( V_R \cup V_P \cup V_T \). Every vertex \( i \in \hat{V}_R \) has a service demand \( d_i \) and a service time \( s_i \). Let \( L \) be the maximum route-length. Moreover, let \( \alpha \) and \( \beta \) be parameters concerning the inclusion of potential links or vertices in some route, respectively for levels I and II (\( \alpha > \beta \)).

The daily problem of selecting potential customers and determining vehicle routes consists of finding a set of at most \(|K|\) routes in such way that: (i) every required customer is serviced exactly once; (ii) every potential customer is serviced at most once; (iii) the vehicle capacity and the maximum route-length are not exceeded. The aim of the problem is minimizing an “amended” total cost, i.e,

\[
\sum_{k \in K} \sum_{(i,j) \in E_R} c_{ij}(x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in \hat{A}_R} c_{ij}x_{ij}^k + \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}(y_{ij}^k + y_{ji}^k) + \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}y_{ij}^k
\]

\[-\alpha \left( \sum_{k \in K} \sum_{(i,j) \in E_P} (x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{i \in \hat{V}_R} x_{ij}^k + \sum_{k \in K} \sum_{i \in V_P} z_i^k \right) \]

\[-\beta \left( \sum_{k \in K} \sum_{(i,j) \in E_T} (x_{ij}^k + x_{ji}^k) + \sum_{k \in K} \sum_{i \in \hat{A}_R} x_{ij}^k + \sum_{k \in K} \sum_{i \in V_T} z_i^k \right), \]

where \( x_{ij}^k \) is a binary variable that takes value 1 if link \((i,j)\) is serviced by vehicle \( k \) which travels from \( i \) to \( j \), \( y_{ij}^k \) is a non-negative variable representing the number of deadheading of link \((i,j)\), from \( i \) to \( j \), by vehicle \( k \), and \( z_i^k \) is a binary variable that takes value 1 if vertex \( i \) is serviced by vehicle \( k \). Note that values of \( \alpha \) and \( \beta \) opportune defined allow to service only convenient potential customers.

### 2.2 Heuristic Algorithm

Finding an optimal solution for the model just described through an exact algorithm is very time-consuming for instances of realistic sizes. In order to reach a good-quality solution for this type of instances, we propose a heuristic algorithm. The steps carried out to heuristically solve the model concerning the \( h \)-th day are briefly described in the following. First, we quickly generate routes servicing only the required customers (initial partial solution). Second, we use a procedure based on the well-known adaptive large

535
neighbourhood search framework to improve the initial partial solution. Third, we try to include in the improved partial solution potential customers of level I and II, by starting from feasible insertions of null cost. In this way, we generate a complete solution, in the sense that we consider all customers involved in the daily process.

3 Promising Outcome and Conclusion

In this section, we briefly discuss the results of a preliminary computational experiment and sketch out the future work.

We have generated an instance for our DMPGRP with recipients on three vertices and seven links of a mixed graph and with a planning horizon including seven days. Then, we have compared our solution strategy ($S_1$) with an alternative one ($S_2$). In both strategies, the branch-and-cut algorithm of Bosco et al. [2], opportunely modified, was used to solve a daily delivery model to optimality. In particular, strategy $S_2$ consists of immediately fulfill all demands, without postponing deliveries. In this case, the model proposed in [2] was extended by adding the maximum route-length constraints and solved for each day of the planning horizon. In strategy $S_1$, the model presented in Section 2.1 was solved by choosing appropriate values for $\alpha$ and $\beta$. The following assumption was made: an item classified as unimportant becomes prominent after two days if it is unfilled, while an item classified as prominent becomes urgent after one day. In addition, in order to make a fair comparison, we have imposed the delivery of all items at the last day of the planning horizon. Strategy $S_1$ outperformed strategy $S_2$, yielding a percentage cost saving approximately equal to 19%. Moreover, the heuristic algorithm described in Section 2.2 provided a solution near to the optimal one for $S_1$.

In the near future, the computational phase will be intensified in order to improve the overall solution approach. In particular, further experiments will be carried out with the aim of: (i) choosing values for $\alpha$ and $\beta$ better that the current ones; (ii) tuning techniques and parameters in the heuristic framework.

References


A multi-graph approach for the periodic vehicle routing problem with time spread constraints

Adria Soriano
Margaretha Gansterer
Karl F. Doerner
Department of Business Administration,
University of Vienna, Vienna, Austria

1 Introduction

In transportation industry, many variants of the well known vehicle routing problem (VRP) have been studied. One of the them refers to the transportation of goods with high value, known as cash-in-transit (CIT) transportation, which refers to transportation companies that deal with the physical transfer of banknotes, coins and items of value. Due to the valuable nature of the transported goods, CIT companies are exposed to a considerable risk of being attacked and robbed while performing their transfer services. Indeed, Yan et al. [7] state that a fair reduction of this risk can be achieved by doing a good study of the routing plan.

CIT companies usually plan over several days and have a fixed and known amount of customers that must be serviced in a daily basis. According to most existing works addressing this problem, a key element in transportation vulnerability is repetition of patterns. Making a routing plan difficult to predict reduces the risks of a vehicle to be intercepted and attacked. Route unpredictability can be addressed in two ways: (1) using dissimilar routes over the planning horizon, or (2) assuring a certain inconsistency in the time of service of customers.

Many works exist regarding route consistency, but very few has been done towards inconsistency. For the case of dissimilar routes, some related work has been done on the family of m-peripatetic vehicle routing problems and k-dissimilar vehicle routing problems (see Ngeveu et al. [2] and Talarico et al. [4]). In the case of periodic vehicle routing problems with inconsistency in the service time, the only works are presented, to the best of our knowledge, by Yan et al. [7] and Michallet et al. [1]. Both works set a minimum required difference for the time of service in a customer among all periods. Yan et al. model the problem as a network flow problem and solve it using a decomposition/collapsing technique.
coupled with the use of mathematical programming software. Michallet et al. develop an iterated local search procedure on a multi-start scheme, where four neighborhood movements are considered. They name the problem the periodic vehicle routing problem with time spread constraints on services (PVRPTS). Both of these works assume a simple representation of the network with a single path between each customer. Consequently, when a customer is visited at a certain time, the arrival time at the next customer can be automatically calculated. This situation can lead to inevitable violations of the time spread constraint, because arrival times at customers are “imposed” by the previous sequence. Nevertheless, if alternative paths were considered between two customers, which is a quite realistic assumption, a vehicle may benefit from choosing, for example, a longer path that allows the vehicle to arrive within the desired time to the next customer. Figure 1 shows an example of the positive effects of considering alternative paths on the PVRPTS.

Figure 1: Blue customer is visited at time 150. Desired visiting time for white customer is between 230 and 280. By using an alternative path instead of the regular edge of a simple shortest-path graph we save 20 units of service time violation, which can be much more costly than 40 extra units of distance/time.

In light of all described above, we address the PVRPTS with a multi-graph network representation. A multi-graph consists on a pair (V,E) of vertices and edges, where for each pair (i, j) of nodes there is a set A(i, j) of edges connecting them, with |A(i, j)| ≥ 1. The idea of considering a multi-graph approach in vehicle routing problems, though original for the PVRPTS, is not new. Ben Ticha et al. [5] show the benefits of a multi-graph setting on the VRP with time windows by comparing the results with the simple graph versions on modified Solomon instances. However, a multi-graph approach is commonly taken when more than one attributes are assigned to edges. In our case, the multi-graph perspective is also interesting in the sense that, even if a single attribute like time or distance is considered, having different alternatives for traveling between customers may lead to great reductions in the risk of assault to vehicles.

The contribution of this work is twofold: (1) we study the benefits of a multi-graph network representation, which can notably reduce the costs of violation of time constraints; (2) we propose an interesting method to tackle a multi-graph network problem and we assess its efficiency in single graph networks by comparing it with the results from Michallet et al [1].
Problem Definition and solution method

The PVRPTS consists of a set $N$ of customers connected by a set of $E$ edges, that have to be visited each day over a planning horizon of $m$ days. Customers might additionally have time windows. The objective of the problem is to minimize transportation costs while satisfying the timing constraints along the planning horizon. Spread time constraints are modeled by a threshold $\epsilon$, representing the minimum difference that must exist between the service times in a customer in different days. For instance, suppose a problem with a planning horizon of 3 days, a day length of 300 minutes and a value $\epsilon = 20$. Suppose also a customer that is serviced in days 1 and 2 at times 50 and 210 respectively. To comply with the time spread restrictions, it is not desired for this customer to be visited on day 3 between the time intervals $[30, 70]$ and $[190, 230]$. A multi-graph approach is taken for representing the transportation network of the problem. That means that in the set of edges $E$, some of them represent different paths connecting the same customers. These paths differ in distance and time, which for simplicity we consider proportional. Since timing constraints can be very demanding and lead to having very few feasible solutions, we solve the relaxed version of the problem where a penalty cost is imposed for the violation of these constraints. Thus the objective to minimize is the sum of distance and timing constraints violation costs.

For tackling the proposed problem we develop a large neighborhood search algorithm (LNS). LNS is a well known metaheuristic algorithm based on the principle of ruin and recreate (see Pisinger and Ropke [3]). These algorithms take a first initial solution and sequentially destroy and repair it through the use of different operators. For the PVRPTS we design a LNS algorithm with three destroy operators(Random, Worst, Related) and three repair operators(Greedy, Regret,2, Regret,3). Because of the problem complexity, a small change in the solution can lead to great differences in the objective function. Hence, for evaluating the moves in the LNS we make use of dynamic programming with forward and backward functions stored in the nodes (see Vidal et al.[6]), which integrate the costs of that node being serviced at a time $t$. Two versions of the LNS are generated corresponding to the single-graph and the multi-graph approaches, differing mainly in the computation of the aforementioned dynamic functions.

We obtain computational experiments for both approaches with the goal of comparing them. The good performance of the algorithm is proved in the good results obtained for the single-graph case, which improve benchmark results from [1] by more than 15%. At the same time, the positive effects of the multi-graph network can be seen for problems with low number of vehicles.
3 Conclusions

The PVRPTS is a relevant problem that can be of interest for CIT transportation companies. However, the scarce work existing in literature has focused on the basic problem with a simple transportation network. In this study, we intend to fill this gap by addressing the problem from a multi-graph network perspective. Thus, we exploit the multi-graph network setting to better tackle this existing trade-off between routing cost and service time inconsistency. We aim at showing that the violations of time service constraints can be greatly improved at a few traveling cost expense. We develop a LNS algorithm supported by dynamic programming functions for the evaluation of the costs of changes in the solution. The algorithm outperforms benchmark results in the single-graph case.

References


1 Introduction and Motivation

Everyday we interact with urban distributed service networks located on street graphs. Postal services empty on a regular basis post-boxes, banks replenish cash points, and municipal authorities collect different types of waste from containers dispersed in urban areas. This work paper examines the districting problem. The goal is to partition (cluster) urban geographical areas into smaller territorial units, called districts, zones or sectors, which are connected, compact and non overlapping, the workload among them is balanced and the dead-heading time to service them is minimised [1]. Common applications is the design public transportation pricing zones, the design of political districts, sales districts, electrical power districting, working zones and utility meter readings (see [2] and [3]).

Districting decisions are hard to make since they should consider various criteria associated with social, geographical, and transportational conditions. Furthermore, one has to model the trade-off between the balance of the workload for servicing the districts and the geographical compactness. The districting problem has received attention; however, few papers examine actual large scale districting problems based on real street graphs [1, 4].

This work contributes to the existing body of literature by providing new more enhanced criteria for the districting problem. It also presents novel solution frameworks including a new construction heuristic algorithm and a multi-objective local search meta-heuristic algorithm. Computational experiments on real data sets demonstrate the applicability, performance, and scalability of the proposed methods as well as indicate the interrelationships among the components of the objective function.

2 Preliminaries and Problem Definition

Let a undirected graph $G = (V, A)$. The set of nodes $1 \leq i, j \leq |V|$ represent street intersections, while the set of edges $1 \leq u, e \leq |A|$ represent the street segments. We
assume $G$ is connected and we measure all distances following the shortest paths. The service subgraph $G_S(V_S, A_S)$ consists of the subset of edges $A_S \subseteq A$ that has a service station and the deadhead subgraph $G_H(V_H, A_H)$ is defined as $G_H = G \setminus G_S$. A district $D_p$ corresponds to a set of edges $e \in A_S$, and $I_p$ denotes the corresponding set of nodes. Let the subgraph $G_p$ induced by $D_p$, to be $G_p = (V_p, A_p)$ with $A_p = D_p \cup A_H$. Note that, $D_p \subseteq A_p$ and $I_p \subseteq V_p$. The total working time $W_p$ is defined as the total time to visit and serve all edges of $G_p$ (service time $\sum_{e \in D_p} s_e$ plus deadhead time $TT_p$). To calculate $W_p$ we first transform the edges of $G_p$ to nodes (transformation procedure described in [6]) and we solve the corresponding travelling salesman problem via a Branch-and-Cut framework.

The districting plan $\mathcal{D}$ is a set of districts $\mathcal{D} = \{D_1, \ldots, D_p\}$, where $p$ is the given number of districts. A districting plan is feasible if each edge $e \in A_S$ belongs exactly to one district and for each $G_p$ there exist a Hamiltonian path visiting all edges $e \in D_p$. Note that, these paths may also traverse edges from the deadhead subgraph. The center of gravity $g_p$ is a point with coordinates $x_{cg_p} = \overline{x_p}$ and $y_{cg_p} = \overline{y_p}$, where the average $x$-coordinate $\overline{x_p} = \frac{\sum_{i \in I_p} x_i}{|I_p|}$ and the average $y$-coordinate $\overline{y_p} = \frac{\sum_{i \in I_p} y_i}{|I_p|}$. The central edge $c_p$ of a district $D_p$ is the edge that has the minimum sum of distances to all other edges of the district, i.e., $\forall e \in D_p \{\sum_{u \in D_p} d_{e,u}\}$. We adopt the algorithm of Graham [7] to find the nodes of the convex hull $CH_p$ of a district $D_p$. Based on the $CH_p$ the algorithm of Rosen [8] is used to determine the nodes of the concave hull $HC_p$ (note that $CH_p \subseteq HC_p$). This algorithm assumes a parameter angle $\theta$. Lastly, the frontier of a district $D_p$ is composed by all the edges that are connected to the nodes of $HC_p$ and belong to $D_p$.

We define six different objectives, namely, maximisation of the local compactness, minimisation of the global compactness, minimisation of the deadhead time, maximisation of the workload balance, maximisation of the coverage, and maximisation of the closeness. Compactness and closeness are important fitness indicators.

**Balancing the Workload.** The goal is to minimize the sum of relative deviations from the average working time $\bar{W}_p = W(\mathcal{D})/p$.

$$Bal(\mathcal{D}) = \sum_{k=1}^{p} \frac{|W_k - \bar{W}_p|}{\bar{W}_p}$$

(1)

**Local Compactness.** We introduce three metrics to measure the local compactness of each individual district. The first focuses on the roundness of the shapes:

$$Loc^{c1}(\mathcal{D}) = \sum_{i=1}^{p} (R_{out}^i - R_{in}^i)$$

(2)

where $R_{in}^p$ and $R_{out}^p$ are the radii of the inner and outer circles of district $p$, respectively. Both inner and outer circles share the same centers, that is, the center of gravity $g_p$. $R_{out}^p$ is equal to the distance between the center of gravity and the most distant node that
belongs to the the convex hull $CH_p$. $R_{in}^p$ is equal to the distance between the center of gravity the closest to it node that belongs to the concave hull $HC_p$.

The second metric focuses on the closeness of the boundaries to the central edge $c_p$. It is defined as the total sum of the square distances of all edges $e \in D_p$ to $c_p$:

$$\text{LoC}^2(D) = \sum_{i=1}^{p} \sum_{e \in D_i} d_{c_p,e}^2$$  (3)

The third metric focuses on the closeness and it is based on the sum of distances between all pairs of edges within a district:

$$\text{LoC}^3(D) = \sum_{i=1}^{p} \sum_{u \in D_i} \sum_{e \in D_i} d_{u,e}$$  (4)

**Global Compactness.** The global compactness examines the interrelationships among the districts. Butch et al. [1] measure the overlapping areas (if any) created by enclosing rectangles. We follow a similar approach, but instead we measure the overlapping area of the circles using the centre of gravity of the district as the centre and $R_{out}$ as the radius. Given the Euclidian distance $d_{p_1,p_2}$ between the two centres and the radius $R$ and $r$ of each district we calculate the overlapping area $E(D_i, D_j)$.

$$\text{GlC}(D) = \sum_{i=1}^{p} \sum_{j=i+1}^{p} E(D_i, D_j)$$  (5)

**Deadhead Time.** It is important to minimize the total deadhead time:

$$\text{DH}(D) = \sum_{i=1}^{p} TT_p$$  (6)

### 2.1 Two-phase Multi-Objective Local Search.

The proposed two-phase solution framework produces a set $F$ of efficient solutions considering all objectives of the problem. It employs a novel p-nearest clustering algorithm (enhanced variation of the well-known k-means algorithm) for producing initial solutions and a Tabu Search (TS) algorithm for local search improvement. Furthermore, we adopt the so-called proximate optimality principle, according to which high quality solutions at one level could potentially be close to high quality solutions of an adjacent level.

During the first phase, TS is applied multiple times, as many as the objectives, to find the initial points of the efficient frontier. Given an initial solution, TS seeks to find the best solution according to the single objective $f_1(D)$. Then, another TS starts from the best solution found in the previous level, and seeks to re-explore the solution space to optimize the second objective $f_2(D)$. This procedure is repeated until all objectives are explored. The efficient points produced (at least 6 plus any pareto solutions) are stored
in $F$. On this basis, during the second phase TS is applied to intensify the search around these points and improve the solution quality according to the global criterion method (see also compromise programming)

$$\min F_m(D) = \max_{i=1,\ldots,6} \left\{ \alpha_i \frac{f_i^{\text{max}} - f_i(D)}{f_i^{\text{max}} - f_i^{\text{min}}} \right\}$$

(7)

where $f_i^{\text{max}}$ and $f_i^{\text{min}}$ is the maximum and minimum values for the objective $i$ throughout the efficient points, respectively. The coefficients $\sum_{i=1,\ldots,6} \alpha_i = 6$ are used to avert the search to be trapped and they are self-tuned.

The TS algorithm performs swaps and re-allocates links, that exist on the boundaries of the districts, between adjacent districts. For this purpose we use the concave hull as the list of links to be swapped or re-allocated to other districts. During the neighborhood evaluation, various tricks are considered to speed up the feasibility checks and to avoid computationally expensive calculations.

### 3 Computational Results

The test-bed for our experiments was the real street network, which is organised in seven different zones. For each different zone, we used different options for the number of districts and in total the number of problem instances we generated is 9. Table 1 provides the % deviations between the solutions produced by using the multi-criteria objective and the single-criterion solutions. Each row corresponds to a problem instance, and we report the best found $F_m$. The deviations from all single criterion solutions vary from 0.02 % up to 510.98%. It seems that the solutions obtained by the multi-criteria objective exhibit large deviations from the best solution derived when $Bal$ is used as a single criterion. Essentially, a perfectly balanced solution is poor in terms of the rest of the criteria. Figure 1 shows a good balance of the compactness among the districts, since a compromise among all objectives is achieved for a better global criterion value.

<table>
<thead>
<tr>
<th>Zone</th>
<th>$k$</th>
<th>$F_m$</th>
<th>% Deviations to the single criterion solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\text{LoC}^*$</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>0.067</td>
<td>1.60</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>0.015</td>
<td>8.69</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>0.04</td>
<td>30.36</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>0.296</td>
<td>11.02</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.0012</td>
<td>7.07</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>0.1074</td>
<td>4.33</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>0.1145</td>
<td>3.83</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.0322</td>
<td>49.56</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>0.0022</td>
<td>29.32</td>
</tr>
</tbody>
</table>
4 Acknowledgment

This work was supported by the Research Centre of the Athens University of Economics and Business (EP262801) and (EP-2855-01).

References


1 Introduction and problem definition

E-commerce is a thriving market around the world and suits very well the busy lifestyle of today’s customers. An annual survey by analytics firm comScore and UPS revealed that consumers in US were purchasing more things online than in stores in 2016\(^1\). eMarketer estimated that e-commerce sales will top $4 trillion. It is obvious that this growing e-commerce poses a huge challenge for transportation companies, especially in the last mile delivery. According to [1], the last mile parcel delivery cost often reaches or even exceeds 50% of the total transportation cost, making it a top concern for many companies.

Nowadays, the most common last mile delivery service is home delivery. Customers wait at home to get their orders. Besides home delivery, companies like Amazon and FedEx, develop locker delivery. When customers shop online, they can choose a nearby locker as a pickup location. In the past two years, a new concept called trunk delivery, has been proposed. Here, customers’ orders can be delivered to the trunk of their cars. Volvo launched its in-car delivery service in Sweden in 2016. The courier has a one-time digital code to get access to the car.

Trunk delivery is different from home delivery and locker delivery since the car moves during the day and can be in different locations during the planning horizon. We study an efficient last mile delivery system that combines all these delivery services: home, locker and car trunk.

The problem can be modelled on a directed graph \( \mathcal{G} = (\mathcal{V}, \mathcal{A}) \). The set of vertices \( \mathcal{V} = \{0, 1, ..., N\} \) is partitioned into \( C_0 = \{0\}, C_1, ..., C_K \) clusters. Cluster \( C_0 \) contains only the depot. Each other cluster \( C_k, \ k > 0 \) represents the set of alternative locations on which client \( k \) can be delivered. Each vertex is associated with a time window (TW) \([E_i, L_i], i \in \{0, 1, ..., N\}\) with \([E_0, L_0] = [0, T]\). A visit can only be made to a vertex during its TW and an early arrival leads to waiting time while a late arrival causes infeasibility.

\(^1\)http://fortune.com/2016/06/08/online-shopping-increases/
Arcs are only defined between vertices belonging to different clusters, that is, \( A = \{(i, j) : i \in C_k, j \in C_l, k \neq l\} \). Each arc \((i, j) \in A\) is associated with a traveling cost \( C_{ij}\) and time \( T_{ij}\). A tour starts from and ends at the depot.

The objective is to find a minimum cost tour visiting each customer at one location in its itinerary during its TW. The problem that arises is called the Generalized Traveling Salesman Problem with Time Windows (GTSPTW). We assume that this problem is static and deterministic, namely all customer locations and the associated TWs are known with certainty in advance. The GTSPTW can be modelled as follows.

\[
\begin{align*}
\min & \quad \sum_{(i,j) \in A} C_{ij}x_{ij} \\
\text{s.t.} & \quad \sum_{i \in C_k} y_i = 1 \quad k \in \{0, 1, \ldots, K\}, \\
& \quad \sum_{(i,j) \in \delta^+(i)} x_{ij} = \sum_{(j,i) \in \delta^-(i)} x_{ji} = y_i \quad \forall i \in V, \\
& \quad E_i y_i \leq t_i \leq L_i y_i \quad \forall i \in V, \\
& \quad t_i - t_j + T_{ij} x_{ij} \leq L_i y_i - E_j y_j - (L_i - E_j) x_{ij} \quad \forall (i, j) \in A, \\
& \quad y_i \in \{0, 1\} \quad \forall i \in V, \\
& \quad x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A.
\end{align*}
\]

where \( \delta^+(i) = \{(i, j) \in A, j \neq i\} \) and \( \delta^-(i) = \{(j, i) \in A, j \neq i\} \). The objective function (1) minimizes the overall costs. Constraints (2) ensure that exactly one vertex from each cluster is visited. Constraints (3) are flow conservation constraints. Constraints (4) ensure that a vertex is visited during its TW. Constraints (5) ensure that the arrival and traveling times are consistent, meanwhile eliminating subtours. Constraints (6) and (7) are variable definitions.

When TWs are not considered, the GTSPTW reduces to the well-known Generalized Travelling Salesman Problem (Fischetti et al. [2]). For the multi-vehicle case, the problem is named Generalized Vehicle Routing Problem with TW (Moccia et al. [4]). The special case where TWs on clients do not overlap has been recently considered by Reyes et al. [3]. The problem is called the Vehicle Routing Problem (VRP) with Roaming Deliveries and models the case when only trunk deliveries are considered.

2 Methodology

2.1 Polynomial families of valid inequalities for GTSPTW

- Arc orientation inequality: \( x_{ij} + x_{ji} \leq y_i \quad i, j \in V, i \neq j. \)

In any feasible solution either \((i, j)\) or \((j, i)\) is used, but not both.
• Connectivity inequality. $x_{ij} + \sum_{h \in S_{ij}} y_h \geq y_i + y_j - 1 \quad i, j \in V, i \neq j,$
where $S_{ij} = \{ h \in V | E_i + T_{ih} \leq L_h, E_h + T_{hj} \leq L_j, E_i + T_{ih} + T_{hj} \leq L_j, i, j, h \in V, i \neq j, i \neq h, j \neq h \}$.
Two vertices are visited either directly or there exists a connection between them.

• Arc-or-vertex inequality. $x_{ij} + \sum_{h \in C_{ij}^k} y_h \leq 1 \quad i, j \in V, i \neq j, k \in \{1, \ldots, K\},$
where $C_{ij}^k = \{ h \in C_k | E_h + SP_{hi} + T_{ij} > L_j \text{ or } E_h + SP_{hi} > L_i, E_i + T_{ij} + SP_{jh} > L_h \text{ or } E_j + SP_{jh} > L_h, i, j \in V \setminus C_k \}.$
$SP_{ij}$ represents the shortest traveling time from vertex $i$ to vertex $j$. When the triangle inequality is not satisfied, the shortest path to go from $i$ to $j$ can include the visit of other vertices. These constraints detect if a vertex and an arc can not be simultaneously selected in a feasible solution due to TWs.

2.2 Exponential families of valid inequalities for the GTSPTW

• Generalized subtour elimination inequalities (GSECs). $\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1, \ S \subset V, \bar{S} = \bigcup_{k \in U} C_k, U \subset \{0, \ldots, K\}.$
GSECs avoid tours visiting a subset of clusters.

• Clique inequality. $\sum_{i \in S} y_i \leq |S| - 1.$
If no feasible path passing through all the vertices of a set $S \subset V$ exists (due to TWs), then the number of vertices of $S$ that can be visited in all the feasible solutions are less than the size of $S$.

2.3 A branch-and-cut scheme

We develop a branch-and-cut algorithm for the GTSPTW. We include at the root node of the branch-and-bound tree all the inequalities defined in Section 2.1. On the other side, due to their exponential cardinality, we separate the GSEC inequalities using a classical approach based on the resolution of a max-flow problem using Gomory-Hu algorithm.

Clique inequalities on subsets $S$ of $V$ with cardinality up to 3 are directly introduced at the root node of the branch and bound tree. On the other side, clique inequalities on subsets $S$ of $V$ with cardinality of 4 are dynamically added whenever a fractional solution violates one of them.

Please note that GSECs and clique inequality are satisfied by any feasible integer solution of the GTSPTW and they are not problem defining. On the opposite, they are not necessarily verified by the LP relaxation and are helpful to improve the lower bound during the branch-and-bound algorithm.
2.4 Initial solution

Initial solutions of the GTSPTW are obtained first creating a sequence of clusters and then solving a shortest path with resources constraints that visits one vertex per cluster respecting the TW. The sequence of clusters is obtained by generalization of the farthest, nearest and random insertion heuristics. Several solutions can then be computed and the best is used as a warm start for the resolution.

3 Discussion

The algorithm is implemented in C++ using Cplex 12.6 and the Concert framework. Preliminary results are obtained on instances that we created based on the benchmark GTSP instances to which we associated customer TWs. When creating an instance, we guarantee that a feasible solution exists.

Tests are run on a PC Intel(R) Core(TM) i5-6200U CPU 2.30Ghz and 8.00G RAM. The computation time is limited to 1 hour. Instances with up to 16 clusters and 76 vertices can be solved to optimality. Letting Cplex solving these 16 instances leads to an average computation time of 130.64 seconds. Using separation procedures decreases the average computation time to 94.79 seconds, while using an initial solution to 68.10.

For bigger instances (up to 24 clusters and 120 vertices) Cplex provides an average optimality gap of 21% after 1 hour of computation. This can be reduced to 18.53% using separation procedures and to 16.32% using the warm start.

Undergoing work is on the development of new families of cuts based on infeasibility detection.

References


Clustered routing for last-mile goods deliveries in urban areas: Formulation and a branch-and-cut algorithm

Antonio Martinez-Sykora, Tolga Bektaş
Southampton Business School
University of Southampton, Southampton, UK
Email: A.Martinez-Sykora@soton.ac.uk, T.Bektas@soton.ac.uk

Tom Cherrett
Transportation Research Group
University of Southampton, Southampton, UK
Email: T.J.Cherrett@soton.ac.uk

1 Challenges in urban deliveries

Parcel deliveries in urban areas pose significant challenges in terms of the limitations imposed by the road transportation infrastructure and the delivery characteristics of individual customers [1]. Limited kerbside space available within cities combined with one-way traffic may not always make it efficient to drive from one customer to another, particularly when the customers are located near one another (e.g., opposite side of a one-way street). In such cases, it may be preferable for a given driver to park the vehicle at a location from where deliveries may be made to several customers on foot. However, the total parking time would be constrained by a certain length of time as dictated by parking restrictions within urban zones, as well as the total weight or volume of cargo a driver could handle whilst delivering on foot. The demands themselves vary with respect to delivery time requirements and the weight or the volume of parcels.

Under such restrictions, a viable strategy for a single driver may be to use a combination of walking and driving, which involves jointly deciding on (i) the sequence of customer nodes to visit, (ii) the grouping (or clustering) of customers, and (iii) the locations of the parking points within each cluster. Under an objective of minimising the overall delivery time, making such decisions gives rise to what is known as the truck-and-trailer routing problem that was previously studied, e.g. [2]. In our setting, the truck corresponds to the driver and the trailer corresponds to the vehicle. To our knowledge, there exist two methods to solve a more general version of the problem, namely those described in [3] and [4], both of which are branch-and-price algorithms based on set-partitioning formulations.
However, our problem involves a separate treatment of the driving and walking options, as well as consideration of additional time constraints within each cluster. For this reason, our contribution in this work is to describe an arc-index mixed-integer programming formulation and a branch-and-cut algorithm to solve this problem.

## 2 Integer Programming

Let $G = (V, A)$ be a complete graph with $V = V' \cup 0$ is the set of nodes, where 0 the depot, $V'$ is the set of customers, and $A$ is the arc set. Each arc $(i, j) \in A$ has two different travel times associated with going from node $i \in V$ to $j \in V \setminus 0$, namely the time $c^d_{ij}$ of driving and the time $c^w_{ij}$ of walking. We allow the costs to be asymmetric. Each customer $i \in V'$ has a demand of parcels $d^i$ to be delivered between the time interval $(t_i^a, t_i^b)$. The aim of the problem is to (i) group the customers into clusters (where singleton clusters are allowed), (ii) find a driving route across the clusters such that one node within each cluster is visited and where the vehicle is assumed to have parked, and (iii) find a walking route within each cluster that starts and ends at the node where the vehicle is parked.

The objective is to minimise the total cost of all travel, under the following constraints: (i) the demands of all customers must be satisfied, (ii) The total walking time within each cluster is bounded by $D_w$, (iii) the total weight of all deliveries made within each cluster is limited to $Q$, and (iv) the driving starts from and ends at the depot. For the model we define a binary variables $x^d_{ij}$, which takes the value 1 if the driver drives from node $i \in V$ to node $j \in V \setminus \{i\}$, and 0 otherwise; $x^w_{ij}$ takes the value 1 if the driver walks from node $i \in V$ to node $j \in V \setminus \{i\}$ and 0 otherwise. A binary variable $z_i, i \in V'$ takes the value 1 if the location of customer $i$ is used to park the vehicle (hereafter called the `park node`), and 0 otherwise. For the time window constraints we define real variables $b^d_i$ and $b^w_i$ to represent the arrival time of the drive and walk to customer $i \in V'$, respectively. The objective function of the model is as follows, that minimizes the travel time:

$$\text{Minimise} \sum_{i \in V} \sum_{j \in V} \left( c^d_{ij} x^d_{ij} + c^w_{ij} x^w_{ij} \right).$$  \hspace{1cm} (1)

The following constraints allow the formation of the vehicle route,

$$\sum_{i \in V} x^d_{0i} = 1$$  \hspace{1cm} (2)

$$\sum_{i \in V} x^d_{i0} = 1$$  \hspace{1cm} (3)

$$\sum_{i,j \in S} x^d_{ij} \leq |S| - 1 \hspace{1cm} \forall S \subset V',$$  \hspace{1cm} (4)

namely, constraints (2) model the arcs leaving the depot, (3) model the arcs arriving to the depot and (4) are the subtour elimination constraints.
The following are constraints relevant to the walking routes from the customer chosen as the location where the vehicle is to be parked.

$$\sum_{j \in V} x_{ij}^w = z_i \quad \forall i \in V \quad (5)$$

$$\sum_{j \in V} x_{ji}^w = z_i \quad \forall i \in V \quad (6)$$

$$\sum_{i,j \in S} x_{ij}^w \leq |S| - 1 + \sum_{i \in S} z_i \quad \forall S \subset V', \quad (7)$$

where (5) and (6) model the arcs arriving into and leaving from the park node, respectively. Constraints (7) serve as to eliminate subtours from each of the walking tours.

One aspect of the problem is to ensure that walking routes must start and end at the same park node. To this end, we present several path elimination constraints that take different forms depending on the number of customers in a given path, as follows:

$$x_{ij}^w + 2 \sum_{u,v \in S \cup \{j,l\}} x_{uw}^w + x_{vl}^w \leq 2|S| + 5 - 2z_i - 2z_i' + z_j + z_l \quad \forall i, i', j, l \in V', \forall S \subset V' \setminus \{i, i', j, l\} \quad (8)$$

$$x_{ij}^w + 3x_{jl}^w + x_{i'l}^w \leq 6 - z_i - z_i' + 2z_j + 2z_l \quad \forall i, i', j, l \in V', \forall S \subset V' \setminus \{i, i', j, l\} \quad (9)$$

$$x_{ij}^w + x_{ij'}^w \leq 3 - z_i - z_i' \quad \forall i, i', j \in V'. \quad (10)$$

$$x_{ij}^w + z_i + z_j \leq 2 \quad \forall i, j \in V'. \quad (11)$$

where constraints (8), (9) and (10) model paths with more than 2, exactly 2 and exactly 1 customers, respectively, connecting two park nodes. The last constraint (11) ensures that two parking nodes cannot be connected.

The following set of constraints model weight restrictions on the walking routes,

$$x^w(\gamma(S)) \leq |S| - \lceil D(S)/Q \rceil + \sum_{i \in S} z_i \quad \forall S \subset V', \quad (12)$$

namely that the weight of parcels the driver can carry in each walking route is at most $Q$, where $x^w(\gamma(S)) = \sum_{i,j \in S} x_{ij}^w$ and $D(S) = \sum_{i \in S} d_i$. The constraints on the length of each walking route are as follows,

$$\sum_{r=1}^{p-1} x_{i_{r+1}i_{r+1}}^w + x_{i_{p}i_{p}}^w \leq p - 1 \quad \forall p > 0, \forall i_1, \ldots, i_p \in V', \sum_{r=1}^{p-1} c_{i_{r+1}i_{r+1}}^w + c_{i_{p}i_{p}}^w \geq \overline{D}, \quad (13)$$

which limits the time spent on each walking route by $\overline{D}$.

The two constraints below link the vehicle route and the walking routes.

$$\sum_{j \in V} (x_{ij}^d + x_{ij}^w) = 2 + 2z_i \quad \forall i \in V' \quad (14)$$

$$x^d(\delta(S)) + x^w(\delta(S)) \geq 2 \quad \forall S \subset V'. \quad (15)$$
The right hand side of (14) is 2 if the customer is not used as a parking node, however, if customer \( i \in V' \) is used as a parking node then it should belong to both the vehicle route and a walking route. Constraints (15) are for general subtour elimination. In addition to the above, the one novelty with the proposed formulation will be to incorporate time window constraints.

3 Branch-and-Cut Algorithm and Experiments

The outline of the algorithm is as follows. We first relax the inequalities (4), (7), (8), (9), (10), (12), (13) and (15). Inequalities (4), (7) are separated at all the nodes in the branching tree by using simple heuristic and exact procedures. Routines to separate inequalities (8), (9), (10), 12) and (15) are based on the ones presented in [2]. Finally, inequalities (13) are included as lazy constraints, i.e., are only separated on the tree nodes where an integer solution is found. Computational results obtained with the algorithm will be presented at the conference using real data sets obtained from urban deliveries.

References


1 Introduction

Bike sharing systems are becoming more and more popular throughout the world, doubling their number from 550 in 2012, to more than 1000 in 2016 [1]. This can be attributed to increasing interest in reducing pollution and traffic as well as promoting healthy lifestyles. Typically, bike sharing systems are financed by public and/or private institutions and managed by service providers, who are involved in strategic, tactical, and operational decision-making. Strategic decisions include determining the number, location, and capacity of stations where bikes can be rented, whereas tactical decisions include fleet sizing decisions. Daily, operational decisions include determining how to allocate, and periodically re-distribute, bikes to stations. This talk will present optimization approaches to assist such a service provider with these operational decisions.

When determining how to allocate and re-distribute bikes, service providers must con-
sider multiple objectives. First, they seek to limit the frequency with which an individual
arrives at a station in hopes of renting a bike, but none is available (which we term lack).
Second, they seek to limit the frequency with which an individual seeks to return a bike
to a station, yet it is full (which we term surplus). Both negatively impact the user’s ex-
perience with the bike-sharing service, as they both (potentially) require the user to travel
to another station. To measure the performance of our approach, we run a simulation
experiment and we focus on different Key Performance Indicators (KPI): the percentage
of surplus, the percentage of lack, the traveled miles for rebalancing, the customer service
level, and the entra inventory quantity. Of course, the likelihood of surplus and lack is
dependent upon how bikes are used during the day, including the stations they are rented
from and the stations they are returned to, both of which we assume exhibit stochasticity.

In this talk, we present stochastic optimization-based approaches to the decision-
making problem of allocating and re-distributing bikes, and show that explicitly acknowledge-
ing uncertainty leads to a significant improvement in both KPIs over using a deter-
ministic model. That said, by analyzing the upgradability of solutions to a deterministic
model, we derive a heuristic procedure for solving the stochastic model that significantly
reduce its solution time without losing solution quality. Finally, as the allocation of bikes
to stations can be seen as an inventory problem, we propose three heuristics based on
a Newsvendor model-type analysis. With our computational study, we show that the
stochastic program proposed outperforms these heuristics.

2 Problem and model description

Two decisions must be taken: (1) how many bikes at a central depot to allocate to each
station at the beginning of a day, and, (2) at a specific point in time later in the day, and
after the system has been in use, how bikes should be re-distributed/re-balanced between
stations. We assume that there is a limited number of bikes at the central depot that can
be allocated to stations. We consider rebalancing is executed by a capacitated vehicle that
travels along a fixed route that begins at the depot, visits each station, and then ends at
the depot. We assume this route is known a priori. While each station has a capacity,
it can also be expanded on a temporary basis. We refer to a station’s “demand” as the
difference between the returned and rented bikes at that station during the period between
the initial allocation of bikes and the opportunity to rebalance them. Note, this means that
demand can be positive (e.g. more rented than returned) or negative (more returned than
rented). Due to the demand nature, we also impose an initial bike quantity requirement
for each station. We consider that when the decision regarding the allocation of bikes
to stations is made, only a probability distribution station demand is known. However,
we also assume that when the rebalancing decisions are made, the service provider has
perfect visibility regarding the number of bikes at each station. As noted, when making these initial allocation and rebalancing decisions, the service provider seeks to both ensure a customer who seeks to rent a bike from a specific station can do so, and, a customer who seeks to return a bike to a specific station can do so. In addition, the service provider wishes to rebalance as little as possible and incurs a cost when allocating each bike to a station.

A complete review of the literature that is relevant to the problem we study is presented in [2]. Due to briefness reasons, we only note that a natural categorization of the relevant literature is into models that recognize uncertainty in bike usage (see for instance [3], [4]) and those that do not (see for instance [5]).

We believe our work presents three contributions to the literature: First, we present a stochastic program (SP) of the allocation and rebalancing problems, and with an extensive computational study assess the value of modeling uncertainty. Second, we present three different heuristics called Sequence Based Heuristics (SBH’s) for determining the initial allocation of bikes and illustrate their effectiveness computationally.

About the SP model, we first linearize it in order to solve with an off-the-shelf mixed integer programming (MIP) solver. Recalling that the service provider seeks to avoid surplus, lack, as well as excess allocation and rebalancing, the objective combines terms that model each occurrence. Lack is measured by the number of bike rental requests at a station that are in excess of the number of bikes that are positioned there at the end of the second stage. Surplus is measured with two terms, with the first measuring the number of bikes at a station above and beyond the number initially allocated, and the second measuring the number of bikes in excess of station capacity. The objective contains a term regarding initial allocation, which is measured as the number of bikes allocated to each station. Finally, the objective contains a term regarding re-balancing, which is measured by the total number of bike moved between stations. Associated with each term is a weight and the model seeks to minimize the sum of these weighted terms.

About the SBH’s, they differ in how the critical ratio is computed. Specifically, we compute it as a function of an underage (the weight for lack) and of an overage (the weight for surplus) component and we determine the number of bikes to allocate ensuring that the minimum requirement is met. Nevertheless, this calculation does not consider the total limited availability and stations capacity. To overcome these limits, we derive specific rules which are explained in detail in [2].

3 Numerical Results and Conclusions

We studied the real bike sharing system of the city of San Francisco through extensive numerical results. At first, we assessed the value of modeling uncertainty. We showed
that the expected value of postponing the delivery after demand realization is high and that the saving a decision maker can get from solving a SP instead of a deterministic one is consistent (29.15%). Motivated by this last result, we analyzed why the deterministic model performs so poorly [6]. We found out that the deterministic solution is too optimistic, since it suggests to deliver lower quantities (-40%) to fewer stations. Nevertheless, we found out that it is perfectly upgradeable and, consequently, it is a valid starting point for solving the SP with the total solution time reduced by 10%.

Furthermore, through the comparison between the solution to the stochastic model and the one obtained through the SBH’s, we showed that the SP outperforms the heuristics. Specifically, through the SP, the frequency of lack and the travelled miles reduce both by 14%, on average, while the inventory quantities reduce by 2% on average, with respect to the heuristics. These results can be attributed to a better allocation of bikes at stations and to the recognition of rebalancing. Consequently, even though our model can be thought as a Newsvendor-type problem, its performance is not as good as the one of our SP, since it does not consider the rebalancing of bikes through stations.

References


[2] Cavagnini, Rossana and Bertazzi, Luca and Maggioni, Francesca and Hewitt, Mike, “Multi-objective stochastic optimization-based approaches for managing a bike sharing system”, (submitted to Transportation Research Part E), 2018


The Bike sharing Rebalancing Problem with Stochastic Demands

Mauro Dell’Amico
Department of Sciences and Methods for Engineering,
University of Modena and Reggio Emilia, Reggio Emilia, Italy

Manuel Iori
Department of Sciences and Methods for Engineering,
University of Modena and Reggio Emilia, Reggio Emilia, Italy

Stefano Novellani
Department of Sciences and Methods for Engineering,
University of Modena and Reggio Emilia, Reggio Emilia, Italy
Email: stefano.novellani@unimore.it

Ananad Subramanian
Departmento de Sistemas de Computação,
Universidade Federal da Paraíba, João Pessoa, Brazil

1 Introduction

Bike sharing systems consist of a set of bikes and bike stations, located in different places of the city. Each station has a number of slots where users can collect or drop off a bicycle. For each station the number of bicycles to be present in the station to make it balanced defines the balanced level of occupation. Note that if all the stations are balanced then the system is balanced. Given a balanced system, after a certain amount of time it may become unbalanced because of the users’ collection and drop-off in different stations. This leads to inefficiencies for the users that may find empty or full stations. To increase the usage of the system and improve the satisfiability it is important to rebalance the system, which means that each station must be brought from the current situation to its balanced level of occupation.

The goal of this paper is to provide a solving method to rebalance an unbalanced situation of a bike sharing system by using a fleet of homogeneous capacitated vehicles at minimum cost where demands are stochastic. To tackle this problem, we propose Stochastic Programming models solved with exact and heuristic algorithms. In particular,
we present dedicated L-Shaped method and branch-and-cut algorithms, and some metaheuristic algorithms that make use of correlation between the stations, based on their stochastic demands.

Bike sharing systems have been treated in many works in the last decade by defining and solving several sets of optimization problems with several algorithms. One of them is the problem of redistributing bicycles among the stations, the Bike sharing Rebalancing Problem (BRP). This problem is part of the class of many-to-many pickup and delivery problem (see, e.g. [?]) and in its deterministic form, it has been firstly defined in [?]. Deterministic static and dynamic versions have been tackled, but, to our knowledge, no stochastic version has been treated. In this work we study the stochastic version of the BRP, namely the Stochastic Bike sharing Rebalancing Problem (SBRP). Other papers that study the bike sharing rebalancing type of problems are, see e.g. [?] and [?]. Stochastic many-to-many pickup and delivery problems have been tackled in [?]. However, according to our knowledge, the BRP with stochastic demands have not been considered yet.

2 Problem Description and Formulation

The Stochastic Bike sharing Rebalancing Problem is modeled on a complete digraph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ is the set of vertices including the $n$ customers and the depot (vertex 0) and $A$ is the set of arcs between each pair of vertices. For each vertex $j \in V \setminus \{0\}$ a request $\tilde{q}_j^\omega$ is given for every scenario $\omega \in \Omega$, where $\Omega$ is the set of all the possible scenarios and $p^\omega$ is the probability of the scenario $\omega$, knowing that $\sum_{\omega \in \Omega} p^\omega = 1$. The request represents the unbalance of the station, i.e. the difference between the current level of occupation and the balanced one. Requests can be positive or negative, so we consider a vertex as a pickup vertex, or a delivery vertex, if the request is positive or negative, respectively. The quantities picked up at pickup vertices can be used to respond to demand of delivery vertices or can be disposed to the depot. To satisfy requests some quantities can come from the depot, if needed. We further impose that a station with null request must be visited. In the SBRP the requests of different vertices and scenarios can be not respected by paying a penalty $k_j$ for each unit of slack and surplus in vertex $j \in V \setminus \{0\}$. These quantities of exceeding and lacking requests are limited.

The objective of the SBRP is to drive the fleet of $m$ identical vehicles of capacity $Q$ available at the depot to a set of routes in order to respond to the scenario requests, if possible, or by paying penalties, and minimizing the sum of the traveling costs and the penalty costs. Each of the station must be visited once and only once.
3 Solving Methods

We propose a Stochastic Integer Programming (SIP) model with simple recourse for the SBRP, that is a large model with all the scenarios. We decompose that model into two stages as typical for the two-stage SP models: the first stage takes into account the deterministic part of the problem, the decisions to take here and now, whilst the second stage model represents a set of models, one for each scenario, that take as input the decisions made at the first stage and make them fit with the request of the various scenarios after the realization. To perform this fitting a cut is produced and inserted into the first stage model. Both models must be solved in an iterative way: when the inequalities produced by the second stage models do not cut the solution of the first stage then the optimal solution has been found. That is most common way to solve a two-stage stochastic model: the L-Shaped method (see, e.g., [?]). One decided to insert one optimality cut for each scenario, this method is called Multi-cut L-Shaped method. In case we solve the problems while inserting feasibility and optimality cuts in a branch-and-cut fashion we call the implementation Multi-cut B&C. Moreover, we defined some good inequalities for the both methods. To strengthen the formulations we present metaheuristic algorithms to obtain an initial upper bound for the exact methods. These algorithms consider the initial solution built by using the well-known Savings and Closest neighborhood algorithms making a novel use of positive and negative correlations between the vertices’ requests in the evaluating function. The initial solution is then improved by three neighborhood that we defined to be applied in a Variable Neighborhood Descent framework. We also defined new properties in order to speed up the feasibility check during the construction of the solution an the exploration of the neighborhood.

4 Results

In Table ?? on can see some preliminary results of the two methods described above. The algorithms were run on an Intel Core i3-2100 CPU, 3-10 GHz, 4.00 GB, and we used CPLEX 12.6 as the MILP solver by imposing the selection of a single processor. We generated instances based on some real life data (see [?]). In each row we report on a single instance. In brackets we give the size of the instance and the number of scenarios. The possible number after the right bracket distinguish instances of the same size. In columns labeled UB and LB we report, respectively, the best upper and lower bound computed. Column labeled %gap reports the percentage error of the UB, versus the LB. Column labeled time reports the CPU time in seconds. A time limit of one hour has been given to each run. Multi-cut B&C results being the best performing method in both percentage gap between the upper and lower bounds and slightly in terms of computing times.
<table>
<thead>
<tr>
<th>Instance</th>
<th>Multi-cut L-Shaped</th>
<th></th>
<th></th>
<th></th>
<th>Multi-cut B&amp;C</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UB</td>
<td>LB</td>
<td>%gap</td>
<td>time (s)</td>
<td>UB</td>
<td>LB</td>
<td>%gap</td>
</tr>
<tr>
<td>Inst(14,172)</td>
<td>16900.0</td>
<td>16900.0</td>
<td>0.00</td>
<td>0.00</td>
<td>16900.0</td>
<td>16900.0</td>
<td>0.00</td>
</tr>
<tr>
<td>Inst(20,91)1</td>
<td>81192.1</td>
<td>81192.1</td>
<td>0.00</td>
<td>2.33</td>
<td>81192.1</td>
<td>81192.1</td>
<td>0.00</td>
</tr>
<tr>
<td>Inst(20,91)2</td>
<td>78824.7</td>
<td>78798.1</td>
<td>0.03</td>
<td>3600.00</td>
<td>79008.2</td>
<td>78253.5</td>
<td>0.96</td>
</tr>
<tr>
<td>Inst(20,30)1</td>
<td>40303.9</td>
<td>40303.9</td>
<td>0.00</td>
<td>54.37</td>
<td>40303.9</td>
<td>40303.9</td>
<td>0.00</td>
</tr>
<tr>
<td>Inst(20,30)2</td>
<td>39456.3</td>
<td>37592.0</td>
<td>4.73</td>
<td>3600.00</td>
<td>40938.0</td>
<td>37149.1</td>
<td>9.26</td>
</tr>
<tr>
<td>Inst(30,91)1</td>
<td>101976.8</td>
<td>101976.2</td>
<td>0.00</td>
<td>7.49</td>
<td>101976.2</td>
<td>101976.2</td>
<td>0.00</td>
</tr>
<tr>
<td>Inst(30,91)2</td>
<td>117158.7</td>
<td>117158.7</td>
<td>0.00</td>
<td>2.72</td>
<td>117158.7</td>
<td>117158.7</td>
<td>0.00</td>
</tr>
<tr>
<td>Inst(30,30)1</td>
<td>52401.3</td>
<td>45957.0</td>
<td>12.30</td>
<td>3600.00</td>
<td>52401.3</td>
<td>44236.9</td>
<td>15.58</td>
</tr>
<tr>
<td>Inst(30,30)2</td>
<td>61468.1</td>
<td>61468.1</td>
<td>0.00</td>
<td>6.30</td>
<td>61468.1</td>
<td>61468.1</td>
<td>0.00</td>
</tr>
</tbody>
</table>

5 Conclusions

We defined and solved the SBRP, the stochastic version of a tactical problem that arises in bike sharing systems. We proposed a two-stage stochastic linear programming model solved by means of a Multi-cut L-Shaped and by a Branch and Cut algorithms. To obtain upper bounds we developed two mataheuristic algorithms. We tested the algorithms on real-world instances proving the effectiveness of our methodologies. Future work could study more complex metaheuristic and new good inequalities.

References

1 Introduction

Station-based Bike-Sharing Systems (BSS) offer inexpensive and one-way capable trips between any station pair. One major challenge for operators is to provide sufficient bikes and free racks whenever requested by users [1]. To this end, operators redistribute bikes among stations over the day using a fleet of vehicles.

Recent studies related to BSS have proved that daily user demand displays stable and repetitive patterns [2]. For example, stations close to working areas tend to run full in the morning and empty in the evening due to commuting. Given these patterns, operators are able to design at the tactical planning level a redistribution plan for guiding daily vehicle operations. A redistribution plan is defined in terms of master tours, i.e., a time-ordered sequence of station visits, together with the necessary time to pick up or deliver bikes at each of such visits. In this work, we produce these master tours by solving a Service Network Design Problem.

Although demand forecasting is possible at the tactical planning level, there exists variations in demand when operators implement master tours at the operational level. In particular, high demand variations can lead to critical pitfalls in the redistribution planning. For this reason, we must evaluate the performance of master tours at the operational level.
2 Research Contribution

In this work, we study under which conditions redistribution planning is suitable for guiding vehicle operations. We provide evidence that master tours obtained from service network design are particularly useful for BSS with noticeable commuter dominance. We also study the effects of safety buffer on mitigating variations in demand. A safety buffer defines the minimal number of bikes and free racks that a station should have anytime [4]. We show that large safety buffers only make sense if a sufficient number of vehicles is available.

3 Methodology

First, we produce master tours with a formulation of service network design (SND) at the core of the tactical planning level. Then, we test the performance of such master tours in a simulation approach which emulates existing conditions in the operational level.

3.1 Service Network Design Formulation

The formulation of SND is based on a time-expanded network flow representation of expected user demand and redistribution decisions. Consider the exemplary time-expanded network in Figure 1. The given network infrastructure consists of two stations and the vehicle depot, see y-axis. The x-axis depicts chronologically-ordered time periods. Each station and time period is represented by two nodes. The first node is called rack node indicating the number of bikes in racks. The second node is the vehicle node indicating the number of bikes in the vehicle. The time-expanded network involves three type of flows. Ride flows are an abstract representation of user’s desired trips. These ride flows are obtained by aggregating recorded trip data over several days as input of SND. Vehicle flows depict one master tours. In the figure, the vehicle firstly visits station 1 to pick up bikes, then it arrives to station 2 to deliver the target bike volume. Notice that the formulation explicitly considers the necessary handling time to pick up and deliver bikes at stations. Redistribution flows depict these handling operations as well as the movement of loaded bikes between stations. The formulation takes the form of a MIP which determines master tours.

3.2 Simulation Approach

The simulation considers stochastic user demand which is gradually revealed over time. Users make decisions to achieve a station of destination in a short travel time [5]. If a rental request fails because the station is empty, the user either roams to the closest station with bikes or walks to the station of destination. If a return request fails because the station
is full, the user must roam among stations until he finds a free rack. Redistribution decisions are guided by the long-term strategy presented by [4]. The simulation receives from a solution of SND the target number of bikes in racks at stations and time period. The long-term strategy tells the number of bikes to pick up or deliver in order to achieve this target number of bikes in racks.

4 Preliminary Results and Discussion

We present the results of a case study including data from a BSS located in Boston, United States. This BSS is characterized by a clear commuter dominance. Let $v$ be resources in terms of vehicles. We denote by $\beta$ the safety buffer of bikes and free racks. Thus, we produce redistribution plans given different tuples $(v, \beta)$ as input.

We report the average improvement ratio and efficiency of 500 runs of the simulation approach. The improvement ratio is the percentage reduction of failed requests with respect to no redistribution. We define efficiency as the number of reduced failed requests per picked up or delivered bike, in comparison with the case where no redistribution takes place.

In Figure 2, the x- and y-axis of each square area depict the values of $v$ and $\beta$, respectively. Black fill areas indicate non-existence of information. On the left, we observe that excessive values of $\beta$ are counterproductive if we do not have enough vehicles to address them. As expected, major improvement ratios are obtained with large safety buffers and vehicles. On the right, we note that large values of $\beta$ leads to an inefficient redistribution of bikes.
Figure 2: Trade-off between resources and safety buffer requirements.

5 Conclusion

Computational experiments provide evidence that the formulation of SND produces suitable master tours if a BSS have clear commuter dominance. Although safety buffers prevent day-to-day variations in demand, they also lead to more redistribution efforts.

References


Modeling and Solving a Real-Life Fuel Delivery Problem with Safety Considerations

Konstantinos N. Androutsopoulos
Department of Management Science and Technology, Athens University of Economics and Business, Athens, Greece

Nikolaos K. Kamaras
Department of Management Science and Technology, Athens University of Economics and Business, Athens, Greece

Email: kandro@aueb.gr

1. Introduction

The fuel distribution problem addressed in this paper aims to determine the delivery routes for a heterogeneous fleet of multi-compartment vehicles for servicing a set of customers taking into account proactive risk mitigation practices. Each customer places multiple orders referring to different products. The problem is modeled as a Multi-compartment Vehicle Routing Problem with Time windows and Loading/Unloading constraints. Two major categories of multi-compartment routing models can be found in the literature depending on whether the use of each compartment is dedicated to a specific product type[1] or not. Fuel delivery problems fall under the latter model category. Depending on whether the orders are specified by the clients or the supplier, the fuel delivery problems are modeled as: i) multi-compartment vehicle routing problems (MC-VRPs) [2,3,4,10] or ii) multi-compartment inventory routing problems (MC-IRPs) [5,6,7,8,9], respectively. The models in both categories of fuel delivery problems can be further classified by enforcing or disregarding the following constraints: i) each compartment is not allowed to host more than one order (i.e., no compartment split is allowed), and ii) each order is serviced by a single vehicle through a single visit (i.e., order split is not allowed). Most of the MC-VRPs [4,10] and MC-IRPs [6,7] addressed in the literature assume that both constraints (i) and (ii) are enforced. However, Derigs et al.[3] and Mendoza et al.[2] deal with a more general MC-VRP in which compartment split is allowed. Cornillier et al.[5] deal with a MC-IRP in which order split is allowed. Coelho et al.[8] model and solve the MC-IRP in which compartment split and order split are allowed. They also address all variations of this problem emerging from enforcing one or both constraints (i) and (ii).

The MC-VRP problem addressed in this paper forbids compartment split and order split. Specific transportation safety constraints are taken into account upon loading and unloading a vehicle. This problem is not previously addressed in the literature. It is worth noting that Cornillier et al. [6] provide a MC-IRP in which additional precedence constraints regarding the sequence of unloading the compartments of the vehicle are taken into account. However their work is narrowed by the assumption that each vehicle cannot be assigned more than two
customers. The objective of this paper is to elaborate on the major features of the problem and provide a solution approach that addresses real life instances of the proposed problem. Results regarding the computational performance of the proposed algorithm are presented. The proposed solution approach has been integrated in a Decision Support System that has been implemented and used for a Fuel Distribution company in Greece. A discussion on the impact of the proposed solution approach on the performance of the actual distribution process is also provided.

2. Problem Definition

It is assumed that a fuel distribution company offers a variety of products \( P \). Each customer \( i \in C \) places a set of orders \( R_i \) each one \( (r \in R_i) \) involving a single (different) product. The company services the customers through a heterogeneous fleet of vehicles \( V \) each one \( v \in V \) comprising of a set of non-identical compartments \( K_v \). The problem involves assigning customers’ orders to vehicle compartments and determining delivery routes for servicing all customers under the following criteria: i) minimising the number of vehicles used, ii) minimizing the total travelled distance and iii) maximizing the average capacity utilization. A significant feature of the proposed problem relates to the rules that dominate the assignment of customers’ orders to vehicle compartments:

- Each compartment can be assigned no more than one order.
- The orders \( R_i \) of customer \( i \) must be assigned to compartments of the same vehicle.
- The weight of the orders assigned to a vehicle must not exceed the maximum net weight of the vehicle. In addition, if the volume capacity of a compartment exceeds 7,500 lt, then the quantity of an order assigned to it must be either above 80% or below 20% of the compartment’s capacity.
- The load of a vehicle throughout a delivery route must be distributed on the axes of the truck in a way that it does not affect the (en-route) stability of the vehicle.

In practice, the last constraint is implemented by applying predefined loading/un-loading rules (that prevail in the Greek fuel distribution industry) which specify the sequence in which the compartments of the vehicle can be unloaded. A basic loading rule is to avoid having one or more empty compartments between loaded compartments. Two alternative unloading rules are the following:

i. Unloading the compartments of the vehicle in the reverse order from their physical ranking, starting from the last and moving to the front compartment of the vehicle.

ii. Sorting the compartments of a vehicle to three groups depending on their physical ranking, i.e. front, middle, tail, and then unload them in the following order: middle, tail, front.

It is worth noting that unloading rule (ii) can be transformed to rule (i) if the numbering of the compartments is modified (reshuffled) appropriately. In the remainder of the paper, we assume unloading pattern (i) is applied without loss of generality. Based on the loading/unloading constraints, any order \( r \in R_i \) of a customer \( i \) can only be assigned to an arrangement of consecutive compartments. Moreover, if two customers \( i \) and \( j \) are visited in
successive order, then the sequences of compartments assigned to the orders corresponding to customers $i$ and $j$ must be consecutive as well.

3. Solution Approach
A GRASP heuristic algorithm was developed for solving the proposed fuel delivery problem. Each iteration of the proposed algorithm involves three phases I-III. Phase I executes a sequential route construction routine which determines the assignment of the customers to the vehicles. The construction routine selects iteratively the vehicle with the highest capacity and builds its route by selecting and inserting customers in the last position of the route (i.e. between the last customer and the destination). The route construction routine for the selected vehicle is initiated by selecting the un-assigned customer that minimizes the travelled distance of the vehicle. Then the proposed routine constructs a candidate list including all the customers that could be feasibly assigned to the vehicle and performs a roulette wheel experiment that selects probabilistically one of the candidate customers, taking into account an insertion metric calculated for each one of them. Upon termination of the route construction of a vehicle, a reloading routine is performed aiming to maximize the relevant capacity utilization. This routine searches in the list of used and un-used vehicles for a new vehicle that could be feasibly assigned the orders of the current vehicle, while attaining improved capacity utilization performance. The route construction routine terminates when all the customers are assigned to vehicles. Phase II involves a co-loading and a merging-route routine aiming to maximize the capacity utilization of the algorithm’s solution. The co-loading routine tries to reduce the number of used vehicles by re-assigning the orders loaded to a vehicle to any other used vehicle. The merging-route routine tries to reload the orders assigned to two different vehicles on a either of the two or alternatively to new un-used vehicle. Phase III involves a local search routine aiming to improve the travelled distance of the solution. The neighborhood of solutions around the current solution is attained by exchanging customers between vehicles and repositioning customers within the same vehicle. If the travelled distance of the solution in the generated neighborhood is lower than the corresponding travelled distance of the current solution, then the identified improved solution is considered as the new solution and the local search process is repeated around the new identified solution. The local search routine is terminated if either a predetermined number of iterations are performed or no further improvement can be achieved.

4. Preliminary Results and Conclusions
A set of test problems were solved in order to assess its performance in terms of: i) computational time and ii) solution accuracy. The proposed GRASP heuristic algorithm was iterated for 100 times (per test problem), and the average computational time (per iteration) required for solving test problems of 50,100 and 150 customers was 1.26, 7.26 and 16.61 seconds, respectively. All computational tests were performed on an Intel Core 3.60 GHz computer with 64-bit Windows operating system and 16GB RAM. The computational tests for assessing the accuracy of the algorithm’s solution are on the way.
The proposed solution approach was used to solve large sized real life problems (300-600 customers) of a Greek fuel distribution company. The emerging solutions were compared with the ones determined by the dispatchers of the company. The comparative assessment was based on the following performance measures: i) the travelled distance of the vehicles and ii) the total number of used vehicles. The travelled distance of the solutions determined by the proposed solution approach for the test problems was 41.22% lower than solutions determined by the dispatchers. The computational time for solving the real life problems ranged from 3-20 min.

5. Acknowledgements
The presented research work was partially supported by the Research Center of the Athens University of Economics and Business (AUEB-RC) through the projects EP-2479-01 & EP-2638-01.

6. References
Express delivery in freight transportation: an application to the trucking industry

Luigi De Giovanni
Dipartimento di Matematica “Tullio Levi-Civita”,
Università di Padova, Padova, Italy Email: luigi@math.unipd.it

Nicola Gastaldon
Dipartimento di Matematica “Tullio Levi-Civita”,
Università di Padova, Padova, Italy

Filippo Sottovia
Trans-Cel Autotrasporti, Albignasego (Italy)

1 Introduction

Small trucking companies are vital to the supply chain as they target a niche market of high quality and customized freight transportation services. They ask for tailor made tools to support daily operations, giving rise to interesting vehicle routing problems. This is the case of Trans-Cel, a small trucking company based in the Venice area (Italy) and developing a web platform to support trucking operations by interconnecting different modules (route planning, on-line demand management, truck tracking, route operation etc.) that share real-time and historical data. The company owns trucks with heterogeneous capacity, loading facilities and operational costs. Clients ask for express pickup-and-delivery services, with regional haul, hard or soft time windows falling within the same day or two consecutive days, and, sometimes, just-in-time requirements (e.g., orders issued a few hours in advance) and multiple pickups or deliveries. A revenue is associated to orders, together with penalties for, e.g., soft time windows violation. Routes start from the last position of the day before (open routes) and are subject to classical hours-of-service regulations, with possible preferences on, e.g., maximum duration, ending position, order-to-vehicle assignments, order execution precedence etc. Given the orders, the fleet status and information on the road network, we want to determine a set of pickup-and-delivery routes that maximizes the profit, taking operational constraints and preferences into account.

The problem is a Multi Attribute Vehicle Routing Problem (MAVRP, e.g. [1]) to be solved in both static and dynamic settings, and presents interesting peculiarities. The static version concerns the a-priori definition of initial daily routes. Literature proposes
exact algorithms (e.g. [2, 3]), or heuristic approaches mostly based on genetic algorithms and tabu search (e.g. [1, 4], or [5], directly inspired to the MAVRP described above). The dynamic version responds to a variety of dynamic events, like on-line issued orders, truck failures, delayed operations, road network modifications. Literature approaches can be classified into reactive policies, that take into account only known information on past pending requests and the new ones, and anticipatory algorithms, exploiting stochastic information on future requests to better accommodate the new ones [6]. Some reactive policies, feasible when the level of dynamism is relatively low, continuously or periodically re-optimize the routes, based on fast algorithms for the static version (e.g. [7]).

The peculiarities of the MAVRP considered here mainly come from mixing attributes of different application settings: on the one hand, order distance, dynamism and size, as well as route properties, are typical of intercity less-than-truckload (LTL) couriers; on the other hand, urgent orders or on-line requests raise issues similar to express delivery in urban context, reducing, for example, the impact of freight consolidation, typical in LTL settings. Moreover, the assumption that, in dynamic settings, fixes next destination on a vehicle route [7] may be not realistic in long haul trucking, where diversion may be compatible with time windows and vehicle capacities. Our objective is to devise a solution algorithm for MAVRP able to address the above peculiarities and suitable for integration in operations’ manager computerized support tools.

2 Algorithmic approach

MAVRP has to be solved by different modules of the support platform. The operations management office interacts with the planning module to determine the initial daily routes, in case suggesting a starting (even partial) solution, which asks for solving MAVRP in static settings. Other modules solve MAVRP in dynamic settings: the demand management module asks for dynamically inserting a new proposed order as to compute a marginal cost, useful during the negotiation phase; the insertion of new orders, as well as network congestion or delayed pickups/deliveries ask for dynamically modifying routes during their execution. The trucking company business model defines a relatively low dynamism allowing for a re-optimizing reactive policy. This imposes response times within seconds and, due to the inner complexity of MAVRP, suggests a metaheuristic approach. We propose the two-levels local search heuristic described in [5] with improved efficiency and assessed solution quality. At the first level, a tabu search determines the order-to-vehicle assignment by exploring neighborhoods obtained by: (1) displacing one order from route $a$ to route $b$; (2) swapping the assignment of two orders; (3) displacing two orders from route $a$ to route $b$; and (4) displacing one order from route $a$ to route $b$ and another order from route $b$ to route $c$. Neighborhoods are explored in the given sequence by
selecting the best solution from the first improving neighborhood. In case of no improvement, we select the best neighbor (deterministic search) or one neighbor picked at random among the best five (randomized search). The evaluation of neighbor solutions (second level) is realized by a 2-opt heuristics determining the pickups and deliveries sequence of each route. The score function is a weighted sum of revenues, operational trucking costs, soft time windows penalties etc., as well as further artificial penalties to take preferences into account and allow visiting unfeasible solutions. A best insertion procedure provides initial routes. The algorithm has been modified to allow hot start optimization from user provided solutions (static setting) or from the real-time fleet, network and order status (dynamic setting). In order to accelerate convergence, we propose a more general definition of neighborhood 3, together with a granular exploration that limits the search to displacing the most promising order subsets, selected on a distance criterion. To further reduce running times, we devise two parallel implementations: intra-neighborhood, that uses independent threads to explore distinct parts of each neighborhood and inter-neighborhood, that simultaneously explores every neighborhood. We get a preliminary bound to the optimal score function value from the linear relaxation of a path-based integer programming formulation of MAVRP, solved by column generation. The pricing algorithm implements a classical dynamic programming procedure for the elementary shortest path problem with resource constraints, adapted to orders with multiple pickups or deliveries.

3 Computational results and discussion

The above algorithm has been implemented in C++ and integrated in the prototype support platform currently in use at Trans-Cel, after a training phase to calibrate parameters and compare to solutions provided by the operations’ manager, which attested for estimated 9% better scores on average [5]. From a managerial point of view, the solution proposed by the support system suggested new operational strategies, including open routes and alternating pickup and delivery operations, which were rarely preferred in the past: in fact, results shows that, in the peculiar settings of our trucking company, the advantages from freight consolidation are compensated by more efficient routes. Table 1 reports further sample results on 22 real static instances solved from scratch with, on average, 35.3 orders (up to 42), 77.4 pickup/delivery operations (up to 116) and 12 to 15 trucks. Columns report results for different versions of the algorithm: deterministic search (DET), three additional randomized searches (RND), granular exploration of pairs of orders from neighborhood 3 (+R), intra- and inter-neighborhood parallelizations on four threads (+P1 and +P2). They are compared in terms of average, minimum and maximum per-cent gap with respect to the baseline RND solution, and running time in seconds on an Intel Core i5-5200 2.20 GHz CPU with 8 GB RAM. The bounding procedure converges
Table 1: Sample computational results.

<table>
<thead>
<tr>
<th></th>
<th>RND</th>
<th>DET</th>
<th>RND+R</th>
<th>RND+R+P1</th>
<th>RND+R+P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gap%</td>
<td>baseline</td>
<td>0.7 (0.0 ; 3.6)</td>
<td>0.4 (-1.0 ; 4.6)</td>
<td>0.4 (-1.0 ; 4.6)</td>
<td>0.5 (-1.1 ; 6.6)</td>
</tr>
<tr>
<td>Time (s)</td>
<td>52.9 (0.9 ; 481.2)</td>
<td>12.1 (0.2 ; 122.9)</td>
<td>26.9 (0.8 ; 135.2)</td>
<td>16.5 (0.7 ; 103.0)</td>
<td>23.0 (0.4 ; 165.3)</td>
</tr>
</tbody>
</table>

for 16 instances, proving the optimality of three solutions from RND, and an average 1.5% gap in the remaining cases. Granular exploration preserves solution quality (only a 0.4% loss) while reducing running times by 50%. Further 39% or 14% reductions come from, respectively, intra- and inter-neighborhood parallelization, so that running times are suitable for Trans-Cel operational settings. Further parallelization is under implementation, where different threads run deterministic and randomized searches, sharing long and short term memory to alternate intensification and diversification phases. The availability of historical and real-time data on orders and route operations triggered current research on dynamic and stochastic versions of MAVRP in small trucking companies.

References


Introducing flexibility and demand-based fairness in slot scheduling decisions

Jamie Fairbrother
Centre for Transport and Logistics,
Lancaster University, Lancaster, United Kingdom

Konstantinos Zografos
Centre for Transport and Logistics,
Lancaster University, Lancaster, United Kingdom

1 Introduction

In many airports around the world, demand by airlines to use airport infrastructure exceeds the available capacity, leading to congestion problems. Outside of the US, demand at airports is managed through an administrative scheme known as the IATA Worldwide Slot Guidelines (WSG) [1]. Under this scheme, congested airports are designated as coordinated, and airlines must obtain slots, which is a time interval during which an aircraft can use the airport for landing or take-off. To obtain slots, airlines biannually make requests for slots and series of slots to the coordinator of an airport for a six-month season. A series of slots is a request for five or more slots at the same time and on the same day of the week. The coordinator then constructs an initial allocation of slots to airlines which matches the requests as far as possible while satisfying airport capacity and aircraft turnaround constraints. In addition to these logistical constraints, the coordinator must also allocate slots according to priority classes.

Due to the complexity of assigning slots, many optimization models have been formulated to solve this problem. These models typically aim in some way to minimize the schedule displacement, that is the time difference between requested and allocated slots. More recent models [2, 3] have incorporated fairness, that is equitably distributing schedule delay among the airlines. Although the requirement of fairness will necessarily increase the overall total displacement, it will ensure that the resulting schedule is more acceptable to the airlines.

The aim of this paper is to develop a new multiobjective optimization model which enhances previous models in three directions. Firstly, our model allows for flexibility in the slot times assigned to a series of slots. Secondly, we propose constraints to fairly allocate
rejected requests, that is requests which cannot be allocated any slot. Finally, to fairly allocated schedule displacement we propose a new demand-based fairness measure.

2 Modelling extensions

The model we develop is an extension of that in [4]. The main differences being that the decision variables controlling the time of the slot assigned to a request is now additionally indexed by the day of the season, and we include an extra set of variables to indicate whether request are rejected, which occur if for some day the total number of slots is exceeded by demand for slots. The notation used in the model is shown in Table 1 and the model itself is in equations (1)–(11).

**Objective, capacity and turnaround constraints** The objective of the model in (1) is the lexicographic minimization of rejected requests, followed by total displacement. We minimize in this order as there will be a much greater aversion to rejections than to displacements. Constraints (3) ensure that each request is assigned a slot, or is explicitly rejected; constraints (4) ensure that the allocation complies with rolling capacity constraints, that is constraints which limit the number of movements scheduled within a given amount of time of each other; finally constraints (5) ensure there is sufficient turnaround for arrival-departure pairs of flights.

**Flexibility** Previous slot allocation models have allocated slots for a series of slots request at the same time. However, the WSG allow for some flexibility in how these are assigned. By indexing the assignment of a slot by the day, we are allowing each day for which a request is made to be assigned a slot at a different time. Generally, it is preferable to assign series of slots which are as close to each other as possible. For each request we therefore place a bound (8) on the range of slot times that one can assign to a request. Constraints (6) and (7) are logical constraints which define the earliest and latest slot times assigned to a request. Allowing flexibility while constraining slot range significantly increases the size of the model and the computational effort required to solve it.

**Fair apportionment of rejected requests and schedule displacement** For both rejections and schedule displacement we use the proportionality principle of fairness proposed in [3]. Rejections of requests occur when there are too many requests to fulfil in a single day. We allocate a proportion of rejections to an airline a proportional to the proportion of requests $p_a$ made by the airlines. To enforce this, we add maximum deviation from absolute fairness (MDA) constraints (9).

In the case of schedule displacement, the amount a request should be displaced depends on how severe is the demand for the requested time slot. We measure the severity through
the marginal cost of a request, denoted by $\nu_m$ for $m \in \mathcal{M}$. The marginal costs are calculated in terms of schedule displacement from an exhaustive sensitivity analysis with respect to the model below, without the two sets of fairness constraints. The values can be thought of as a request’s contribution to the violation of capacity constraints. We then require the proportion of schedule displacement assigned to an airline to be proportional to a weighted sum of the marginal costs of its requests via MDA constraints (10). By only counting requests which increase total displacement, airlines are not penalised for making requests in off-peak periods, where there is sufficient capacity to meet demand.

$$\text{lexmin} \left( \sum_{m \in \mathcal{M}} z_m, \sum_{m \in \mathcal{M}} s_m \right)$$ 

subject to $s_m \geq \sum_{d \in \mathcal{D}_m} \sum_{t \in \mathcal{T}} f_{td} m y_{td}$

$$\sum_{t \in \mathcal{T}} y_{td}^m + z_m = 1, \ m \in \mathcal{M}, \ d \in \mathcal{D}_m$$

$$\sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{T}} a_{m} b_{mc} y_{td}^m \leq u_{ds}^c, \ c \in \mathcal{C}, \ d \in \mathcal{D}, \ s \in \mathcal{T}_c$$

$$\sum_{t \in \mathcal{T}} y_{td}^m - \sum_{t \in \mathcal{T}} y_{td}^{m_2} \geq l_{(m_1,m_2)}, \ (m_1, m_2) \in \mathcal{P}, \ d \in \mathcal{D}_m$$

$$\tau_m \leq \sum_{t \in \mathcal{T}} y_{td}^m, \ d \in \mathcal{D}_m$$

$$\tau_m \geq \sum_{t \in \mathcal{T}} y_{td}^m, \ d \in \mathcal{D}_m$$

$$\tau_m - \tau_m \leq r_m, \ m \in \mathcal{M}$$

$$\frac{\sum_{m \in \mathcal{M}} z_m}{\sum_{m \in \mathcal{M}} s_m} - 1 \leq \epsilon_1 \quad \text{for all } a \in \mathcal{A}$$

$$\frac{\sum_{m \in \mathcal{M}} s_m}{\sum_{m \in \mathcal{M}} \nu_m} - 1 \leq \epsilon_2 \quad \text{for all } a \in \mathcal{A}$$

$$y_{td}^m, z_m \in \{0,1\}$$

### 3 Numerical experiments

This new model will be tested for real slot request data for a medium-sized airport. Three aspects of the new model will be studied in particular. Firstly, the sensitivity of the model with respect to the range constraints (8). Secondly, the price of fairly distributing rejections. Finally, the price of fairly distributing schedule displacement. Compared to previous fairness measures not based on demand, we expect there to be an improved trade-off between fairness and total schedule displacement. The models will be solved using a branch-and-cut scheme with a commercial MILP solver.
Sets

\( \mathcal{A} \) \hspace{1cm} \text{set of airlines} \\
\( \mathcal{D} (\mathcal{D}_m) \) \hspace{1cm} \text{set of days (for which movement } m \text{ requested)} \\
\( \mathcal{M}(\mathcal{M}_a) \) \hspace{1cm} \text{set of movement requests (by airline } a \text{)} \\
\( \mathcal{P} \subset \mathcal{M} \times \mathcal{M} \) \hspace{1cm} \text{set of arrival-departure pairs } (m_a, m_d) \\
\( \mathcal{C} \) \hspace{1cm} \text{set of airport capacity constraints} \\
\( \mathcal{T} = \{1, \ldots, T\} \) \hspace{1cm} \text{set of coordination time intervals} \\

Parameters

\( t_m \) \hspace{1cm} \text{requested time for movement } m \\
\( \delta_c \) \hspace{1cm} \text{duration of constraint } c \\
\( p_a \) \hspace{1cm} \text{proportion of flights requested by airline } a \\
\( f_m^t \) \hspace{1cm} \text{displacement cost for assigning slot } t \text{ to movement } m \\
\( l_p \) \hspace{1cm} \text{turnaround time for movement pair } p \in \mathcal{P} \\
\( d_m^d \) \hspace{1cm} \text{indicates whether constraint } c \in \mathcal{C} \text{ is active on day } d \in \mathcal{D} \\
\( b_{m,c} \) \hspace{1cm} \text{contribution of movement } m \text{ to constraint } c \\
\( u_c^{ds} \) \hspace{1cm} \text{capacity for constraint } c \in \mathcal{C} \text{ on day } d \in \mathcal{D} \text{ at time period } s \\
\( r_m \) \hspace{1cm} \text{maximum range of slot times for request } m \\
\( \nu_m \) \hspace{1cm} \text{marginal cost of request } m \\

Decision variables

\( y_{m,d}^{td} \) \hspace{1cm} \text{indicates whether movement } m \text{ is assigned slot } t \text{ on day } d \\
\( z_m \) \hspace{1cm} \text{indicates if request for movement } m \text{ is rejected} \\
\( s_m \) \hspace{1cm} \text{schedule displacement for request } m \\
\( \tau_m(t_m) \) \hspace{1cm} \text{earliest (latest) slot time assigned to } m \\

Table 1: Notation used for single airport slot allocation model

References


1. Introduction

Warehouses are used to decouple supply from demand in supply chains. In order to make the warehouse operation more efficient, focus on the order picking process is a prime candidate, as it is the most labor-intensive process in the warehouse [1]. The efficiency of order picking which is defined as the number order lines picked within a given time period, depends largely on the storage assignment policy used. Random storage policy, randomly allocate products to the available space. It is very common in practice and has been widely studied in the literature ([2], [3] and [4]). Class-based storage (ABC) policy consider turnover frequency to classify the products into several classes (normally two or three) based their size and turnover. Then, the product class with the highest turnover and the smallest size is assigned to a block of storage locations (a storage zone) which is most accessible. However, within each zone, products of the same class are assigned randomly to the locations of the storage zone.

In the Full Turnover-Based (FTB) policy, products are stored based on COI (Cube per Order Index) proposed by [5], which determines how frequently a product is requested per unit of stock space required. In such storage assignment policies, the product attribute taken into consideration are turn over and size. Current storage assignment policies ignore information on product affinity, which is the frequency by which products are ordered jointly. In practice products in the orders are correlated and each order can consist of more than one order line. Some product combinations appear in the orders more often than the others and by identifying the affinity between the products, it is possible to cluster them based on their correlation. These clusters then can be allocated to the storage locations to reduce the order picking time. We propose an integrated cluster allocation (ICA) storage assignment model
that uses the information regarding the product turnover and the affinity between the products to assign products to the storage locations, specifically where each storage location (e.g. storage bin) consists of multiple sub-locations (such as sub-bins). Therefore a bin can be shared by multiple products, given the capacity of the bin and the inventory of the product.

2. ICA Mathematical model

Based on the affinity between the products and popularity of the products derived from a given order set, we propose a mathematical model that allocates product \( i \in P \) to the cluster at locations \( l \in L \), in order to minimize the total travel distance of the retrieval requests. We assume a unit load retrieval system. Additionally, the product inventory is not split up. The notations that are used are as following.

Parameters in the model:

- \( O \) The set of given orders from a certain period of time.
- \( L \) The set of available storage locations in the system.
- \( P \) The set of products in the assortment.
- \( C_l \) The number of sub-locations (sub-bins) available in cluster at location \( l \in L \).
- \( I_i \) The number of sub-bins needed to store the required inventory of product \( i \in P \).
- \( d_l \) The one-way distance from the I/O point to location \( l \in L \).
- \( s_{o \in l} \) \( s_{o \in l} = 1 \) if in order \( o \in O \) there is a request for product \( i \in P \), \( s_{o \in l} = 0 \) otherwise.

Variables in the model:

- \( x_{i \in l} \) \( x_{i \in l} = 1 \) if product \( i \in P \) is assigned to the cluster at location \( l \in L \), \( x_{i \in l} = 0 \) otherwise.
- \( t_{o \in l} \) \( t_{o \in l} = 1 \) if picking order \( o \in O \) requires a visit to location \( l \in L \), \( t_{o \in l} = 0 \) otherwise.

The proposed ICA mathematical model is as following:

\[
\begin{align*}
\text{min} & \quad \sum_{o \in O} \sum_{l \in L} d_l t_{o \in l} \\
\text{Subject to} & \quad \sum_{i \in L} x_{i \in l} = 1, \quad \forall l \in P \tag{2} \\
& \quad \sum_{i \in P} I_i x_{i \in l} \leq C_l, \quad \forall l \in L \tag{3} \\
& \quad \sum_{i \in P} I_i \leq \sum_{i \in L} C_i \tag{4} \\
& \quad t_{o \in l} \geq s_{o \in l} + x_{i \in l} - 1, \quad \forall i \in P, \forall o \in O, \forall l \in L \tag{5} \\
& \quad x_i = 0,1, \quad \forall i \in P, \forall l \in L \tag{6} \\
& \quad t_{o \in l} = 0,1, \quad \forall o \in O, \forall l \in L \tag{7}
\end{align*}
\]

The objective function (1) minimizes the total travel distance generated by trips to pick all orders. Constraint (2) ensures that each product is assigned to exactly one storage location. Constraints (3) and (4) take into account the capacity of each storage location and the whole
storage system, respectively. Constraint (5) guarantees necessary visits to storage locations of the requested products for all customer order. Constraints (6), (7) define the binary variables.

3. Numerical results

In order to show the benefits of the cluster-based policy in the storage system we consider a single-deep mini-load AS/RS consisting of one aisle with racks on both sides. The system dimension is square in time. Several assumptions are made. The horizontal speed is 2 m/s and the vertical speed is 0.5 m/s. A bin can consists of 2, 4 or 8 sub-bins. Inventory for the most popular products is the size of a half bin, and for less popular products, it is the size of a quarter bin (not possible for the bins with two sub-bins), and for the remaining products it is 1 sub-bin. Each storage location (slot) is 1 meter wide and 1 meter high. We take a data set from the warehouse science website [6] as the base example for the numerical experiment. Table 1 and shows the results for the base example.

Table 1-Time saving using the ICA model compared to FTB and ABC storage policy for the base example

<table>
<thead>
<tr>
<th>Nr. of Orders</th>
<th>Order size</th>
<th>Assortment</th>
<th>Nr. of sub-bins</th>
<th>Affinity</th>
<th>Exec. Time (s)</th>
<th>ICA-TT (s)</th>
<th>Gap (%)</th>
<th>FTBS-TT (s)</th>
<th>ABC-TT (s)</th>
<th>Savings ICA/FTBS</th>
<th>Savings ICA/ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>4</td>
<td>Low</td>
<td>188000</td>
<td>1265.25</td>
<td>9.9</td>
<td>1701.5</td>
<td>1990</td>
<td>25.6%</td>
<td>36.4%</td>
</tr>
</tbody>
</table>

To have a better insight of the model, in this table we vary five different parameters of our interest. The number of orders takes discrete values of 20, 30, 50, 100, 200. The order size has a discrete uniform distribution between 1 and 4. The assortment consists of 30, 50, 100, 200 or 400 products. The number of sub-bins can take the value of 4, 6 or 8. The affinity between products is either low, medium or high. In order to make it easier to solve the ICA model, we first linearized the model. We programmed the linear model in AIMMS and used Gurobi 6.5 to solve small instances of the ICA problem. A greedy construction heuristic solution is generated which is used as an initial solution for the solver. Table 2 shows the results for random instances. Exec. Time shows the execution time. FTBS-TT and ABC-TT show the travel time for FTB and ABC storage policies respectively. And the savings achieved by applying ICA model, compared to these two is given in next columns. The results are compared with the FTB and ABC storage system.

Table 2-Time saving using the ICA model compared to FTB and ABC storage policy for varying parameters

<table>
<thead>
<tr>
<th>Nr. of Orders</th>
<th>Order size</th>
<th>Assortment</th>
<th>Nr. of sub-bins</th>
<th>Affinity</th>
<th>Exec. Time (s)</th>
<th>ICA-TT (s)</th>
<th>Gap (%)</th>
<th>FTBS-TT (s)</th>
<th>ABC-TT (s)</th>
<th>Savings % ICA/FTBS</th>
<th>Savings % ICA/ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>U[1, 3]</td>
<td>200</td>
<td>4</td>
<td>Low</td>
<td>3602</td>
<td>57.25</td>
<td>1.85</td>
<td>79.75</td>
<td>115</td>
<td>28.2</td>
<td>50.2</td>
</tr>
<tr>
<td>30</td>
<td>U[1, 3]</td>
<td>200</td>
<td>4</td>
<td>Low</td>
<td>3601</td>
<td>103</td>
<td>3.67</td>
<td>144.25</td>
<td>199.75</td>
<td>28.6</td>
<td>48.4</td>
</tr>
<tr>
<td>50</td>
<td>U[1, 3]</td>
<td>200</td>
<td>4</td>
<td>Low</td>
<td>3600</td>
<td>198.5</td>
<td>6.01</td>
<td>293</td>
<td>409.5</td>
<td>32.3</td>
<td>51.5</td>
</tr>
<tr>
<td>100</td>
<td>U[1, 3]</td>
<td>200</td>
<td>4</td>
<td>Low</td>
<td>3600</td>
<td>500</td>
<td>11.64</td>
<td>720.75</td>
<td>853.5</td>
<td>30.6</td>
<td>41.4</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>200</td>
<td>4</td>
<td>Low</td>
<td>3603</td>
<td>1234.75</td>
<td>15.23</td>
<td>1549</td>
<td>1738.5</td>
<td>20.3</td>
<td>29.0</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 1]</td>
<td>300</td>
<td>4</td>
<td>Low</td>
<td>62</td>
<td>737</td>
<td>0</td>
<td>765.5</td>
<td>879</td>
<td>3.7</td>
<td>16.2</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 2]</td>
<td>300</td>
<td>4</td>
<td>Low</td>
<td>7203</td>
<td>962.75</td>
<td>1.78</td>
<td>1179.75</td>
<td>1388.5</td>
<td>18.4</td>
<td>30.7</td>
</tr>
</tbody>
</table>
4. Conclusion

The integrated cluster-based storage policy we proposed in this paper based on the affinity between products, contributes to more efficient assignment and delivers reduced order picking process. Results show that the ICA outperforms class-based and full-turnover based storage policies for products that would be stored in an automated storage and retrieval system and on average 33% and 22% savings can be achieved respectively. The proposed ICA storage policy can help fulfillment centers, warehouses and distribution centers to perform faster and reduce labor cost. Although ICA storage may have substantial benefits, depending on affinity, it is also more data intensive and dependent on order patterns.

References


<table>
<thead>
<tr>
<th>Unit</th>
<th>Category</th>
<th>Order Unit</th>
<th>Cost</th>
<th>Profit</th>
<th>Margin</th>
<th>Savings</th>
<th>Labor Cost Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>U[1, 4]</td>
<td>300</td>
<td>Low</td>
<td>28811</td>
<td>1604</td>
<td>20.62</td>
<td>2013.75</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>30</td>
<td>Low</td>
<td>7200</td>
<td>630.25</td>
<td>16.34</td>
<td>723.5</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>50</td>
<td>Low</td>
<td>7200</td>
<td>791</td>
<td>19.52</td>
<td>906</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>100</td>
<td>Low</td>
<td>7200</td>
<td>1018.25</td>
<td>15.76</td>
<td>1260</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>200</td>
<td>Low</td>
<td>7200</td>
<td>1234.75</td>
<td>15.23</td>
<td>1549</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>400</td>
<td>Low</td>
<td>7200</td>
<td>1412.25</td>
<td>16.24</td>
<td>1826</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1119.25</td>
<td>13.14</td>
<td>1474.25</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1361.5</td>
<td>22.7</td>
<td>1827.75</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>28812</td>
<td>10.57</td>
<td>1474.25</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>7202</td>
<td>1412.25</td>
<td>1826</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1119.25</td>
<td>13.14</td>
<td>1474.25</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1361.5</td>
<td>22.7</td>
<td>1827.75</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>28812</td>
<td>10.57</td>
<td>1474.25</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1119.25</td>
<td>13.14</td>
<td>1474.25</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>1361.5</td>
<td>22.7</td>
<td>1827.75</td>
</tr>
<tr>
<td>200</td>
<td>U[1, 3]</td>
<td>300</td>
<td>Low</td>
<td>7200</td>
<td>28812</td>
<td>10.57</td>
<td>1474.25</td>
</tr>
</tbody>
</table>
An integrated user-system approach for
shelter location and evacuation routing

Maria Paola Scaparra
Kent Business School, University of Kent, Canterbury, UK

Antonio Sforza, Claudio Sterle
Department of Information Technology and Electrical Engineering (DIETI),
University of Naples Federico II, Naples, Italy

Annunziata Esposito Amideo
Kent Business School, University of Kent, Canterbury, UK
Email: ae306@kent.ac.uk

1 Introduction

Shelter location and evacuation routing are crucial operations of Disaster Operations Management (DOM) ([1], [6]). A shelter is a facility where people experiencing perilous circumstances, due to either a natural or man-made disaster [14], can be supplied with services such as first-aid, drinkable water, food and beds. The aim of shelter location problems is to identify the best location of these facilities, given a pool of candidate sites, so as to maximise accessibility from the disaster affected zones (e.g., minimize the total travelling distance or time between affected zones and shelters). The process of people leaving their own houses and moving towards safe zones is named evacuation. Depending on aspects such as the transportation mode (e.g., foot, car, bus, train, boat, helicopter) and the disaster type (e.g., earthquake, flood, hurricane), different types of evacuation can occur [9].

The scope of evacuation routing problems is to optimally identify the evacuee routes so that the total evacuation time is minimized.

Over the years, researchers have developed various optimization models for shelter location (strategic decisions), car-based evacuation and bus-based evacuation (tactical decisions) as separate problems. However, there is little evidence of their combination, especially when it comes to consider the three problems in a unified framework ([5], [8]). Moreover, in order to mimic the evacuation process more accurately, other complicating aspects, such as disaster propagation, evolving infrastructure conditions, availability of shelters over time, and egalitarian policies [4], should also be incorporated into mathematical models. This work attempts at filling these gaps in the literature by tackling together the aforementioned aspects. To this aim, the Multi-Period Scenario-Indexed Shelter Location and Evacuation Routing (MP-SISLER) problem is defined through an original mixed integer
linear program, which extends and enriches the model introduced in [5]. The proposed formulation has been successfully solved by a MIP solver on small instances, but it requires the usage of ad-hoc exact and heuristic approaches ([10], [11], [12]) to deal with larger and real test cases. Hence, a branch-and-cut algorithm and an ad-hoc scenario generation based solution approach have been developed [3]. Preliminary results confirm the effectiveness of the proposed formulation and solution approaches to deal with this complex problem integrating both strategic and tactical decisions.

2 Problem Statement

We assume to plan for the evacuation of a calamity-prone region affected by a disaster over a certain time horizon. Three different categories of evacuees can be identified: self-evacuees (i.e., those who move autonomously) towards a shelter (SES), self-evacuees towards other destinations (SED), and supported evacuees (i.e., special-needs population who require assistance from public authorities) towards a shelter (SE). We focus on SES and SE that, in the following, will also be referred to as car-based evacuees and bus-based evacuees, respectively. SES and SE start evacuating from network points that are assumed to be the centroids of different evacuation zones. SES and SE share their final destination, which can be different types of shelters such as schools, community centres, and open spaces, each of them with specific features (e.g., capacities, relief supplies, staff members).

To account for disaster propagation, as in [7], we assume that road network conditions and, consequently, travelling times between evacuation zones and shelters are time-dependent. In particular, time-dependent travelling times allow to mimic the fact that, due to the dynamic disaster evolution, a connection may become either lengthier to travel (i.e., initial travelling time subjected to a finite delay) or no longer available (i.e., initial travelling time subjected to an infinite delay). Moreover, we define network disruption scenarios to capture disaster magnitude (i.e., under different disastrous circumstances, for the same time period, travelling times can be diverse). The demand for shelters and shelter availability are also considered to be time-dependent. Specifically, we assume that in each time period, a different percentage of evacuees abandons the disaster-prone zones and that resources to set up shelters become available over time.

To model the car-based evacuation and to account for an egalitarian allocation policy, we assume that SES should reach their final destinations within a given travelling time threshold which, where possible, could be their available shortest path. As for the bus-based evacuation, buses are stored in a depot and some are dispatched in each time period to accomplish supported-evacuation. Moreover, split-delivery of bus-based evacuees is allowed (i.e., more than one bus can pick up people from the same evacuation zone and bring them to different shelters). Finally, we also assume contraflow lane reversal to ease congestion over the network.

Given the above assumptions, the proposed optimization problem consists in identifying how many shelters should be opened and where as well as the routes of both car-based and
bus-based evacuees in each time period, over the entire time horizon, and for different network disruption scenarios.

### 3 MP-SISLER Problem Formulation

The objective of the MP-SISLER is to minimize the bus-based evacuation completion time, while assuring that car-based evacuees do not travel routes that are too lengthy over the entire time horizon. An interesting feature of our model is that it allows decision planners to identify a trade-off between *self-evacuation* and *supported evacuation oriented* solutions, by changing a parameter which represents the route length car-evacuees are willing to accept. This allows to balance the bus-based evacuation completion time objective and the car-based evacuation equity requirement. Figure 1 displays an example.

![Figure 1. Self-evacuation oriented solution (a) and supported evacuation oriented solution (b)](image)

Triangle, square, and round shapes represent, respectively, the depot, candidate shelter sites, and evacuation zone centroids. Selected shelters are marked with a cross and centroids are identified with the acronyms of the evacuees departing from there (i.e., SES, SE, and M = mixed demand, i.e., a combination of SES and SE). Normal and dashed arrow lines represent, respectively, SES assignments and SE routes. A tighter threshold (a) favours self-evacuation by inducing the opening of shelters close to car-based or mixed evacuation zones; if the threshold becomes looser (b), the supported evacuation completion time improves while self-evacuation becomes lengthier.

### 4 Conclusion

This work introduces a novel multi-period scenario-based model to optimize shelter location and evacuation routing decisions simultaneously. The MP-SISLER model can be deployed for either the preparedness or response phase of DOM. Given the uncertainty characterising disastrous events at the preparedness stage, it is paramount that multiple scenarios are captured into the modelling framework. During a disaster, in fact, some roads may become unavailable and the transport network changes over time as the disaster effects propagate in
the affected area. However, once the disaster has occurred (at the response stage), the scenario is identified and optimal solutions can be determined for that specific case. The proposed model and solution algorithms are applied to both testbed and real case instances. Detailed experimental results will be presented at the conference however, preliminary results demonstrate that the approach is able to find robust and efficient evacuation plans, thus providing policy-makers with a valuable decision support tool. A future development could be merging the MP-SISLER with a simulation approach so as to capture the dynamics of routing behaviors during an evacuation process in a more accurate manner ([2, 13]).

References


A multi-commodity location-routing problem in a maritime urban area

Teodor Gabriel Crainic¹, Massimo Di Francesco², Enrico Gorgone², and Paola Zuddas²

¹CIRRELT and Département de management et technologie, Université du Québec à Montréal, Canada
²Dipartimento di Matematica e Informatica, Università degli studi di Cagliari, 09134, Cagliari, Italy. Email: egorgone@unica.it

1 Introduction

Despite the very active research on location-routing problems, most of the studies investigated the single-commodity case [1, 2]. Limited attention has been devoted to the multi-commodity variant, which plays an important role in the context of City Logistics, because the demand of goods is highly customized [3, 4]. This work aims to fill this gap and investigates a multi-commodity location-routing problem defined in a City Logistics perspective in the case of a maritime urban area, where large-size vehicles enter in the center of the city from its port.

2 Problem statement

Consider a fleet of inbound containers at a port. Each container is filled with pallets, which have different destinations (or customers) in the city surrounding the port. Containers cannot be opened in the port because of the lack of space, and/or this operation is too costly or disallowed. Containers cannot enter many parts of the city because of narrow streets and environmental policies, to reduce the impact on city living conditions in terms of congestion, emissions, and pollution.

To deal with this problem, a two-tiered distribution structure is adopted: in the first tier, containers are moved by selected vehicles from the port to selected satellites, where pallets are transshipped to smaller and environment-friendly vehicles, which move pallets to their final destinations in the second tier. Therefore, a number of satellites is strategically located in the city and their selection comes at a cost, which is independent from the
workload performed in satellites. Specific vehicles are dedicated to each tier: they are referred to as container-compatible vehicles (CCVs) in the first tier and pallet-compatible vehicles (PCVs) in the second tier. Each vehicle of both tiers is distinct and its selection results in a fixed cost (independent from the distance covered after the selection). Yet, additional costs must be paid for each vehicle and pallet served in each selected satellite. In addition, the number of vehicles, containers and pallets served in each satellite must be lower than given capacities, which depend on its layout and its inner organization. In this study, CCVs are supposed to carry one container at a time in the first tier and each container is allowed to be unpacked at a satellite only. Moreover, the repositioning of empty containers unpacked at satellites is ignored and the time synchronization of vehicle operations is not considered. The routes of PCVs are open, because vehicles are not required to return to any satellite or the port. Moreover, splitting is allowed for all destinations of the second tier. To sum up, in this problem one is required to determine the location of satellites, the selection of CCVs and PCVs, the routes of PCVs from satellites to customers in the second tier, the assignment of containers from the port to satellites in the first tier and the flows of pallets from satellites to customers in the second tier.

3 Problem modeling

The problem is formulated on a network, where nodes represent the port, satellites and customers. Let $n_0$ be the port, $S$ the set of satellites and $\Gamma$ the set of customers. Consider a graph $G = (N, A)$, where the set $N$ of nodes is defined as $\{n_0\} \cup S \cup \Gamma$ and the set $A$ of arcs consists of the union of two subsets $A_1$ and $A_2$, which are associated with the first and the second echelon, respectively:

$$A_1 = \{(n_0, s) : s \in S\}$$
$$A_2 = \{(i, j) : i \in S \cup \Gamma, j \in \Gamma, i \neq j\}.$$

Figure 1 shows a possible network with one port, four satellites and five customers, as well as their possible connections. The arcs $A_1$ and $A_2$ are denoted by discontinuous and continuous lines, respectively. We propose an optimization model minimizing the costs of selection of satellites, those of selection of vehicles and their assignment to satellites, the costs of operations for each pallet handled at satellites, the transportation costs of containers and pallets, and vehicle routing costs in the second tier. Constraints enforce that:

- each container is picked up from the port and moved to a satellite by a CCV;
- the transit in each satellite of the pallets demanded by each customer;
Figure 1: A possible network

- each customer receives his/her freight moved by pallets;
- the flow balance of pallets at customers different from their final destination and the flow balance of PCVs;
- CCVs and a PCVs can be assigned to a satellite at most;
- the number of containers entering, pallets leaving and PCVs serving a satellite cannot be larger than predefined capacities, provided that the satellite is selected;
- the capacity of PCVs holds in the second tier.

4 Solution method

Since no connection is supposed to exist among satellites and routes are supposed to be open, this solution method is based on the idea that the routing problem in the second echelon can be decomposed by satellites, once containers and PCVs are assigned to satellites. Therefore, the overall problem can be divided into three subproblems:

1. A location-allocation problem, to select satellites and allocate each container to a satellite, while accounting for its capacity. This problem is denoted by $Prob_1$ and is solved by a MIP solver.

2. An assignment problem, to select and assign PCVs to satellites, such that each satellite receives sufficient transportation capacity for all pallets moved there according to the solution of $Prob_1$. This assignment problem is denoted by $Prob_2$ and can be easily solved by a solver like Cplex.

3. An open routing problem with splits for to deliver pallets from each satellite determined in the solution of $Prob_1$ by the PCVs determined in the solution of $Prob_2$. 
New heuristics will be proposed to solve Prob$_3(s)$ effectively, as the vehicle routing problems with splits and heterogeneous vehicles has be seldom investigated [5].

The subproblems are solved sequentially and included in an iterative procedure. In Prob$_1$ one not only minimizes the costs of satellite location and container allocation to satellites, but also a tailor-made approximation of the costs generated in the second echelon by the related allocation of containers to satellites. Next, Prob$_2$ and Prob$_3(s)$ are solved for each satellite and used to compute this approximation for the following iteration. Moreover, at the end each iteration we compute the objective function of the optimization model in Section 3 by the decision variables on satellite location and container allocation taken from Prob$_1$ and those on vehicle routing and pallet flows taken from Prob$_3(s)$, for each satellite $s \in S$ selected in Prob$_1$. The algorithm stops after a maximum execution time or a maximum number of iterations without any improvement in the objective function.

5 Conclusion

During the presentation, the problem motivating this study will be illustrated. The mathematical model will be presented and several instances with different characteristics will be used to make a proof of concept on the model. Finally, the solutions returned by the proposed solution method will be compared to those provided by a MIP solver. Future research paths will be also discussed.

References


