OPTIMIZATION ALGORITHMS FOR VEHICLE ROUTING AND PACKING PROBLEMS

Submitted in partial fulfillment of the requirements for

the degree of

DOCTOR OF PHILOSOPHY

in

Chemical Engineering

AKANG WANG

B.S., Chemical Engineering, Tianjin University, Tianjin, China
B.A., Finance, Nankai University, Tianjin, China

Carnegie Mellon University
Pittsburgh, PA

May, 2020

To my parents.

ACKNOWLEDGMENTS

The past five years at Carnegie Mellon University (CMU) have been the most intellectually stimulating experience for me to date. This was made possible by support and company of many people.

First and foremost, I would like to thank my thesis advisor, Prof. Chrysanthos E. Gounaris, for his guidance and support throughout my time at CMU. Besides strong technical skills, I'm also amazed by his persistence, attention to details, and high standards in research. I greatly appreciate that he has always treated me as an equal researcher and given me enough room to explore research topics. It was a great honor to work with him.

I would like to acknowledge my thesis committee members—Prof. Ignacio E. Grossmann, Prof. Nikolaos V. Sahinidis, Prof. Willem-Jan Van Hoeve, Prof. Alexandre Jacquillat, and Dr. Jeffrey E. Arbogast—for providing resourceful feedback on my thesis. Their comments definitely help to improve my work.

I acknowledge financial support from Air Liquide and Braskem throughout the Center of Advanced Process Decision-making. I'm also grateful to James C. Meade Graduate Fellowship and H. William and Ruth Hamilton Prengle Graduate Fellowship at CMU.

I had a wonderful experience in collaborating with several colleagues: Christopher L. Hanselman, Anirudh Subramanyam, Steffen J. Bakker, Jeffrey E. Arbogast, Gildas Bonnier, Zachary Wilson, Nicholas Ferro, Rita A. Majewski, Yuhao He, Victor Anselmo Silva, and Aliakbar Izadkhah. I learned insightful research ideas and valuable practical perspectives from academic and industrial collaborators. I was very fortunate to work with three master students: Aditya Pasari, Xiandong Li and Tushar Rathi.

I'm greatly indebted to former and current Gounaris group members. In particular, Anirudh Subramanyam, Nikolaos H. Lappas and Christopher L. Hanselman taught me C++ coding skills during my junior years. I also benefit significantly from discussions with other group mates: Natalie Isenberg, Hua Wang, Aliakbar Izadkhah, Xiangyu Yin, and William Strahl. One of the best moments during my time at CMU was having whiteboard discussions with them about classes, research, and beyond. Additionally, I'm very thankful to our lab managers, Nikolaos H. Lappas and Natalie Isenberg, for their immediate and effective technical support.

During my life in Pittsburgh, I had a great time hanging out with many interesting people: David Paul Molina Thierry, Saif Kazi, Rajarshi Sengupta, Eyan Peter Noronha, Carlos Jose Nohra Khouri, Connor Brem, Charles Sharkey, Dana McGuffin, Benjamin Sauk, Burcu Karagoz, Christina Schenk, Michael Radetic, Hector David Perez Parra, and David Bernal. I also had unforgettable memories with the Chinese community in the Chemical Engineering Department. In particular, I want to thank Wei Wan, Zixi Zhao, Yijia Sun, and Zicheng Cai for hosting festival parities; I had fun playing poker games with Yajun Wang, Yuhan Wen; I enjoyed Karaoke with Xiaoxiao Yu and Yun-Ru Huang.

A few non-academic friends I made at CMU are worth mentioning. Auntie Hsiao Wu, Uncle Tony Chen, and Uncle Tsu Huang provided me with authentic Chinese meals during the past a few years, which enriched my campus life a lot; Cindy and Mark Vicker held great conversations with me during the AIChE receptions; Janet Latini, Laura Shaheen, and Allyson Danley are always there with big smiles and ready to help. They are sincerely nice people; it was a great pleasure to know them.

Above all, I would like to thank my family in China, especially my parents, Mr. Shuiping Wang and Mrs. Hui'e Luo. Except for giving me love and support, they have instilled in me two core values: *responsibility* and *principle*. Without these, none of my past experiences could have been possible.

I coined the following sentence.

The best way to learn a topic is to discuss; the second best way is to write it in LATEX.

I have spent the past five years discussing research problems with colleagues, now it is time to formalize my thoughts and write them down.

Akang Wang
Pittsburgh, PA
May 4, 2020

ABSTRACT

Many optimization problems can be formulated as monolithic mathematical models (e.g., mixed-integer linear programs), which are ready to be solved by the state-of-the-art optimization solvers. However, by exploiting the problem structures, one may design specialized solution approaches that can manage to efficiently solve problems of a much larger size. In this thesis, we develop tailored branch-and-bound algorithms for solving complex problems that arise in supply chain logistics as well as object packing.

The primary focus of this thesis is on the mathematical modeling and algorithmic development for addressing vehicle routing problems. We consider the following four settings: (i) continuous-time inventory routing, which integrates inventory management, vehicle routing, and delivery-scheduling decisions; (ii) full truckload pickup and delivery, which entails full truckload shipments between distribution centers and delivery locations; (iii) marginal cost estimation, which seeks for the average routing cost of serving an individual customer in a stochastic supply chain system; (iv) routing under uncertainty, which incorporates customer demand and/or travel time uncertainty while designing delivery routes.

The second part of the thesis is focused on global optimization with application to cutting and packing problems. In particular, we are interested in two variants of packing problems: shape nesting and circle packing. The former entails irregular polygons while the latter involves circles. The goal is to identify a feasible configuration for these objects to be packed in a container such that no overlap exists among these objects and the material waste is minimized. Modeling both problems is simple, however, solving the resulting mathematical models is extremely challenging due to the presence of non-overlapping constraints.

For each of the aforementioned problems, we explore its intrinsic structure and propose a specialized algorithm. Through extensive computational studies on benchmark instances, we demonstrate the effectiveness and efficiency of our proposed methods.

PUBLICATIONS

The following papers related to the work presented in this thesis have been published, submitted, or are in preparation to be submitted in peer-reviewed journals:

1. **A. Wang**, C. L. Hanselman, and C. E. Gounaris, 2018. A customized branch-and-bound approach for irregular shape nesting. *Journal of Global Optimization*, 71(4), pp.935-955.

2. **A. Wang** and C. E. Gounaris, 2019. On tackling reverse convex constraints for non-overlapping of circles. *Under Review*. E-print available at http://www.optimization-online.org/DB_HTML/2019/12/7531.html.

3. **A. Wang**, J. E. Arbogast, G. Bonnier, Z. Wilson, and C. E. Gounaris, 2020. Estimation of marginal cost to serve individual customers. *Under Review*. E-print available at http://www.optimization-online.org/DB_HTML/2020/01/7573.html.

4. **A. Wang**, X. Li, J. E. Arbogast, G. Bonnier, and C. E. Gounaris, 2020. A novel branch-and-cut algorithm for continuous-time inventory routing. *Under Review*.

5. **A. Wang**, A. Subramanyam, and C. E. Gounaris, 2020. A branch-price-and-cut algorithm for robust vehicle routing under uncertainty. *In Preparation*.

6. **A. Wang**, N. Ferro, R. Majewski, Y. He, and C. E. Gounaris, 2020. Mixed-integer linear optimization for full truckload pickup and delivery. *Under Review*.

The following papers that will not be presented in this thesis have been published, submitted, or are in preparation in peer-reviewed journals:

1. A. Subramanyam, **A. Wang**, and C. E. Gounaris, 2018. A scenario decomposition algorithm for strategic time window assignment vehicle routing problems. *Transportation Research Part B: Methodological*, 117, pp.296-317.

2. S. Bakker, **A. Wang**, and C. E. Gounaris, 2019. Vehicle routing with endogenous learning: Application to offshore plug and abandonment campaign planning. *Under Review*.

3. V. A. Silva, **A. Wang**, V. J. M. Ferreira Filho, and C. E. Gounaris, 2020. Routing and scheduling of platform supply vessels in offshore oil and gas logistics. *In Preparation*.

CONTENTS

1 INTRODUCTION 1

1.1 Vehicle Routing Problems . . . 1

1.1.1 Problem Description . . . 1

1.1.2 Solution Approaches . . . 2

1.1.3 Challenges . . . 4

1.2 Packing Problems . . . 4

1.2.1 Problem Description . . . 4

1.2.2 Solution Approaches . . . 5

1.2.3 Challenges . . . 6

1.3 Aims and outline of the thesis . . . 6

2 A NOVEL BRANCH-AND-CUT ALGORITHM FOR CONTINUOUS-TIME INVENTORY ROUTING 8

2.1 Introduction . . . 8

2.2 Problem Definition . . . 12

2.3 Mathematical Modeling . . . 13

2.4 Tightening Techniques . . . 18

2.4.1 Tightening of Time Windows . . . 18

2.4.2 Variable Elimination . . . 18

2.4.3 Minimal # Visits . . . 19

2.4.4 Rounded Capacity Inequalities . . . 19

2.5 Computational Studies . . . 20

2.5.1 Benchmark instances . . . 21

2.5.2 Computational results on literature data . . . 22

2.5.3 Computational results on roadef instances . . . 25

2.6 Conclusions . . . 29

3 MIXED-INTEGER LINEAR OPTIMIZATION FOR FULL TRUCKLOAD PICKUP AND DELIVERY 31

3.1 Introduction . . . 31

3.2 Problem Definition . . . 33

3.3 Mathematical Modeling . . . 33
3.4 Computational Studies . . . 37
3.4.1 Model Performance . . . 37
3.4.2 Evaluating the pre-loading policy . . . 39
3.5 Conclusions . . . 41

4 ESTIMATION OF MARGINAL COST TO SERVE INDIVIDUAL CUSTOMERS 42
4.1 Introduction . . . 42
4.2 Problem Definition . . . 47
4.3 Proposed Framework . . . 48
4.3.1 Marginal Cost Estimation . . . 48
4.3.2 Bounding the Sample Size . . . 50
4.3.3 A General Framework . . . 51
4.4 Solving Routing Problems . . . 53
4.4.1 Set Partitioning Model . . . 54
4.4.2 Branch-Price-and-Cut Algorithm . . . 55
4.5 Computational Studies . . . 63
4.5.1 Evaluation of BPC Implementation Performance . . . 63
4.5.2 Marginal Cost Analysis . . . 67
4.6 Conclusions . . . 72

5 A BRANCH-PRICE-AND-CUT ALGORITHM FOR ROBUST VEHICLE ROUTING PROBLEMS UNDER UNCERTAINTY 75
5.1 Introduction . . . 75
5.2 Problem Definition . . . 80
5.3 Uncertainty Sets . . . 81
5.3.1 Cardinality-constrained sets . . . 81
5.3.2 Budget sets . . . 82
5.3.3 Factor models . . . 83
5.3.4 Ellipsoidal sets . . . 83
5.3.5 Discrete sets . . . 83
5.4 Polyhedral Studies . . . 84
5.5 Branch-Price-and-Cut . . . 87
5.5.1 Set Partitioning Model . . . 87
5.5.2 A Branch-Price-and-Cut Algorithm . . . 88
5.5.3 Pricing Subproblems . . . 89

5.6 Solution Approaches . . . 91
5.6.1 Robust Pricing Approach . . . 92
5.6.2 Robust Cutting-Plane Approach . . . 95
5.6.3 Comparison . . . 102
5.7 Computational Studies . . . 104
5.7.1 Computational Results on RCVRP Instances . . . 104
5.7.2 Computational Results on RVRPTW Instances . . . 109
5.8 Conclusions . . . 113
5.9 Appendix: Detailed Tables of Results . . . 114

6 A CUSTOMIZED BRANCH-AND-BOUND APPROACH FOR IRREGULAR SHAPE NESTING 121
6.1 Introduction . . . 121
6.2 Mathematical Modeling . . . 125
6.2.1 Nonconvex QCP Model . . . 125
6.2.2 Customized Model Relaxation . . . 127
6.3 The New Algorithm . . . 131
6.3.1 Feasibility Checking . . . 133
6.3.2 Branching Rule Selection . . . 135
6.3.3 Feasibility-based Node Tightening . . . 136
6.3.4 Tighter Variable Bounds . . . 138
6.3.5 Symmetry Breaking Constraints . . . 138
6.4 Computational Studies . . . 139
6.4.1 Instances . . . 139
6.4.2 Fixed Orientation Case . . . 140
6.4.3 Free Rotation Case . . . 143
6.5 Conclusions . . . 145
6.6 Appendix: Nomenclature . . . 146

7 ON TACKLING REVERSE CONVEX CONSTRAINTS FOR NON-OVERLAPPING OF CIRCLES 148
7.1 Introduction . . . 148
7.2 Problem Definition . . . 152
7.3 Solution Approach . . . 152
7.3.1 Customized Model Relaxation . . . 152
7.3.2 The Branch-and-Bound Algorithm . . . 154

7.4 Strengthening Techniques 156
7.4.1 Intersection Cuts 156
7.4.2 Feasibility-based Tightening 163
7.4.3 Implementation Details 165
7.5 Computational Studies 166
7.5.1 Circle Packing 167
7.5.2 Computational Results 169
7.6 Conclusions 173

8 CONCLUSIONS AND FUTURE WORK 175
8.1 Contributions 175
8.2 Future Directions 179

BIBLIOGRAPHY 182

LIST OF TABLES

Table 1.1 Comparison between arc-based and route-based modeling frameworks 3

Table 2.1 Comparison among `LBS20` and our proposed algorithms on 90 benchmark instances . . . 24

Table 2.2 Detailed results for `LBS20` and our Branch-and-Cut algorithm on 4 benchmark instances . . . 25

Table 2.3 Detailed results for our Branch-and-Cut algorithm on 45 `clustered` instances . . . 26

Table 2.4 Detailed results for our Branch-and-Cut algorithm on 45 `random` instances . . . 27

Table 2.5 Computational results for our Branch-and-Cut algorithm on 63 `roadef` instances . . . 28

Table 2.6 Detailed results for our Branch-and-Cut algorithm on 63 `roadef` instances . . . 30

Table 3.1 Computational results for our MILP formulation on 66 real-life instances . . . 38

Table 3.2 Computational results for evaluating the pre-loading policy . . . 41

Table 4.1 Computational results for our BPC algorithm on 25-, 40-, and 70-customer `random` instances . . . 66

Table 4.2 Computational results for our BPC algorithm on 25-, 40-, and 70-customer `CGL` instances . . . 66

Table 4.3 Marginal cost estimation under three customer location cases . . . 71

Table 4.4 Marginal cost estimation under three customer demand levels . . . 73

Table 5.1 Closed-form expressions, time and storage complexities for computing the worst-case load of a vehicle route . . . 100

Table 5.2 Applicability of BPC algorithms to different uncertainty sets . . . 103

Table 5.3 Computational results for BPC algorithms on 90 RCVRP ($\mathcal{Q}_G$) instances . . . 107

Table 5.4 Computational results for BPC algorithms on 90 RCVRP ($\mathcal{Q}_B$) instances . . . 108

Table 5.5 Computational results for the BPC algorithm on 90 RCVRP ($\mathcal{Q}_F$ / $\mathcal{Q}_E$ / $\mathcal{Q}_D$) instances . . . 109
Table 5.6 Computational results for BPC algorithms on 90 RVRPTW ($\mathcal{Q}_G$) instances . . . 111
Table 5.7 Computational results for BPC algorithms on 56 RVRPTW ($\mathcal{T}_G$) instances . . . 112
Table 5.8 Computational results for BPC algorithms on 56 RVRPTW ($\mathcal{Q}_G \times \mathcal{T}_G$) instances . . . 113
Table 5.9 Detailed results for our BPC algorithm on RCVRP instances (A) . . . 115
Table 5.10 Detailed results for our BPC algorithm on RCVRP instances (B) . . . 116
Table 5.11 Detailed results for our BPC algorithm on RCVRP instances (E, F and M) . . . 117
Table 5.12 Detailed results for our BPC algorithm on RCVRP instances (P) . . . 118
Table 5.13 Detailed results for our BPC algorithm on RVRPTW instances (C1, R1 and RC1) . . . 119
Table 5.14 Detailed results for our BPC algorithm on RVRPTW instances (C2, R2 and RC2) . . . 120
Table 6.1 Parameters used in the new algorithm. . . . 140
Table 6.2 Comparison between *QP-Nest* and the new algorithm for the case of fixed orientation, using a naive solution as incumbent. . . . 141
Table 6.3 Comparison between *QP-Nest* and the new algorithm for the case of fixed orientation, using an optimal solution as incumbent. . . . 142
Table 6.4 Comparison between *QP-Nest* and the new algorithm for the case of free rotation, using a naive solution as incumbent. . . . 143
Table 6.5 Comparison between *QP-Nest* and the new algorithm for the case of free rotation, using an optimal fixed-orientation solution as incumbent. 144
Table 7.1 Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a circle . . . 171
Table 7.2 Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a rectangle . . . 171
Table 7.3 Computational results for global solvers on solving instances of packing circles into a circle . . . 172
Table 7.4 Computational results for global solvers on solving instances of packing circles into a rectangle . . . 172

LIST OF FIGURES

Figure 2.1 Log-scaled performance profiles across all benchmark instances. The left graph compares the performance in `clustered` instances, while the right graph compares the performance on `random` instances. For each curve, the value at $t = 0$ gives the fraction of benchmark instances for which it is fastest, while the limiting value at $t \to \infty$ gives the fraction of instances which it could solve within the time limit of 10 hours. 28

Figure 4.1 A scenario-sampling framework 52

Figure 4.2 The Branch-Price-and-Cut algorithm. 56

Figure 4.3 Example progress of the average marginal cost, as the sample size N increases 69

Figure 4.4 Effect of a customer's proximity to the depot facilities on the marginal routing cost. The proximity is defined as the length of the direct round-trip route from the depot that is closest to the target customer. 70

Figure 4.5 Effect of a customer's demand level on the marginal routing cost . 72

Figure 6.1 Approximation of a polygon's area via a set of inscribed circles. . . 125

Figure 6.2 Identifying the largest circle within the area where two polygons overlap. 127

Figure 6.3 The *QP-Nest* approach. 128

Figure 6.4 The trigonometric constraint and its dynamically tightened relaxation. 130

Figure 6.5 The non-overlapping constraint and its dynamically tightened relaxation. 132

Figure 6.6 A customized BB algorithm for solving the Nesting Problem. . . . 134

Figure 6.7 Feasibility-based tightening (special case when $F_{p\tilde{m}q\tilde{n}} = \mathbb{R}^2$) 137

Figure 6.8 Feasibility-based tightening (general case when $F_{p\tilde{m}q\tilde{n}} \subset \mathbb{R}^2$). 137

Figure 6.9 Polygon shapes used in our benchmark study. 140

Figure 6.10 A naive solution. 141

Figure 6.11 Optimal solutions for representative six-polygon (left) and seven-polygon (right) instances under fixed orientation. 142

Figure 6.12 Optimal solutions for several five-polygon instances under free rotation. 144
Figure 7.1 The non-overlapping constraint and its dynamically tightened relaxation (adapted from Chapter 6). 155
Figure 7.2 Generation of intersection cuts 159
Figure 7.3 Valid convex sets $\mathcal{S}_{ij}$ for generating intersection cuts 161
Figure 7.4 Strengthened intersection cuts 161
Figure 7.5 Feasibility-based tightening 164
Figure 7.6 Two variants of circle packing problems 167
Figure 7.7 Performance profiles across all benchmark instances of each problem variant. In both graphs, "This work" refers to the performance of our proposed algorithm using strengthened intersection cuts and feasibility-based tightening (BB+SIC+FBT). For each curve, the value at $t = 0$ provides the fraction of benchmark instances for which the corresponding solver/algorithm is fastest, while the limiting value at $t \to \infty$ provides the fraction of instances that could be solved within the time limit of 1 hour. 174

1

INTRODUCTION

1.1 VEHICLE ROUTING PROBLEMS

1.1.1 *Problem Description*

We borrow from [167] the following definition of the *family of vehicle routing problems*.

> *Given a set of transportation requests and a fleet of vehicles, the goal is to determine a set of vehicle routes to perform all (or some) transportation requests with the given vehicle fleet at minimum cost; in particular, it is to decide which vehicle handles which requests in which sequence so that all vehicle routes can be feasibly executed.*

In this type of problem, subsumed under the term *vehicle routing problem* (VRP), transportation *requests* usually denote the service of moving goods from distribution centers to customers (or vise versa, from suppliers to collection centers), which is performed by a fleet of vehicles that are initially located at one or more depots; the *feasibility* of vehicle routes are defined by operational constraints such as vehicle capacity restrictions, customer-visiting time windows, route duration limits, just to name a few; the *cost* can be monetary, distance, time, or otherwise.

The simplest VRP is the *capacitated vehicle routing problem* (CVRP), in which given a distribution center, a fleet of homogeneous vehicles, and a group of customers with known demands to be served, one aims to identify the minimum-cost routes for vehicles to traverse, such that customer demands are satisfied and vehicle capacity constraints are respected. There are numerous VRP variants, each of which is featured by one or more practical considerations such as heterogeneous fleets, time windows, multiple depots, multiple trips, multiple periods, among others [171]. Except for operational efficiency and economy, other factors are also considered as performance metrics. The service quality, e.g., driver consistency [85], can help businesses to differentiate themselves from

their competitors and is thus receiving more and more attention when designing routes. Due to the negative environmental impact from greenhouse gas emission by vehicles, some literature considered the objective of VRPs to be not necessarily economic but environmental or a mixture of both [69]. In recent years, the trend of integrating routing designs with other operational decisions, e.g., inventory and production management [53], has been observed. To sum up, the VRP is a "rich" family of transportation problems with various practically relevant attributes.

The VRP has a broad range of industrial applications. Since the seminal work [61] introduced the VRP as a real-world application concerning the delivery of gasoline to gas stations, many real-life problems spanning various industries, such as liquified natural gas transportation [87], waste collection [105], drone delivery [137], maritime surveillance [70], offshore wind farm maintenance [97], meal delivery [175], among others, have been modeled and tackled from a VRP perspective. Furthermore, though vehicle routing decisions are fundamentally operational, they are often linked with other decisions taken at a strategic or tactical level over a longer planning horizon, such as the fleet sizing and composition [93], facility location [151], to just name a few. We can claim that the VRP is one of the key components of supply chain logistics in practice.

1.1.2 *Solution Approaches*

The VRP is a generalization of the well-known *traveling salesman problem* (TSP) [14]. Since the TSP itself is $\mathcal{NP}$-hard, so is the VRP. In another word, no polynomial-time algorithms exist for addressing the VRP. Since the seminal work [61], lots of efforts have been devoted to the development of heuristic and exact solution algorithms.

– Heuristic algorithms. For the past a few decades, a wide variety of *constructive* and *improvement heuristics* have been proposed, culminating in recent years with the development of powerful *metaheuristics* capable of computing within a few seconds solutions whose value usually lies within less than one percent of the best known values. The constructive heuristics (e.g, the savings algorithm [50]) are usually employed to provide a starting solution to an improvement heuristic while classic improvement heuristics usually perform intra-route and inter-route moves efficiently. There are mainly two types of metaheuristics: (i) neighborhood-centered methods (e.g., tabu search and variable neighbor search), which generally proceed by iteratively exploring the neighborhoods of a single incumbent solution; (ii) population-based strategies (e.g., ant colony and genetic algorithms), which evolve

Table 1.1: Comparison between arc-based and route-based modeling frameworks

	Arc-based modeling	Route-based modeling
# variables	polynomial	exponential
Feature	easy to enforce addition constraints	tight LP relaxations
Algorithm	branch-and-cut	branch-price-and-cut

a set of solutions by generating one of several "new" solutions out of combinations of existing ones. We refer readers to [112] for a succinct survey on this topic.

– Exact algorithms. The traditional wisdom of modeling VRPs is through defining the selection of an arc between two nodes as a binary variable, yielding an *arc-based* formulation (e.g., *two-index vehicle flow* model [111]). The resulting mathematical model is an mixed-integer linear program (MILP) with polynomially many variables, hence it is solved by the *branch-and-cut* method. Recently, the *route-based* modeling idea (e.g., *set partitioning* model [28]) is becoming more appealing due to its tight linear programming (LP) relaxation. Specifically, it treats the choice of a feasible route as a binary variable. Due to the existence of exponentially many feasible routes, the new modeling framework yields an MILP with a huge number of variables. This necessitates the *branch-price-and-cut* algorithm, in which the LP relaxations at every node in the branch-and-bound tree are solved via *column generation* while cutting planes are added to strengthen the relaxations. We refer readers to [57] for a comprehensive review on the branch-price-and-cut algorithm. Both modeling frameworks have their strengths and weaknesses, as Table 1.1 shows. The arc-based framework is more adaptive in terms of enforcing additional operational constraints while the route-based one induces mathematical models with tighter LP relaxations and is thus usually computationally more efficient. We remark that the latter can be considered as a result from the *Dantzig-Wolfe decomposition* of the former.

Heuristic algorithms are usually effective, efficient, and scalable in a sense that they can provide high quality feasible solutions efficiently even for practical problems of a large size. By contrast, exact algorithms possess the unique value of being able to quantify the quality of returned feasible solutions and to provide guarantee of optimality. The work of [169] has demonstrated the state-of-the-art branch-price-and-cut algorithms could optimally solve CVRP instances of up to 400 customers within hours, while the heuristics proposed in [89]

could identify solutions of the same quality within seconds, albeit without awareness of their optimality.

1.1.3 *Challenges*

Though the VRP is one of the most well-studied combinatorial problems in the literature, there is still considerable research and practical relevance for continuation of exploring this topic. We identify the following three challenges that will arise when developing exact solution approaches for addressing VRPs.

- From a practical perspective, many real-life logistics problems have increasingly complex operational constraints that need to be incorporated while designing routes. Modeling these practical considerations is becoming burdensome.
- Mathematical models for VRPs are notoriously difficult to solve. Though significant advances have been made to improve the commercial solvers' capability of tackling these MILPs, the deployment of customized solution algorithms is still essential for the success of tackling VRPs. For branch-and-cut algorithms, developing problem-specific strengthening techniques is still badly needed; for branch-price-and-cut algorithms, the most critical issue is how to efficiently solve pricing subproblems in the column generation step.
- The information necessary to formulate the relevant routing problems is not always known with precision due to the presence of uncertainty affecting the parameters of the problem. For example, the work of [78] has identified three types of uncertainty sources that are commonly found in VRP applications: customer demands, travel/service times, and customer orders. The big challenge is how to model routing problems when uncertainty intrinsically exists and how to solve the resulting mathematical models.

1.2 PACKING PROBLEMS

1.2.1 *Problem Description*

The packing problem is a class of optimization problems that involve attempting to pack objects together into one or more containers. The goal is to either pack a single container as densely as possible or pack all objects using as few containers as possible. In the literature,

this problem is also known as the *cutting and packing problem*, since carving out desired shapes from the raw material is analogous to packing them within the equivalent container. The most well-known packing example is the *bin packing problem*, where items of different volumes must be packed into a finite number of bins each with a fixed given volume in a way that minimizes the number of bins used. A popular two-dimensional packing problem is the *strip-packing problem*, which aims to cut standard-sized pieces of stock material, such as paper rolls or sheet metal, into rectangles of specified sizes while minimizing material wasted. The objects to be packed are not necessarily standardized like rectangles but can be irregular polygons, giving rise to the *nesting problem* [81]. This problem is a rather general two-dimensional cutting and packing problem where the shapes to be packed can be different to each other, irregular and non-convex, and may possibly contain holes. The shapes, which are usually represented (approximated to arbitrary precision) by polygons, are to be packed in a stock that comes in the form of a fixed-width rectangular sheet, whose length is to be minimized. Furthermore, one may be interesting in packing circles rather than polygons. The *circle packing* problem is an archetypal family of problems that comes in many variants, such as packing identical circles into a rectangular container with the objective of minimizing the container's area [90], or identifying the minimal radius of a circle within which other circles can simultaneously be placed [149], among others. Readers are referred to [173] for a topology of cutting and packing problems.

The family of packing problems has numerous industrial applications. For example, obtaining tight nesting solutions is of great practical importance when cutting metal parts for automobiles, airframes and other machinery, as well as cutting leather and fabrics for apparel and upholstery applications. Related circle packing applications include container loading, cyclinder packing and wireless communication network layout, to name but a few [46].

1.2.2 *Solution Approaches*

The packing problem is generally $\mathcal{NP}$-hard. For example, the nesting problem generalizes the two-dimensional bin packing problem [30] that is $\mathcal{NP}$-hard; the work of [63] showed that the circle packing problem is $\mathcal{NP}$-hard. Stated differently, no polynomial-time algorithms exist for solving the packing problem. In the literature, the vast majority of efforts have been channeled into the development of heuristic algorithms. For example, numerous metaheuristics (e.g., genetic algorithm [45], tabu search [115, 116]) were proposed to solve bin packing problems; genetic algorithms [41], GRASP [10], and tabu search [44]

were presented for the nesting problem. By contrast, the research on exact algorithms for addressing packing problems is scarce, in particular for those that cannot be simply modeled as monolithic MILPs. For example, the only exact approach for addressing the nesting problem is from [100], which entails solving non-convex quadratically constrained programs iteratively.

1.2.3 *Challenges*

To solve the packing problem exactly will raise concerns from both modeling and algorithmic perspectives.

- From a modeling perspective, enforcing no overlap between two objects is generally non-trivial, especially for irregular shapes, e.g., polygons.
- Introducing non-overlapping constraints often induces a non-convex feasible region, as a result, the mathematical model becomes a global optimization problem that is notoriously challenging to solve.

1.3 AIMS AND OUTLINE OF THE THESIS

The overarching aim of this thesis is to develop mathematical models and solution algorithms for vehicle routing problems as well as packing problems so as to push the boundaries of their tractability. To that end, we exploit the problem structures and present customized solution approaches that are either comparable or superior to the state-of-the-art methods in terms of computational performance.

The remainder of this thesis is organized as follows. In Chapter 2, we present a novel arc-based MILP to model the inventory routing problem occurring in industrial gas distribution. To solve the resulting formulation more efficiently, we propose a tailored branch-and-cut algorithm that incorporates problem-specific strengthening inequalities. In Chapter 3, we are interested in the full truckload pickup and delivery problem faced by the petrochemical company Braskem. This problem has several routing and scheduling attributes of practical relevance, such as multiple pickup points, optional orders, loading dock capacity restrictions, among others. We formulate it as an arc-based MILP that demonstrates satisfactory performance in solving real-life instances. In Chapter 4, we focus on a strategic problem faced by the industrial gas company Air Liquide. This problem is to quantify the expected marginal cost of serving individual customers in a stochastic

supply chain system. It often arises when the distributor wants to estimate the customer lifetime value, in order to distinguish profitable customers. We propose a scenario-sampling approach that combines classic probability theory with exact vehicle routing techniques and thus provides statistical insurance for the estimation accuracy. To solve the routing problem of our interest, we develop a customized branch-price-and-cut algorithm. In Chapter 5, we turn our attention to VRPs under demand/travel time uncertainty, which is ubiquitous in supply chain and logistics. We model this problem from a robust optimization perspective and present a novel approach that embeds robust cutting planes into a branch-price-and-cut algorithm. The unique feature of our proposed approach is that it can handle various, general types of uncertainty sets. Chapter 6 and Chapter 7 are concerned with exact solution algorithms for addressing shape nesting and circle packing problems, respectively. Optimizing these problems is relevant when carving out desirable shapes from stock materials, such as aluminum rolls or leather sheets. We enforce no overlap among objects via circle-circle non-overlapping constraints. From a geometric perspective, we present a convexification strategy for these non-overlapping constraints. Furthermore, reverse convexity-based and feasibility-based tightening techniques are proposed and embedded into a customized branch-and-bound algorithm. Chapter 8 summarizes the contributions of this work in addition to proposing several future research directions.

2

A NOVEL BRANCH-AND-CUT ALGORITHM FOR CONTINUOUS-TIME INVENTORY ROUTING

This chapter is focused on the continuous-time inventory routing problem. Inventory management, vehicle routing, and delivery-scheduling decisions are simultaneously considered in the context of inventory routing. The continuous-time feature requires that the distributor has to both monitor inventory levels at customers and make product replenishment decisions in continuous time, so as to ensure that stock levels are maintained within the desired intervals at any moment of the planning horizon. In this chapter, we develop a compact mixed-integer linear programming formulation to model the continuous-time inventory routing problem. To expedite the solution process, we propose various tightening techniques, including the adaption of well-known rounded capacity inequalities, among others. We implement a branch-and-cut algorithm to solve the model optimally. Through extensive computational studies on a suite of 90 benchmark instances from the literature, we show that our branch-and-cut algorithm significantly outperforms the state-of-the-art approach. The computational studies on 63 real-life instances demonstrate our algorithm's practical value of solving instances with up to 20 customers.

2.1 INTRODUCTION

The Inventory Routing Problem (IRP) integrates inventory management, vehicle routing, and delivery-scheduling decisions. The IRP arises in the context of *vendor-managed inventory* [174], where a supplier monitors inventory levels at customers and makes the replenishment decisions for products delivered to customers. This is often a win-win strategy for both suppliers and customers: suppliers can coordinate shipment made to customers so as to save distribution costs, while customers can benefit from avoiding efforts for explicit inventory control. The integration of inventory control and delivery planning is being practiced in industry. For example, the industrial gas company Air Liquide has

pointed out that its business model is transitioning from customer-managed inventory to vendor-managed inventory, which necessitates inventory routing optimization techniques. The ROADEF/EURO challenge 2016 [8] was even dedicated to a real-life inventory routing project in Air Liquide. Hence, developing optimization techniques for inventory routing is of great interest for both academic research and industrial applications.

There are many IRP variants and they can be categorized based on different criteria. The first one is the inventory replenishment policy. Replenishment policies define pre-established rules to be imposed on the quantity delivered to a customer in each visit. There are two common policies: order-up-to-level policy and maximum-level policy. The order-up-to-level policy is to fill the inventory to the tank capacity whenever a customer is replenished [17], while the maximum-level policy allows for flexible replenishment and only requires that tank capacities are respected [22, 64]. Clearly, the maximum-level policy provides more flexibility for replenishment decisions and is thus more preferable from the cost-saving perspective. The second criterion is the objective function. Traditionally, the objective of an IRP is to minimize the total cost, including inventory cost and routing cost [17, 22, 64]. Recently, some research effort [2, 16, 27, 88, 104, 153, 158] has been focused on minimizing the so-called "logistic ratio", which is the ratio of the total cost to the amount of product distributed. In a short planning horizon, it is not incentivized to deliver more product than necessary because the objective is to reduce the distribution cost. However, the logistic ratio is considered to be a better metric of evaluating distribution policies in the long term. In fact, it is a common industrial practice that distributors assess the efficiency of a distribution policy through measures that consider both the distribution cost and the amount of product distributed. Unfortunately, choosing the ratio function as an objective results in a fractional programming model [140], which is usually much harder to solve. Interested readers are referred to the work of [52] for a comprehensive review on the IRP and its variants.

A new perspective to categorize the IRP is whether to discretize the planning horizon. The discrete-time IRP first discretizes a given time horizon into multiple time periods and then assumes that customers receive their deliveries at the beginning of each period and can use them to fulfill their demands in that period [64]. This assumption simply considers both vehicle routing and inventory replenishment to be instantaneous. In this case, one only needs to enforce inventory constraints at the beginning/end of each time period. In contrast, the continuous-time IRP (CIRP) explicitly accounts for the time of distributing product by vehicles and manages inventory in continuous time. In particular, it requires that inventory constraints are satisfied at any time point within the planning

horizon [108]. In such a case, vehicle departure/arrival times at customers cannot be neglected but should be properly accounted for, in order to monitor inventory levels continuously. Clearly, in the case of CIRP, vehicle routing is intertwined with inventory management in a much more complicated way, and thus the CIRP is significantly more challenging than the discrete-time IRP.

The IRP is notoriously hard because three types of decisions have to be made simultaneously: (i) when to serve a given customer, (ii) how much to deliver to a customer when the latter is served, and (iii) how to combine customers into vehicle routes. In the literature, the discrete-time IRP has been extensively studied. The work of [51] considered the multi-vehicle IRP with and without consistency requirements, and applied an adaptive large neighborhood search scheme in which some subproblems are solved exactly as mixed-integer linear programs. The first exact algorithm for IRP is from the work of [17] in which the authors considered a single-vehicle IRP and proposed a branch-and-cut approach that could solve, to optimality, benchmark instances with up to 50 customers and 3 time periods. The work of [54] and [55] extended the branch-and-cut approach from [17] to the multi-vehicle IRP and enhanced it with new valid inequalities. Their algorithm obtained improved upper and lower bounds for many IRP instances. One of the most successful exact approaches for solving the discrete-time IRP was presented in [64], where the authors proposed a sophisticated branch-price-and-cut algorithm, solving benchmark instances with up to 50 customers and up to 5 vehicles optimally. In the recent work [22], single-period tightening cuts called "disjoint route inequalities" were proposed and incorporated into a branch-and-cut algorithm. The computational results show that the new branch-and-cut algorithm returns gaps that are significantly smaller than the algorithms in [55] and [64].

The CIRP, notwithstanding its significance, has received very little attention in the literature. The work of [67] considered the IRP with driver rest constraints and decomposed it into an upper-level routing problem and a lower-level continuous-time scheduling problem. An iterative approach based on the upper and lower levels was then presented to identify the optimal routing and scheduling design. The work of [76] focused on the supply-driven cyclic CIRP in which inventory is held in containers that act as both a storage container and a movable transport unit, in the application of biogas transportation. The authors proposed a tour-based formulation that could optimally solve instances of up to 7 customers. Another approach was put forth in the work of [108]. The authors first proved that when the CIRP instance data includes only rational numbers, there exists a sufficiently fine time discretization such that both routing and inventory replenishment decisions

made in the optimal solution will always happen at discretizated time points. Based on this theory, the authors then proposed a *dynamic discretization discovery* algorithm: (i) the planning horizon is first discretized with different resolutions; (ii) for each discretization, after rounding up and down the adjusted travel times, two mixed-integer linear programs are formulated to produce an upper bound and a lower bound, respectively; (iii) if the difference between the best upper and lower bounds is below a pre-specified tolerance, then the optimality of a CIRP is achieved. The novelty in the dynamic discretization discovery approach is that it discovers exactly which times are needed to obtain an optimal, continuous-time solution by solving a sequence of relatively easy mixed-integer linear programs. The authors generated 90 benchmark instances with up to 15 customers and conducted extensive computational studies on them. Their computational results show that the dynamic discretization discovery approach could solve 26 of them to optimality.

In this chapter, we consider the CIRP defined in [108]: (i) the maximum-level inventory replenishment policy is adopted; (ii) the objective is to minimize the total routing cost; (iii) inventory levels are monitored and controlled continuously. As a result, the CIRP has several complicated characteristics. First, the planning horizon becomes a single but relatively long time period and thus vehicles are now allowed to perform multiple trips. Second, a customer's total product consumption across the whole planning horizon may exceed the tank capacity, then split delivery to a customer and thus multiple visits at this customer will happen. Last, both vehicle routing designs and delivery-scheduling decisions are now constraining the inventory control space, since they are all taken into account explicitly in time. Though the branch-price-and-cut approach proposed by [64] was quite successful in solving the discrete-time IRP, it cannot be easily extended to the CIRP due to complications from having to model features like continuous arrival/departure time, multiple vehicle trips, multiple customer visits, among others. Furthermore, even formulating the CIRP as a mixed-integer linear program and solving it via a branch-and-cut approach are not a straightforward extension from the discrete-time case. In this work, we duplicate nodes and arcs to represent routing features like multiple visits at a customer and multiple trips by a vehicle, respectively. We model the vehicle arrival/departure times at customers as continuous variables and manage stock levels in continuous time. Based on these novel modeling ideas, we then propose a compact mixed-integer linear programming (MILP) formulation for solving the CIRP. We also adapt the well-known *rounded capacity inequalities* (RCI) to strengthen linear programming (LP) relaxations. Our branch-and-cut algorithm is also enhanced with other tightening techniques, such as variable elimination.

The distinct contributions of our work can be summarized as follows.

- We propose a novel mixed-integer linear programming formulation to model the continuous-time inventory routing problem. This formulation incorporates several ingenious modeling ideas to handle the multi-trip, multi-visit features as well as continuous-time inventory management.
- We propose various types of tightening techniques to strengthen the linear programming relaxations. In particular, we adapt rounded capacity inequalities into our model and develop protocols to dynamically separate and add them during the branch-and-cut process.
- We conduct extensive computational studies on 90 benchmark instances from the literature and the computational results show that our branch-and-cut algorithm significantly outperforms the state-of-the-art approach. In particular, our algorithm could solve 56 of them to optimality within a reasonable amount of time and return a small residual gap for the remaining ones, while the state-of-the-art algorithm could only solve 26 out of 90 to guaranteed optimality.
- We further evaluate our algorithm on newly generated benchmark instances that are inspired by the real-life data from ROADEF/EURO Challenge 2016. Out of 63 benchmark instances, our branch-and-cut algorithm solved 56 of them to optimality, including a few 20-customer instances. This further demonstrates our proposed algorithm's efficiency.

The remainder of the chapter is organized as follows. A formal problem definition is given in Section 7.2. In Section 3.3, we present a novel mixed-integer linear programming formulation to model the CIRP. We then propose several tightening techniques that can expedite the solution process in Section 2.4. Section 7.5 presents detailed computational results on the performance of our branch-and-cut algorithm. Finally, we conclude our work in Section 7.6.

2.2 PROBLEM DEFINITION

The continuous-time inventory routing problem is defined on a directed graph $G = (V, E)$ where $V := \{0\} \cup V_c$ denotes the set of nodes that itself is composed of a set of customers $V_c := \{1, 2, ..., n\}$ and the depot 0, and $E := \{(i, j) : i \in V, j \in V \setminus \{i\}\}$ is the set of arcs. The planning horizon is $H \in \mathbb{R}_{>0}$. Customer $i \in V_c$ consumes product at a constant rate

$r_i \in \mathbb{R}_{\geq 0}$. Associated with customer $i \in V_c$ are the initial product inventory level $I_i^0 \in \mathbb{R}_{\geq 0}$ at the beginning of the planning horizon, the minimum inventory level $I_i^H \in \mathbb{R}_{\geq 0}$ at the end of the planning horizon, and the inventory lower/upper bound $I_i^l / I_i^u \in \mathbb{R}_{\geq 0}$ during the planning horizon. Without loss of generality, we assume that each customer has to be visited at least once, e.g., $I_i^H + r_i H - I_i^0 > 0$ for $i \in V_c$. We assume that the depot has unlimited supply of product. A fleet of K identical vehicles of capacity $Q \in \mathbb{R}_{>0}$, initially located at the depot, will be used to deliver product to customers so that inventory levels at customers are maintained within the desired intervals. Vehicles are allowed to perform multiple trips and customers are allowed to be visited multiple times during the planning horizon. We allow a vehicle to wait at a customer's location and to make multiple deliveries. Loading at the depot and unloading at customers are considered to be instantaneous. Associated with arc $(i, j) \in E$ are the travel time $t_{ij} \in \mathbb{R}_{\geq 0}$, travel cost $c_{ij} \in \mathbb{R}_{\geq 0}$ by a vehicle. Assume that both the travel time and travel cost matrices satisfy the triangle inequality. We do not allow multiple vehicles to visit a customer at the same time. Hence, when a vehicle is waiting at a customer for another delivery, no vehicle is allowed to visit this customer but other vehicles can wait until the customer becomes available. No inventory cost at the depot or customers but travel cost is considered. The objective is to identify the minimum-cost routing plan and schedule such that (i) the inventory constraints are satisfied; (ii) the vehicle capacity is respected in each trip; (iii) every used vehicle has to return to the depot by the end of the planning horizon.

2.3 MATHEMATICAL MODELING

In this section, we present a mathematical formulation for the CIRP. Modeling the CIRP is not a straightforward task and we are facing the following challenges: (i) the first one is how to handle multiple visits at a customer in the continuous-time setting; (ii) another layer of complexity emanates from the possibility that a vehicle might wait at a customer's location and make several deliveries; (iii) the third one comes from multiple use of a vehicle; (iv) the last challenge is how to monitor and manage inventory levels continuously. In what follows, we will discuss how to tackle these challenges.

Before we discuss how to model the CIRP, we first introduce some notations. During the planning horizon, every customer consumes product at a fixed rate and the total consumption by a customer may exceed the vehicle capacity Q. In such a case, split delivery will happen and vehicles have to deliver product to this customer several times. We assume that customer $i \in V_c$ can be visited at most n_i times during the planning

horizon. Note that this has a motivation in practice. On the one hand, each customer is not willing to be visited too frequently during the planning horizon; on the other hand, the distributor usually sets a minimal amount of product delivery at every customer and hence the amount can be utilized to compute a valid value for n_i. Define $N_i := \{1, 2, ..., n_i\}$ to be the set of visit numbers to $i \in V_c$, and we also define $N_0 := \{1\}$ for convenience. The CIRP can be represented on a network where each node is represented by a pair (i, α), where $i \in V$ indicates either the depot or a customer and $\alpha \in N_i$ indicates the visit number to i. Let (i, α, j, β) represent the arc from node (i, α) to node (j, β) in the network. The similar idea of duplicating nodes to represent multiple visits at a customer was also employed in the work of [7] where the authors considered a maritime inventory routing problem and created several copies to denote the visit number of a port by a ship. In order to handle multiple trips by a vehicle, we propose a concept called "mode" to indicate arc copies. Let $D := \{0, 1\}$ denote the set of two modes. A vehicle traverses an arc from (i, α) to (j, β) directly in mode "0", while in mode "1" a vehicle starts from (i, α), makes a detour to the depot $(0, 1)$ for replenishment, and then goes to the node (j, β) for delivery. One can also create duplicate nodes of the depot to keep track of the number of trips performed by a vehicle, however, two issues will arise: (i) the maximum number of trips that a vehicle is allowed to perform has to be imposed; (ii) in order to keep track of arrival/departure times for each vehicle, a vehicle-index based model is resulted. These burdens are avoided when using our proposed concept "mode". A similar idea was also applied in the work of [102], in which the authors duplicated the arcs so as to represent the detour trip to the depot while solving the *multi-trip vehicle routing problem*. Let A^d represent the set of valid arcs in mode $d \in D$. Clearly, $A^0 := \{(i, \alpha, j, \beta) : i \in V, \alpha \in N_i, j \in V \setminus \{i\}, \beta \in N_j\}$ and $A^1 := \{(i, \alpha, j, \beta) : i \in V_c, \alpha \in N_i, j \in V_c \setminus \{i\}, \beta \in N_j\} \cup \{(i, \alpha, j, \beta) : i = j \in V_c, \alpha \in N_i, \beta \in N_j : \beta > \alpha\}$. Note that in mode "1", arc (i, α, i, β) with $\alpha < \beta$ is valid. This represents a case where a vehicle leaves customer i, returns to the depot for replenishment, and then visits customer i for delivery again. Let $\delta_d^+(j, \beta) := \{(i, \alpha) : (i, \alpha, j, \beta) \in A^d\}$ and $\delta_d^-(j, \beta) := \{(i, \alpha) : (j, \beta, i, \alpha) \in A^d\}$ denote the set of nodes in the network that are connected with node (j, β) by in-coming and out-going arcs in mode $d \in D$, respectively.

Objective Function. The objective is to minimize the total travel cost. The cost of traversing an arc from node (i, α) to node (j, β) by a vehicle in mode d is denoted by c_{ij}^d. Clearly, $c_{ij}^0 = c_{ij}$ since it is simply a direct traversal from node (i, α) to node (j, β), while $c_{ij}^1 = c_{i0} + c_{0j}$ because the vehicle makes a detour to the depot $(0, 1)$ in mode "1". Let $x_{i\alpha j\beta}^d$ be a

binary variable that is equal to 1 if the arc from (i, α) to (j, β) is traversed in mode d and 0 otherwise. The objective function is thus defined as follows:

$$\sum_{d \in D} \sum_{(i,\alpha,j,\beta) \in A^d} c_{ij}^d x_{i\alpha j\beta}^d. \tag{2.1}$$

Degree Constraints. Let $y_{j\beta}$ be a binary variable that is equal to 1 if node (j, β) is visited during the planning horizon and 0 otherwise. If a visit is made to node (j, β), there must be an arc entering this node and an arc leaving it, as shown by (2.4) and (2.5).

$$x_{i\alpha j\beta}^d \in \{0, 1\} \qquad \forall (i, \alpha, j, \beta) \in A^d, \forall d \in D \tag{2.2}$$

$$y_{j\beta} \in \{0, 1\} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.3}$$

$$\sum_{d \in D} \sum_{(i,\alpha) \in \delta_d^+(j,\beta)} x_{i\alpha j\beta}^d = y_{j\beta} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.4}$$

$$\sum_{d \in D} \sum_{(i,\alpha) \in \delta_d^-(j,\beta)} x_{j\beta i\alpha}^d = y_{j\beta} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.5}$$

Fleet Size Constraint. There is no detour that starts from the depot, hence the number of used vehicles is computed as the number of arcs that leave the depot node $(0, 1)$. The fleet of K vehicles are initially located at the depot. Therefore, the number of used vehicles is bounded from above by K, as the constraint (2.6) shows.

$$\sum_{(j,\beta) \in \delta_0^-(0,1)} x_{01j\beta}^0 \leq K \tag{2.6}$$

Time Constraints. We now define parameters $[w_{j\beta}^l, w_{j\beta}^u]$ for node $(j, \beta), j \in V_c$ to indicate the time window during which a vehicle will possibly arrive at this node. One can immediately come up with a valid time window, e.g., $[w_{j\beta}^l, w_{j\beta}^u] = [t_{0j}, H - t_{j0}]$, the earliest arrival time and the latest departure time. Let non-negative variables $a_{j\beta}$ and $d_{j\beta}$ denote the arrival and departure time at node (j, β) if it is visited, respectively. Let $\tilde{a}_{i\alpha j\beta}$ be a non-negative variable that represents the arrival time from node (i, α) to node (j, β) if this arc is traversed in any mode and 0 otherwise. Equations (2.7) - (2.9) enforce that any vehicle can only leave the depot after time 0 and has to return back to the depot by the end of the time horizon H. Constraints (2.10) build the relationship between $a_{j\beta}$ and $\tilde{a}_{i\alpha j\beta}$, $a_{j\beta}$ and $d_{j\beta}$. Since the unloading process is instantaneous, thus the departure from node (j, β) can even happen immediately after the arrival. The time for traversing an arc from node (i, α) to node (j, β) by a vehicle in mode d is denoted by T_{ij}^d. Clearly, $T_{ij}^0 := t_{ij}$ and

$T^1_{ij} := t_{i0} + t_{0j}$. Constraints (2.11) relate the departure time at node (i, α) to the arrival time at some node (j, β) when a vehicle travels from (i, α) to (j, β) in mode 0 or 1.

$$w^l_{j\beta} x^0_{01j\beta} \leq \tilde{a}_{01j\beta} \leq w^u_{j\beta} x^0_{01j\beta} \qquad \forall (j,\beta) \in \delta^-_0(0,1) \qquad (2.7)$$

$$0 \leq \tilde{a}_{i\alpha 01} \leq H x^0_{i\alpha 01} \qquad \forall (i,\alpha) \in \delta^+_0(0,1) \qquad (2.8)$$

$$0 \leq \tilde{a}_{i\alpha j\beta} \leq w^u_{j\beta} \sum_{d\in D:(i,\alpha,j,\beta)\in A^d} x^d_{i\alpha j\beta} \qquad \forall (i,\alpha,j,\beta) \in \cup_{d\in D} A^d : i,j \in V_c \qquad (2.9)$$

$$\sum_{(i,\alpha)\in \cup_{d\in D}\delta^+_d(j,\beta)} \tilde{a}_{i\alpha j\beta} = a_{j\beta} \leq d_{j\beta} \qquad \forall \beta \in N_j, \forall j \in V_c \qquad (2.10)$$

$$d_{i\alpha} + \sum_{d\in D} \sum_{(j,\beta)\in\delta^-_d(i,\alpha)} T^d_{ij} x^d_{i\alpha j\beta} \leq \sum_{(j,\beta)\in\delta^-_0(i,\alpha)} \tilde{a}_{i\alpha j\beta} \qquad \forall \alpha \in N_i, \forall i \in V_c \qquad (2.11)$$

Non-overlapping Visit. When node $(i, \alpha+1)$ is visited (e.g., $y_{i\alpha+1} = 1$), that is, customer i is visited at the $(\alpha+1)$-th time, then this customer must have been visited before (e.g., $y_{i\alpha} = 1$), as indicated by (2.12). Constraints (2.13) require that the recent arrival must happen after the previous departure since a customer is not allowed to be visited by more than one vehicle at the same time.

$$y_{i\alpha+1} \leq y_{i\alpha} \qquad \forall \alpha \in N_i \setminus \{n_i\}, \forall i \in V_c \qquad (2.12)$$

$$d_{i\alpha} \leq a_{i\alpha+1} + (H - t_{i0})(y_{i\alpha} - y_{i\alpha+1}) \qquad \forall \alpha \in N_i \setminus \{n_i\}, \forall i \in V_c \qquad (2.13)$$

Loading an Unloading Constraints. To model the loading and unloading constraints, we define the following continuous variables: $f_{i\alpha j\beta}$ denotes the amount of product that a vehicle transports from node (i, α) to node (j, β) in mode 0; $l_{i\alpha}$ represents the amount of product that a vehicle, after finishing its current trip, reloads at the depot before immediately coming to node (i, α); $q_{i\alpha}$ is the amount of product delivered to customer i at the α-th visit. We remark that $q_{i\alpha}$ might be larger than $I^u_i - I^l_i$, since vehicles are allowed to wait at a customer's location and to make multiple deliveries. Equations (2.14) require that the quantity on a vehicle that travels from node (i, α) to the depot is zero. Constraints (2.15) and (2.16) require that the vehicle capacity is respected. Constraints (2.17) impose upper limits on the unloading quantities. Equations (2.18) are the flow conservation constraints

at node (j, β), ensuring that the amount of product that flows into node (j, β) minus the delivery is equal to the amount that flows out.

$$f_{i\alpha 01} = 0 \qquad \forall (i,\alpha) \in \delta_0^+(0,1) \tag{2.14}$$

$$0 \leq f_{i\alpha j\beta} \leq Q x^0_{i\alpha j\beta} \qquad \forall (i,\alpha,j,\beta) \in A^0 : j \in V_c \tag{2.15}$$

$$0 \leq l_{j\beta} \leq Q \sum_{(i,\alpha)\in\delta_1^+(j,\beta)} x^1_{i\alpha j\beta} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.16}$$

$$0 \leq q_{j\beta} \leq Q y_{j\beta} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.17}$$

$$\sum_{(i,\alpha)\in\delta_0^+(j,\beta)} f_{i\alpha j\beta} + l_{j\beta} - q_{j\beta} = \sum_{(i,\alpha)\in\delta_0^-(j,\beta)} f_{j\beta i\alpha} \qquad \forall \beta \in N_j, \forall j \in V_c \tag{2.18}$$

Inventory Constraints. The inventory constraints are considered for each customer. They ensure that inventory levels fall within the corresponding desired levels at any moment of the planning horizon. In order to achieve this, we only need to guarantee that at the moment $a_{i\alpha}$ a vehicle is arriving at customer $i \in V_c$, the inventory level is above the minimum level I_i^l and that at the moment $d_{i\alpha}$ a vehicle is departing from customer i, the inventory level is below the maximum level I_i^u, as indicated by constraints (2.19) and (2.20). Note that during the time $[a_{i\alpha}, d_{i\alpha}]$, the vehicle is waiting at the customer i and can perform multiple deliveries at zero cost, thus inventory limits I_i^l and I_i^u will never be overridden. At the end of the planning horizon, the inventory level at customer $i \in V_c$ should be above the minimum inventory level I_i^H, as shown by (2.21).

$$I_i^0 y_{i\alpha} + \sum_{\alpha' \in N_i : \alpha' < \alpha} q_{i\alpha'} - r_i a_{i\alpha} \geq I_i^l y_{i\alpha} \qquad \forall \alpha \in N_i, \forall i \in V_c \tag{2.19}$$

$$I_i^0 y_{i\alpha} + \sum_{\alpha' \in N_i : \alpha' \leq \alpha} q_{i\alpha'} - r_i d_{i\alpha} \leq I_i^u y_{i\alpha} + \sum_{\alpha' \in N_i : \alpha' < \alpha} Q\,(y_{i\alpha'} - y_{i\alpha}) \qquad \forall \alpha \in N_i, \forall i \in V_c \tag{2.20}$$

$$I_i^0 + \sum_{\alpha \in N_i} q_{i\alpha} - r_i H \geq I_i^H \qquad \forall i \in V_c \tag{2.21}$$

The CIRP is to minimize the objective (2.1) subject to constraints (3.2) - (2.21). We remark that this is a compact MILP formulation for the CIRP and that this formulation can handle complex routing/scheduling features such as multiple customer visits, multiple trips by a vehicle and continuous-time inventory management.

2.4 TIGHTENING TECHNIQUES

In this section, we derive various tightening inequalities to strengthen the LP relaxations of our proposed model. In particular, we tighten time windows of any visit to a customer and use them to eliminate variables from our proposed MILP formulation. We then consider the minimum quantity of product delivered to a customer and propose capacity constraints as strengthening cuts.

2.4.1 *Tightening of Time Windows*

We can tighten the time window $[w^l_{j\beta}, w^u_{j\beta}]$ by taking advantage of inventory constraints. For example, the latest time point at which node (j, β) can be visited is when the inventory level at customer j, after the previous $(\beta - 1)$ full truckload deliveries, is going to reach the minimum required level I^l_j, e.g., $(I^0_j + Q(\beta - 1) - I^l_j)/r_j$. The node (j, β) cannot be visited too early so that after $(n_j - \beta)$ full truckload deliveries, the final inventory level is still above I^H_i at the end of the planning horizon. Hence, we obtain the following:

$$w^l_{j\beta} = \max\left\{t_{0j}, H - \frac{I^u_j + Q(n_j - \beta) - I^H_j}{r_j}\right\},$$
$$w^u_{j\beta} = \min\left\{H - t_{j0}, \frac{I^0_j + Q(\beta - 1) - I^l_j}{r_j}\right\}.$$

The valid time windows we derive, e.g., $[w^l_{j\beta}, w^u_{j\beta}]$, are usually not tight, such that applying the data preprocessing procedure proposed in [101] to further tighten time windows and to eliminate variables does not make an effect. Thus, we do not include this procedure.

2.4.2 *Variable Elimination*

Given node (i, α) and node (j, β), whenever $w^l_{i\alpha} + T^d_{ij} > w^u_{j\beta}$, then $x^d_{i\alpha j\beta} = 0$. This represents a case where a vehicle that leaves node (i, α) even at its earliest cannot arrive at node (j, β) by the end of the visit time window.

2.4.3 *Minimal # Visits*

Define $\theta_i := I_j^H + r_j H - I_j^0$ for customer $i \in V_c$. The distributor has to ship at least θ_i unit of product to customer $i \in V_c$ during the time horizon, so as to guarantee that the the inventory level is above I_i^H at the end of the planning horizon. We denote by m_i the minimum number of visits at customer i that have to be performed. Clearly, $m_i = \lceil \theta_i / Q \rceil$ since at most a full truckload Q can be delivered to customer i in each visit. Hence, constraints (2.22) are valid. Since we know that node (i, α) will be visited in every feasible solution, we can enforce valid lower bounds for arrival time variables $a_{i\alpha}$, as shown by constraints (2.23).

$$y_{i\alpha} = 1 \qquad \forall \alpha \in N_i : \alpha \leq m_i, \forall i \in V_c \tag{2.22}$$

$$a_{i\alpha} \geq w_{i\alpha}^l \qquad \forall \alpha \in N_i : \alpha \leq m_i, \forall i \in V_c \tag{2.23}$$

2.4.4 *Rounded Capacity Inequalities*

The *rounded capacity inequalities* [110] are well-known strengthening inequalities that have demonstrated great success in the expedition of solving routing models. We adapt them into the context of CIRP, as shown by constraints (2.24).

$$\sum_{j \in S} \sum_{\beta \in N_j} \left(\sum_{(i,\alpha) \in \delta_0^+(j,\beta) : i \notin S} x_{i\alpha j\beta}^0 + \sum_{(i,\alpha) \in \delta_1^+(j,\beta)} x_{i\alpha j\beta}^1 \right) \geq r(S) \quad \forall S \subseteq V_c \tag{2.24}$$

Given a set of customers $S \subseteq V_c$, let $\theta(S)$ denote $\sum_{i \in S} \theta_i$. The left hand side of (2.24) denotes the number of vehicles that are going to deliver product to customers in the set S. Note that the second term of the left hand side accounts for multiple use of a vehicle. The right hand side, $r(S)$, denotes the minimum number of vehicles required to serve the customers in S. That is, $r(S)$ is the optimal value to the *bin-packing problem with item fragmentation* [122] (BPPIF) with bin capacity Q, item sizes given by the minimum demands of the customers in S, and each item fragmented into at most n_i pieces. Calculating $r(S)$ exactly is as hard as the BPPIF and is therefore strongly $\mathcal{NP}$-hard. However, the formulation remains valid if one replaces $r(S)$ on the right hand side with the obvious lower bound $k(S) := \lceil \theta(S)/Q \rceil$, which yields the so-called *rounded capacity inequalities*. We remark that capacity constraints (2.24) are also used in the *vehicle routing problem with split delivery* [31, 38].

In order to separate RCI, we first construct the support graph $\bar{G} = (V, E)$ that corresponds to the LP fractional solution at a given branch-and-bound node. In particular, we loop through all positive $\bar{x}^d_{i\alpha j\beta}$ values for $(i, \alpha, j, \beta) \in A^d, d \in D$ and we then increment the weight of arc $(i, j) \in E$ by $\bar{x}^d_{i\alpha j\beta}$ if $d = 0$ and increment the weights of both arc $(i, 0) \in E$ and arc $(0, j) \in E$ by $\bar{x}^d_{i\alpha j\beta}$ if $d = 1$. We emphasize that the resulting graph $\bar{G}$ is not the support graph that corresponds to the classic *capacitated vehicle routing problem* because the degree of each customer node is not necessarily exactly 2. Hence, the commonly used CVRPSEP package [120] is not applicable here. We apply three heuristics to separate RCI. As [121] suggested, we first identify all *connected components* in the graph $\bar{G}$ as candidates. The second heuristics is to separate the so-called *fractional capacity inequalities* via the max-flow algorithm. If both heuristics fail, we then resort to the *tabu search* method proposed by [21] to identify violated RCI. Readers are referred to [21] for implementation details.

A special case of RCI is when $S = V_c$, as shown by constraint (2.25). We notice that adding this constraint generally expedites the branch-and-cut algorithm, thus in our implementation, we always append this one to our proposed model before the branch-and-cut search starts.

$$\sum_{j \in V_c} \sum_{\beta \in N_j} \left(x^0_{01j\beta} + \sum_{(i,\alpha) \in \delta_1^+(j,\beta)} x^1_{i\alpha j\beta} \right) \geq \left\lceil \frac{\theta(V_c)}{Q} \right\rceil \tag{2.25}$$

In our model, since customer nodes are optionally visited (e.g., $y_{i\alpha} \in \{0, 1\}$), we then incorporate *logical inequalities* and *generalized subtour elimination constraints* proposed in the work of [73] as strengthening cuts. However, our computational studies show that these inequalities are not effective and thus we decide not to include them. Finally, we propose the following branch-and-cut algorithms: a polynomial-size mixed-integer linear program (2.1) - (2.23) and (2.25), is given to an MILP solver and RCI (2.24) are dynamically separated and added into the model at each branch-and-bound node.

2.5 COMPUTATIONAL STUDIES

In this section, we test our branch-and-cut algorithm on CIRP benchmark instances and compare it against the state-of-the-art approach from [108]. Our algorithm was implemented in C++ and the mixed-integer linear program was solved using the Gurobi Optimizer 9.0 through the C application programming interface. User cuts were implemented via the Gurobi callback function. Branching was performed in priority on variables $y_{i\alpha}$, and then on $x^d_{i\alpha j\beta}$ variables. Unless otherwise mentioned, all Gurobi settings were kept

being default, except that the relative optimality gap tolerance was set to be 0 and that the absolute optimality gap tolerance was set properly based on different datesets. The experiments were run on an Intel Xeon CPU E5-2689 v4 server running at 3.10 GHz. The 128 GB of available RAM was shared among 10 copies of the algorithm running in parallel on the server. Each instance was solved by one copy of the algorithm using a single thread.

2.5.1 *Benchmark instances*

We consider two CIRP datasets for testing our algorithm. The first one comes from the work of [108], in which the authors generated two types of CIRP instances: `clustered` (C) and `random` (R).[1] There are 45 instances for each type, with the number of customers ranging from 5 to 15 and the number of vehicle ranging from 3 to 15. All instances can be found at https://github.com/felipelagos/cirplib. As [108] did, the travel costs and travel times are taken to be the Euclidean distance and then rounded to two decimal places. The triangle inequality still holds after rounding.

To evaluate the performance of our algorithm on solving real-life CIRP data, we generate the second dataset that is inspired by ROADEF/EURO Challenge 2016 [8]. In particular, we obtain the ROADEF/EURO Challenge 2016 dataset (version 1.1) from https://www.roadef.org/challenge/2016/en/instances.php and apply the following procedure: (i) a heterogeneous fleet is reduced to a homogeneous one; (ii) driver-trailer scheduling is neglected; (iii) the product consumption rate is considered to be constant throughout the planning horizon; (iv) the objective is to minimize the travel cost, rather than the logistic ratio. We remark that incorporating all the aforementioned features into our model is doable but it will result in a gigantic formulation that could not be solved within a reasonable amount of time, hence we decide not to do so in this chapter. The ROADEF/EURO Challenge 2016 dataset is a excerpt of real problems and it includes 9 medium-size instances, with the number of customers ranging from 53 to 89. For each instance, we choose the first 5, 7, 10, 12, 15, 17 and 20 customers, respectively, and create a new instance. The number of vehicles ranges from 1 to 6. In total, we generate a suite of 63 `roadef` (RF) instances, each with a name "RF-X-nY" representing a Y-customer instance adapted from the X^{th} original instance. All the generated instances can be found at gounaris.cheme.cmu.edu/dataset/cirp.

1 The authors of [109] consider a special case of the CIRP: the CIRP with only out-and-back routes (i.e., a vehicle route starts at the depot visits a single customer and returns to the depot). One can simply fix $x^0_{i\alpha j\beta} = 0$ for $(i, \alpha, j, \beta) \in A^0 : i, j \in V_c$ in our proposed model, resulting in a valid formulation for the CIRP with out-and-back routes. For the sake of brevity, we do not consider these instances.

For each instance from both datasets, the number of visits to customer $i \in V_c$ is limited to 2 plus the minimal number of visits at customer i, i.e., $n_i := 2 + m_i$, since we did not notice any improvement in the total travel cost when we increase this number. We remark that the choice of n_i might be still limiting in a sense that when increasing their values, one may identify a feasible solution with a smaller objective value than that of the optimal solution in the case of our chosen n_i values.

2.5.2 *Computational results on literature data*

We first test our branch-and-cut algorithm (denoted by "Branch-and-Cut") on the first dataset (`clustered` and `random` instances) and compare its performance with that from the work of [108] (denoted by `LBS20`). To analyze the effect of adding RCI on the solution process, we consider a variant of our algorithm in which RCI are disabled. In this case, no user cuts are added to the model and hence we denote it by "Gurobi (default)". In our implementation, the absolute optimality gap tolerance was set to be 0.0099. Considering that the travel costs are rounded to two decimal places, the objective value of a feasible solution has at most two decimal places and thus the absolute optimality gap tolerance is indeed valid. In the Branch-and-Cut algorithm, all Gurobi-generated cuts were disable because we observed that using these general-purpose cuts increased overall computation times.

We impose a time limit of 10 hours for each instance and compare our computational results with those from `LBS20` in Table 2.1. Columns "# cust." and "# inst." report the number of customers in a instance and the number of instances of such an input size, respectively. The column "# opt." reports the number of instances that were solved to optimality within the time limit, while columns "# feas." and "# no feas." denote the number of instances for which a feasible solution was and was not identified by the time the algorithm terminated due to time limit, respectively. For those solved instances, we report the average solution time (rounded to the nearest integer, in seconds) in column "Avg. t(s)". For those unsolved instances but with feasible solutions identified, we report the average residual gap (a residual gap is defined as (UB - LB) / UB) in column "Avg. gap (%)". Note that in Table 2.1, we do not report the solution time for `LBS20`, because their adopted approach was to solve many (around 40) parametric MILP formulations, each with a time limit of 2 hours of CPU time, and the authors did not report the total solution time.

Out of 90 CIRP instances, `LBS20` solved 26 of them to optimality and returned average residual gaps of $1.6\% - 17.7\%$, while our Branch-and-Cut algorithm solved 56 instances

to optimality within the time limit of 10 hour and did not successfully generate feasible solutions for 3 instances, leaving the remaining 31 instances an average residual gap below 2.0%. In particular, our algorithm solved all 5- and 7-customer instances, half of 10-customer instances, one third of 12-customer instances, and a few 15-customer instances to optimality. We remark that for each of those 3 instances to which our Branch-and-Cut algorithm did not identify a feasible solution, our algorithm did find one when the time limit increases. Based on these observations, we claim that our Branch-and-Cut approach significantly outperforms the approach from LBS20. We remark that the CIRP is extremely challenging, considering that the discrete-time IRP instances with up to 50 customers have been solved optimally while even the CIRP of 15 customers is almost intractable.

We present detailed results on `clustered` (C) instances and `random` (R) instances, respectively, for the Branch-and-Cut algorithm in Table 2.3 and Table 2.4. In these tables, if an instance could be solved to optimality within 10 hours, then "Opt [UB]" reports the corresponding optimal objective value, while "t (s) [LB]" reports the time (rounded to the nearest integer, in seconds) to solve the instance to optimality. Otherwise, the columns respectively report in brackets the best upper and lower bounds found within the time limit. We also report the average number of visits across all customers in the returned best known solution to each instance in column "Avg. # visits" and the number of RCI that have been dynamically added in column "# RCI". If our algorithm could not identify a feasible solution, then the entries for "Opt [UB]" and "Avg. # visits" are left blank. The average number of visits to every customer ranges from 1.2 to 2.2, which is far from the maximum number of allowable visits, n_i, that ranges from 3 to 5. This indicates that our choice of the number of allowable visits is not restrictive.

To have a closer look at the solution quality, we consider four instances of which the detailed computational results were presented in the appendix of LBS20. We report the instance name (Inst.) and the final upper bound (UB), root node lower bound[2] (Root LB), root node gap[3] (Root gap (%)), final lower bound (LB), residual gap (Gap %), and solution time (t (s)) for this instance in Table 2.2. The instance name with "C" ("R") as an initial belongs to `clustered` (`random`) instances, and the number after the initial denotes the number of customers in this instance. Our Branch-and-Cut algorithm solved the first three instances with up to 10 customers efficiently and returned either equivalently good or better solutions. For the 15-customer instance "C15U2Q2", the feasible solution returned by our algorithm is also better than the one from LBS20 and the residual gap generated is smaller. Another observation is, in our proposed algorithm, the root node gap is quite

2 The "root node lower bound" denotes the root node relaxation value before branching.
3 The "root node gap" is defined as (UB - Root LB) / UB.

Table 2.1: Comparison among LBS20 and our proposed algorithms on 90 benchmark instances

# cust.	# inst.	LBS20			Gurobi (default)					Branch-and-Cut				
		# opt.	# feas.	Avg. gap (%)	# opt.	# feas.	# no feas.	Avg. t (s)	Avg. gap (%)	# opt.	# feas.	# no feas.	Avg. t (s)	Avg. gap (%)
clustered														
5	9	7	2	1.55	9	0	0	18	–	9	0	0	2	–
7	9	3	6	1.92	6	3	0	1,278	3.79	9	0	0	4,439	–
10	9	1	8	1.90	2	7	0	11,449	3.92	3	6	0	8,066	1.00
12	9	1	8	3.26	0	9	0	–	4.11	2	7	0	37	0.91
15	9	1	8	8.55	0	3	6	–	7.05	1	7	1	307	1.03
random														
5	9	5	4	2.93	9	0	0	3	–	9	0	0	1	–
7	9	5	4	3.27	9	0	0	791	–	9	0	0	22	–
10	9	3	6	4.10	5	4	0	1,681	1.85	7	2	0	104	1.99
12	9	0	9	8.93	1	8	0	926	4.77	5	4	0	4,051	1.59
15	9	0	9	17.67	0	8	1	–	5.90	2	5	2	14,599	0.82
Total	90	26	64		41	42	7			56	31	3		

small for some instances, e.g., R5U2Q2 and C15U2Q2. Again, this analysis demonstrates the effectiveness and efficiency of our Branch-and-Cut algorithm over the approach from LBS20.

Table 2.2: Detailed results for LBS20 and our Branch-and-Cut algorithm on 4 benchmark instances

Inst.	# cust.	LBS20			Branch-and-Cut					
		UB	LB	Gap (%)	UB	Root LB	Root gap (%)	LB	Gap (%)	t (s)
R5U2Q2	5	36.59	36.42	0.46	36.51	36.42	0.25	36.51	0.00	1
R10U1Q2	10	64.02	64.02	0.00	64.02	58.56	8.53	64.02	0.00	6
R10U2Q2	10	87.92	82.71	5.93	82.92	75.54	8.90	82.92	0.00	269
C15U2Q2	15	97.00	93.98	3.11	95.25	92.64	2.74	93.98	1.33	36,000

We now revisit Table 2.1 for evaluating the effect of RCI on the branch-and-bound process. Compared with the Gurobi (default) version, the Branch-and-Cut algorithm with RCI enabled as strengthening inequalities solved 15 more benchmark instances to optimality within the time limit of 10 hours, including a few 15-customer instances. Both the solution time for each solved instance and the residual gaps for unsolved instances were decreased when enabling RCI. The total number of RCI that were identified and introduced as tightening inequalities is hundreds for small-size CIRP instances and thousands for large-size ones (see Table 2.3 and Table 2.4). In order to better demonstrate the effectiveness of RCI, we present the performance profile [66] in Figure 7.7. We can claim that RCI are very effectual in the expedition of the branch-and-cut search.

2.5.3 *Computational results on roadef instances*

We now turn our attention to the second dataset (roadef instances) and evaluate our Branch-and-Cut algorithm's performance on solving real-life CIRP instances. In our implementation, the absolute optimality gap tolerance was set to be 0.099, since in these instances all entries of the travel cost matrix have one decimal place.

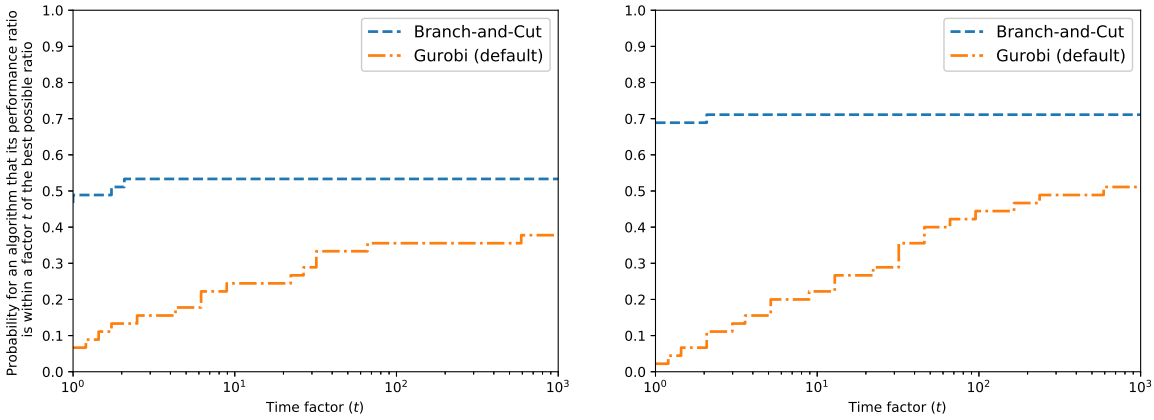
We impose a time limit of 10 hour for each instance and present a synopsis of our computational results in Table 2.5. All column names have the same meanings as described in Section 2.5.2. Out of 63 roadef instances, our Branch-and-Cut algorithm solved 56 of them to optimality and returned an average residual gap of 0.43% – 1.83% for the remaining 7 instances. In particular, our algorithm solved all 5-, 7-, 10- and 17-customer instances, almost all 12- and 15-customer instances, about half of 20-customer instances.

Table 2.3: Detailed results for our Branch-and-Cut algorithm on 45 `clustered` instances

Inst.	Opt [UB]	t (s) [LB]	Avg. # visits	# RCI	Inst.	Opt [UB]	t (s) [LB]	Avg. # visits	# RCI
C5U1Q1	37.85	1	1.4	2	C10U2Q3	[56.10]	[54.87]	1.7	417
C5U1Q2	30.61	2	1.4	21	C10U3Q1	104.37	6,807	2.1	101
C5U1Q3	28.24	1	1.2	7	C10U3Q2	[83.36]	[83.21]	1.9	281
C5U2Q1	50.09	1	1.8	8	C10U3Q3	[65.62]	[64.85]	1.7	183
C5U2Q2	38.28	2	1.6	6	C12U1Q1	85.07	13	1.4	49
C5U2Q3	30.61	3	1.4	17	C12U1Q2	[69.28]	[68.82]	1.7	369
C5U3Q1	57.74	10	2.2	24	C12U1Q3	[57.07]	[56.51]	1.5	348
C5U3Q2	43.36	1	1.8	10	C12U2Q1	[104.58]	[104.11]	2.0	2,905
C5U3Q3	36.16	1	1.8	10	C12U2Q2	[80.01]	[79.28]	1.8	655
C7U1Q1	52.14	1,708	1.7	57	C12U2Q3	[69.49]	[68.83]	1.6	285
C7U1Q2	42.99	9,195	1.7	170	C12U3Q1	124.43	61	2.1	61
C7U1Q3	31.17	26	1.4	56	C12U3Q2	[101.91]	[101.05]	2.1	9,581
C7U2Q1	59.75	1	1.9	14	C12U3Q3	[80.19]	[78.94]	1.8	428
C7U2Q2	50.48	21,254	2.0	138	C15U1Q1	99.78	307	1.5	398
C7U2Q3	42.18	51	1.6	70	C15U1Q2	[77.32]	[76.59]	1.5	699
C7U3Q1	78.08	2	2.1	16	C15U1Q3	[64.48]	[63.90]	1.4	1,075
C7U3Q2	59.44	11	1.6	24	C15U2Q1	[122.45]	[122.06]	1.9	7,476
C7U3Q3	50.18	8,194	1.7	102	C15U2Q2	[95.25]	[93.98]	1.7	698
C10U1Q1	70.97	13	1.4	63	C15U2Q3	[78.64]	[76.59]	1.7	5,301
C10U1Q2	[55.62]	[54.85]	1.6	221	C15U3Q1	[147.06]	[147.02]	2.1	806
C10U1Q3	[45.24]	[45.22]	1.5	151	C15U3Q2		[116.43]		15,242
C10U2Q1	86.02	17,377	1.8	133	C15U3Q3	[94.67]	[93.62]	1.9	823
C10U2Q2	[65.91]	[65.24]	1.8	438					

Table 2.4: Detailed results for our Branch-and-Cut algorithm on 45 `random` instances

Inst.	Opt [UB]	t (s) [LB]	Avg. # visits	# RCI	Inst.	Opt [UB]	t (s) [LB]	Avg. # visits	# RCI
R5U1Q1	36.42	1	1.4	2	R10U2Q3	64.02	9	1.5	45
R5U1Q2	29.06	1	1.2	3	R10U3Q1	116.04	48	2.2	56
R5U1Q3	28.45	1	1.2	12	R10U3Q2	[93.93]	[92.17]	2.0	203
R5U2Q1	41.42	1	1.8	7	R10U3Q3	82.92	204	1.8	142
R5U2Q2	36.51	1	1.6	8	R12U1Q1	[101.88]	[99.90]	1.8	280
R5U2Q3	30.90	1	1.4	9	R12U1Q2	75.45	27	1.4	90
R5U3Q1	45.31	1	2.0	10	R12U1Q3	66.09	11,251	1.3	320
R5U3Q2	39.44	3	2.0	8	R12U2Q1	[118.75]	[118.48]	1.9	187
R5U3Q3	33.54	4	1.6	15	R12U2Q2	98.18	1,598	1.8	682
R7U1Q1	63.03	1	1.7	8	R12U2Q3	76.27	7,006	1.5	210
R7U1Q2	43.93	2	1.1	19	R12U3Q1	139.88	373	2.1	163
R7U1Q3	40.43	157	1.6	79	R12U3Q2	[109.65]	[107.61]	2.1	1,376
R7U2Q1	69.37	1	1.7	6	R12U3Q3	[95.74]	[93.52]	1.7	1,216
R7U2Q2	60.12	1	1.7	22	R15U1Q1	[119.52]	[117.52]	1.7	82,070
R7U2Q3	43.93	3	1.4	25	R15U1Q2	87.67	21,065	1.5	820
R7U3Q1	81.58	8	2.1	30	R15U1Q3	75.25	8,133	1.3	470
R7U3Q2	67.06	4	2.0	23	R15U2Q1	[139.63]	[138.72]	1.9	19,291
R7U3Q3	57.47	19	1.9	36	R15U2Q2		[114.43]		76,645
R10U1Q1	[87.72]	[85.88]	1.8	210	R15U2Q3	[89.45]	[88.75]	1.6	507
R10U1Q2	64.02	6	1.5	42	R15U3Q1	[162.61]	[161.98]	2.1	1,836
R10U1Q3	54.35	182	1.3	114	R15U3Q2	[128.52]	[127.75]	1.9	2,345
R10U2Q1	99.48	7	1.8	60	R15U3Q3		[111.07]		45,783
R10U2Q2	82.92	269	1.8	102					



(a) Clustered (b) Random

Figure 2.1: Log-scaled performance profiles across all benchmark instances. The left graph compares the performance in `clustered` instances, while the right graph compares the performance on `random` instances. For each curve, the value at $t = 0$ gives the fraction of benchmark instances for which it is fastest, while the limiting value at $t \to \infty$ gives the fraction of instances which it could solve within the time limit of 10 hours.

This indicates that our proposed algorithm is able to solve real-life instances of up to 20 customers within a reasonable amount of time. The solution time generally increases with the problem size, ranging from hundreds of seconds to quite a few hours. Another observation is that in the returned solutions, every customer is visited several times (about 3 times) during the planning horizon. For the sake of completeness, we report detailed results in Table 2.6.

Table 2.5: Computational results for our Branch-and-Cut algorithm on 63 `roadef` instances

# cust.	# inst.	# opt.	# feas.	Avg. t (s)	Avg. gap (%)	Avg. # visits
5	9	9	0	149	–	2.7
7	9	9	0	1,579	–	2.8
10	9	9	0	2,314	–	3.0
12	9	8	1	7,978	1.83	3.0
15	9	8	1	6,557	0.43	2.8
17	9	9	0	13,728	–	2.8
20	9	4	5	9,995	0.87	2.7
Total	63	56	7			

2.6 CONCLUSIONS

In this chapter, we consider the continuous-time inventory routing problem. This problem often arises in the context of vendor-managed inventory where the distributor monitors the product usage and manages the tank inventory level in continuous time so as to meet the service commitments. We propose a novel mixed-integer linear programming formulation which incorporates several ingenious modeling ideas to handle multi-trip and multi-visit features as well as continuous-time inventory management. To expedite the solution process, we propose various types of tightening techniques, including the adaption of well-known rounded capacities inequalities, among others. We conduct extensive computational studies on 90 benchmark instances from the literature. The computational results show that our branch-and-cut algorithm performs significantly better than the state-of-the-art approach from the work of [108]. In particular, we close 30 previously open CIRP benchmark instances and return an average gap below 2.0% for unsolved instances. We also conduct detailed analysis on the effect of strengthening inequalities. Further computational studies on real-life data show that our proposed algorithm could solve CIRP instances of up to 20 customer.

Table 2.6: Detailed results for our Branch-and-Cut algorithm on 63 `roadef` instances

Inst.	Opt [UB]	t (s) [LB]	Inst.	Opt [UB]	t (s) [LB]	Inst.	Opt [UB]	t (s) [LB]
RF-3-n5	1,043.7	1	RF-6-n10	1,998.9	450	RF-9-n15	2,000.7	3,062
RF-4-n5	1,043.7	1	RF-7-n10	1,652.4	4,808	RF-10-n15	2,520.0	11,311
RF-5-n5	566.1	1	RF-8-n10	1,837.5	10,953	RF-11-n15	2,520.0	331
RF-6-n5	572.4	1	RF-9-n10	1,631.1	319	RF-3-n17	4,888.1	16,550
RF-7-n5	492.0	9	RF-10-n10	1,869.6	266	RF-4-n17	4,603.2	1,650
RF-8-n5	524.7	84	RF-11-n10	1,869.6	60	RF-5-n17	2,306.1	35,607
RF-9-n5	480.6	1,186	RF-3-n12	3,804.5	3,643	RF-6-n17	2,421.9	4,296
RF-10-n5	1,323.9	45	RF-4-n12	3,757.6	1,450	RF-7-n17	2,094.0	6,212
RF-11-n5	1,323.9	18	RF-5-n12	2,034.9	33,546	RF-8-n17	2,366.1	20,146
RF-3-n7	2,618.7	23	RF-6-n12	[2,197.8]	[2,157.6]	RF-9-n17	2,050.8	2,980
RF-4-n7	2,618.7	62	RF-7-n12	1,760.7	11,680	RF-10-n17	3,145.8	9,221
RF-5-n7	927.3	7	RF-8-n12	1,945.8	7,056	RF-11-n17	3,173.7	26,889
RF-6-n7	1,007.1	2,097	RF-9-n12	1,739.4	748	RF-3-n20	5,518.8	6,057
RF-7-n7	739.8	161	RF-10-n12	1,996.5	580	RF-4-n20	7,410.2	18,717
RF-8-n7	781.5	4,946	RF-11-n12	1,996.5	5,121	RF-5-n20	[2,326.2]	[2,313.9]
RF-9-n7	731.7	3,975	RF-3-n15	4,552.8	836	RF-6-n20	2,436.9	8,505
RF-10-n7	1,368.9	42	RF-4-n15	4,027.8	411	RF-7-n20	2,306.4	6,701
RF-11-n7	1,370.4	2,896	RF-5-n15	[2,280.9]	[2,271.0]	RF-8-n20	[2,598.9]	[2,580.6]
RF-3-n10	3,086.3	140	RF-6-n15	2,392.8	21,099	RF-9-n20	[2,286.6]	[2,265.9]
RF-4-n10	2,913.4	134	RF-7-n15	2,052.6	11,612	RF-10-n20	[3,401.4]	[3,335.1]
RF-5-n10	1,879.8	3,696	RF-8-n15	2,314.2	3,796	RF-11-n20	[3,433.8]	[3,424.2]

3

MIXED-INTEGER LINEAR OPTIMIZATION FOR FULL TRUCKLOAD PICKUP AND DELIVERY

This chapter is focused on solving the full truckload pickup and delivery problem. This problem often arises when the carriers are required to perform full truckload shipments between distribution centers and delivery locations. In this chapter, we propose a novel mixed-integer linear programming formulation to model the full truckload pickup and delivery problem. We first evaluate this model using industrial data and the computational results demonstrate that our proposed model could efficiently solve real-life instances. We then embed this mathematical formulation into a simulation engine and conduct extensive computational studies to quantify the economic effect of the pre-loading policy.

3.1 INTRODUCTION

Supply chain logistics is central to the competitiveness of many businesses, and thus receives lots of attention for optimization. The most classic setting is the *vehicle routing problem* (VRP) [167] where, given a distribution center, a fleet of vehicles and a group of customers to be served, one aims to identify the minimum-cost routes for vehicles to traverse, such that customer demands are satisfied and relevant operational constraints, e.g., vehicle capacities and time windows, among others, are respected. In this chapter, we consider a VRP variant, called the *full truckload pickup and delivery problem* (FTPDP), in which the most typical feature is that all orders are consolidated to full truckloads and thus we do not deal with vehicle capacity constraints and cross-docking options. This problem has numerous applications especially in the trucking industry [18], where the carriers are required to move truckloads of goods between specified distribution centers and delivery locations, using a fleet of tractor trailers available to them. Therefore, studying this problem is of considerable practical importance.

The VRP with full truckloads has been addressed by both heuristic and exact approaches in the literature. The work of [86] presented *savings based algorithms* for time constrained pickup and delivery of full truckloads. The authors of [84] considered the FTPDP with trucks being synchronized on pickup or delivery locations based on unitary loading and unloading resources. They proposed an *adaptive neighborhood search* method and evaluated their heuristic approach using real-life instances. [154] proposed a mathematical modeling approach enhanced by *variable neighbor decomposition search* to address an interesting FTPDP application arising in the biomass supply chain. A prominent contribution for exactly solving the VRP with full truckloads comes from [18], in which the authors presented a *mixed-integer linear programming* (MILP) formulation with exponentially many route-based variables and solved it via the advanced *branch-and-price* approach. Their computational studies on randomly generated instances demonstrate that the proposed approach could optimally solve problems with up to 100 consignments.

In this chapter, we focus on a variation to the FTPDP that is faced by the petrochemical company Braskem. The FTPDP of our interest has the following characteristics: (i) a truck may visit several places for pickup before going to a delivery location; (ii) orders are optionally served; (iii) scheduling decisions have to made at each pickup location due to the loading dock capacity restrictions. In this chapter, we aim to develop an exact mathematical approach for addressing this problem. The distinct contributions of our work can be summarized as follows.

- We propose a novel MILP formulation to model the FTPDP, which is featured by full truckload shipments, multiple pickup points, optional orders, and loading dock capacity restrictions.
- We test our proposed model using real-life operational data and the computational results demonstrate the effectiveness and efficiency of our proposed approach. In particular, our MILP model could solve practical problems of up to 52 consignments to optimality within hundreds of seconds.
- We embed the mathematical formulation into a simulation engine so as to evaluate the economic effect of allowing for pre-loading trucks. Our computational studies show that adopting the pre-loading policy could save up to 2% of the operational cost in the application setting of our interest.

The remainder of the chapter is organized as follows. In Section 7.2, we give a formal problem definition. In Section 3.3, we propose a novel MILP formulation. Section 7.5 presents computational results. Finally, we conclude our work in Section 7.6.

3.2 PROBLEM DEFINITION

The FTPDP of our interest is defined as follows. Let P be a set of plants and C be a set of clients. We denote by $V_c := \{1, 2, ..., n\}$ a set of orders. A heterogeneous fleet of trucks, initially located at the depot 0, will be routed to serve orders. Every truck can leave the depot no earlier than W^L to perform a route and has to return no later than W^U. Let H be a set of truck types. For type $h \in H$, we denote by N_h, f_h, γ_h the number of available trucks, the fixed cost for a truck[1], and the overtime pay of rate if a truck travels after time W, respectively. Let $\mathcal{C}_{hij}$ and $\mathcal{T}_{hij}$, respectively, denote the travel cost and time from i to j, where $i, j \in \{0\} \cup P \cup C$, by a truck of type $h \in H$. Serving an order $i \in V_c$ decomposes into two steps: a truck first visits a set of specified plants $P_i \subseteq P$ in some sequence for picking up at each plant $p \in P_i$ a fractional truckload of product (pickup time α_{hip}) as a full truckload shipment, and then goes to a designated client $c^i \in C$ for delivering the full truckload of product (delivery time β_{hi}). If order i is not served, a bid cost Δ_i will be incurred. Let $H_i \subseteq H$ denote the set of truck types that are compatible with order i [2]. For plant $p \in P$, we denote by T_p a set of non-overlapping time slots during which trucks will be loaded. Let w_{pl} denote the starting point of time slot $l \in T_p$. Plant p starts to load trucks only at w_{pl}'s while the number of trucks that can be loaded simultaneously during the slot l cannot exceed Q_{pl}. A truck is allowed to wait if it arrives before the next time slot starts while the task of loading a truck will always be finished by the end of this time slot (e.g., pickup times are always shorter than time slot durations). The total operational cost consists of fixed costs, overtime costs, travel costs, and bid costs. The objective is to identify cost-effective routes and schedules such that the relevant physical constraints are respected and the total cost is minimized.

3.3 MATHEMATICAL MODELING

We model the above problem in a directed graph $G := (V, A)$ where $V := \{0\} \cup V_c$ denotes the set of nodes, consisting of the depot and shipping orders, and $A := \{(i, j) \in V \times V : i \neq j\}$ denotes the set of arcs. We associate with node $i \in V_c$ a set of modes $M_i := \{1, 2, ..., |P_i|!\}$[3] that indexes all possible plant-visiting sequences for serving

1 In this chapter, a fixed cost is always incurred regardless of the use of a truck, since it consists of truck depreciation/maintenance cost and driver regular pay.

2 A truck of type $h \in H$ is compatible with order i if: (i) every pickup plant $p \in P_i$ could load trucks of type h; (ii) client c^i could be served by trucks of type h.

3 In our application, $|P_i| \leq 3$ for $i \in V_c$.

order i. Let S_{im} denote the corresponding sequence in mode $m \in M_i$. For example, if $P_i = \{p_1, p_2\}$, then $M_i := \{1, 2\}$, $S_{i1} := \langle p_1, p_2, c^i \rangle$ and $S_{i2} := \langle p_2, p_1, c^i \rangle$. Let v^{im} denote the first element of S_{im}. For convenience, we define $M_0 := \{1\}$ and $S_{01} := \langle c^0 \rangle$, where $c^0 = 0$. Traversing arc $(i, j) \in A$ in mode $m \in M_j$ by a truck of type h denotes that this truck travels from c^i to v^{jm} and is ready to serve order j in sequence S_{jm}. Let c^m_{hij} and t^m_{hij}, respectively, denote the traversal cost and time on arc (i, j) in mode m by a truck of type h. By definition, $c^m_{hij} = \mathcal{C}_{hc^i v^{jm}}$ and $t^m_{hij} = \beta_{hi} + \mathcal{T}_{hc^i v^{jm}}$[4]. For notation convenience, let $\delta^+(j) := \{i \in V : (i, j) \in A\}$ and $\delta^-(j) := \{i \in V : (j, i) \in A\}$ denote the set of nodes in the graph that are connected with node j by in-coming and out-going arcs, respectively.

Degree Constraints. Let z_i be a binary variable indicating whether order $i \in V_c$ is served. Let x^m_{hij} be a binary variable that is equal to 1 if arc $(i, j) \in A$ is traversed by a truck of type $h \in H$ in mode $m \in M_j$. Constraints (3.3) simply enforce the balance between the in-coming degree and out-going degree at each node $i \in V_c$ for every truck type while constraints (3.4) relate x^m_{hij} to z_i.

$$z_i \in \{0, 1\} \qquad \forall i \in V_c \tag{3.1}$$

$$x^m_{hij} \in \{0, 1\} \qquad \forall m \in M_j, \forall (i, j) \in A, \forall h \in H \tag{3.2}$$

$$\sum_{j \in \delta^+(i)} \sum_{m \in M_i} x^m_{hji} = \sum_{j \in \delta^-(i)} \sum_{m \in M_j} x^m_{hij} \qquad \forall h \in H, \forall i \in V_c \tag{3.3}$$

$$\sum_{h \in H} \sum_{j \in \delta^-(i)} \sum_{m \in M_j} x^m_{hij} = z_i \qquad \forall i \in V_c \tag{3.4}$$

Fleet Size Constraints. The fleet is initially located at the depot 0. There are N_h available trucks of type $h \in H$, thus the number of used trucks is bounded from above by its availability, as constraints (3.5) show.

$$\sum_{j \in \delta^-(0)} \sum_{m \in M_j} x^m_{h0j} \leq N_h \qquad \forall h \in H \tag{3.5}$$

Loading Dock Capacity Constraints. Let a binary variable u_{ipl} be 1 if order $i \in V_c$ is assigned to the time slot $l \in T_p$ of plant $p \in P_i$ and 0 otherwise. If order i is served, then exactly one time slot will be chosen at every designated plant $p \in P_i$, as shown by (3.7).

4 Without loss of generality, we incorporate delivery times into the travel times along the arcs for convenience.

Constraints (3.8) impose that at each time slot of plant p, the total number of trucks that are being loaded simultaneously should not exceed its loading dock capacity.

$$u_{ipl} \in \{0,1\} \qquad \forall l \in T_p, \forall p \in P_i, \forall i \in V_c \tag{3.6}$$

$$\sum_{l \in T_p} u_{ipl} = z_i \qquad \forall p \in P_i, \forall i \in V_c \tag{3.7}$$

$$\sum_{i \in V_c : p \in P_i} u_{ipl} \leq Q_{pl} \qquad \forall l \in T_p, \forall p \in P \tag{3.8}$$

Plant Scheduling Constraints. Let a_{ip}^m be a non-negative variable that is equal to the loading start time at plant p if order i is served in mode m and 0 otherwise. This is properly enforced by constraints (3.9) - (3.11). Constraints (3.12) build the consistency between a_{ip}^m and the assigned time slot of order i, w_{pl}.

$$0 \leq a_{ip}^m \qquad \forall p \in P_i \cup \left\{c^i\right\}, \forall m \in M_i, \forall i \in V_c \tag{3.9}$$

$$a_{iv}^m + \sum_{h \in H} \sum_{j \in \delta^+(i)} \left(\alpha_{hiv} + T_{hvv'}\right) x_{hji}^m \leq a_{iv'}^m \qquad \forall \left(v, v'\right) \in S_{im}, \forall m \in M_i, \forall i \in V_c \tag{3.10}$$

$$a_{ic^i}^m \leq \sum_{h \in H} \sum_{j \in \delta^+(i)} W^U x_{hji}^m \qquad \forall m \in M_i, \forall i \in V_c \tag{3.11}$$

$$\sum_{m \in M_i} a_{ip}^m = \sum_{l \in T_p} w_{pl} u_{ipl} \qquad \forall p \in P_i, \forall i \in V_c \tag{3.12}$$

Travel Time Constraints. Let $\tilde{a}_{ij}$ be a non-negative variable that represents the loading start time at node j if a truck traverses arc $(i,j) \in A$ and 0 otherwise, as shown by constraints (3.13) and (3.14). Constraints (3.15) build the relationship between $\tilde{a}_{ji}$ and $a_{iv^{im}}^m$. Constraints (3.16) relate the arrival time at location c^i to the arrival time at some node j when a truck traverses arc (i,j).

$$0 \leq \tilde{a}_{ij} \leq \sum_{h \in H} \sum_{m \in M_j} W^U x_{hij}^m \qquad \forall (i,j) \in A \tag{3.13}$$

$$\sum_{h \in H} \sum_{m \in M_j} \left(W^L + t_{h0j}^m\right) x_{h0j}^m \leq \tilde{a}_{0j} \qquad \forall j \in \delta^-(0) \tag{3.14}$$

$$\sum_{m \in M_i} a_{iv^{im}}^m = \sum_{j \in \delta^+(i)} \tilde{a}_{ji} \qquad \forall i \in V_c \tag{3.15}$$

$$\sum_{m \in M_i} a_{ic^i}^m + \sum_{h \in H} \sum_{j \in \delta^-(i)} \sum_{m \in M_j} t_{hij}^m x_{hij}^m \leq \sum_{j \in \delta^-(i)} \tilde{a}_{ij} \qquad \forall i \in V_c \tag{3.16}$$

Truck-order Compatibility Constraints. If a truck type h is not compatible with order $i \in V_c$, we forbid all relevant arcs from use, as shown by (3.17).

$$\sum_{j \in \delta^+(i)} \sum_{m \in M_i} x^m_{hji} + \sum_{j \in \delta^-(i)} \sum_{m \in M_j} x^m_{hij} = 0 \qquad \forall h \in H \setminus H_i, \forall i \in V_c \tag{3.17}$$

Symmetry-breaking Constraints. If order i is identical to order i' of a higher index (i.e., $i < i'$), then we can enforce the following symmetry-breaking constraints (3.18) - (3.20).

$$z_i \geq z_{i'} \tag{3.18}$$

$$\sum_{h \in H} \sum_{m \in M_{i'}} x^m_{hii'} = 0 \tag{3.19}$$

$$\sum_{j \in \delta^+(i)} \tilde{a}_{ji} \geq \sum_{j \in \delta^+(i')} \tilde{a}_{ji'} \tag{3.20}$$

Computing Overtime. Let y_{ih} denote the amount of overtime for a truck of type h that finishes its trip immediately after serving order $i \in V_c$. Since only work hour after W is counted as overtime, y_{ih} is bounded from above by $W^U - W$, as constraints (3.21) indicate. Constraints (3.22) properly computes the amount of overtime for the truck with $i \in V_c$ being its last serving order on a returning trip.

$$0 \leq y_{ih} \leq \left(W^U - W\right) \sum_{m \in M_0} x^m_{hi0} \qquad \forall h \in H, \forall i \in V_c \tag{3.21}$$

$$\sum_{h \in H} y_{ih} \geq \tilde{a}_{i0} - \sum_{h \in H} \sum_{m \in M_0} W x^m_{hi0} \qquad \forall i \in V_c \tag{3.22}$$

Objective Function. The objective is to minimize the total travel cost that consists of fixed costs, overtime costs, travel costs and bid costs. Let $\theta^m_{hj} := \sum_{(v,v') \in S_{jm}} C_{hvv'}$ denote the total cost for traversing sequence S_{jm} by a truck of type h. We define $\theta^m_{hj} := 0$ whenever S_{jm} only contains a single location. The objective function is thus defined as follows:

$$\sum_{h \in H} N_h f_h + \sum_{i \in V_c} \sum_{h \in H} \gamma_h y_{ih} + \sum_{h \in H} \sum_{(i,j) \in A} \sum_{m \in M_j} \left(c^m_{hij} + \theta^m_{hj}\right) x^m_{hij} + \sum_{i \in V_c} \Delta_i \left(1 - z_i\right) \tag{3.23}$$

The FTPDP is to minimize the objective (3.23) while decision variables are subject to constraints (3.1) - (3.22). We remark that this is a compact MILP formulation for the FTPDP and that this formulation can handle complex routing and scheduling features such as a heterogeneous fleet, full truckload shipments, multiple pickup points, optional orders and loading dock capacity restrictions.

3.4 COMPUTATIONAL STUDIES

In this section, we test our proposed MILP formulation on FTPDP benchmark instances. Our algorithm was implemented in C++ and the mixed-integer linear program was solved using the Gurobi Optimizer 9.0 through the C application programming interface. All Gurobi settings were kept being default. The experiments were run on an Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz. The 128 GB of available RAM was shared among 5 copies of the algorithm running in parallel on the server. Each instance was solved by one copy of the algorithm using a single thread.

We utilize the real-life data from the petrochemical company Braskem to generate benchmark instances. In particular, we consider the 3-month (in total 66 work days) operational data at a chosen region. A heterogeneous fleet, consisting of 3 types of trucks, is dedicated to serve orders on a daily basis. For each work day, orders that need to be served are known in advance and have a "today" due date. The number of orders ranges from 7 to 52 across 66 days, and there are on average 27.2 orders per day. We have in total 66 FTPDP benchmarks, numbered from 1 to 66 for convenience.

3.4.1 *Model Performance*

We first evaluate the computational performance of our proposed MILP formulation on solving real-life benchmarks. We impose a time limit of 3,600 seconds for each considered instance and present the computational results in Table 3.1. Columns "Inst." and "# orders" denote the instance name and the number of orders in this instance, respectively. The column "t (sec) [Gap]" reports the solution time (rounded to the nearest integer) if this instance was solved to optimality within the time limit; otherwise, it reports the residual gap (defined as $(UB - LB)/UB$) in bracket. We also report in column "Served (%)" the percentage of orders that are served in the returned best known solution. Our proposed MILP model solved all 66 benchmarks to optimality with the solution time ranging from 1 second to 1,793 seconds. We can claim that the formulation we proposed in Section 3.3 could efficiently solve practical FTPDP instances. A noticeable observation is that the percentage of served orders is only 47.5% on average. Next we aim to increase this number by allowing for pre-loading trucks, with the goal of reducing the total operational cost.

Table 3.1: Computational results for our MILP formulation on 66 real-life instances

Inst.	# orders	t (s) [Gap]	Served (%)	Inst.	# orders	t (s) [Gap]	Served (%)	Inst.	# orders	t (s) [Gap]	Served (%)
1	32	7	53.1	23	15	1	66.7	45	26	2	46.2
2	28	2	50.0	24	16	2	62.5	46	12	1	66.7
3	14	1	64.3	25	28	1	39.3	47	19	1	52.6
4	12	1	91.7	26	23	4	56.5	48	25	39	48.0
5	18	1	61.1	27	23	21	52.2	49	34	129	38.2
6	22	1	59.1	28	26	2	42.3	50	20	1	60.0
7	37	912	45.9	29	22	4	54.5	51	30	183	46.7
8	22	27	54.5	30	43	1,793	41.9	52	24	35	54.2
9	18	1	55.6	31	52	38	28.8	53	27	3	55.6
10	32	24	46.9	32	39	361	35.9	54	18	1	61.1
11	20	78	70.0	33	23	4	47.8	55	7	1	100.0
12	27	128	44.4	34	30	102	50.0	56	21	3	47.6
13	19	8	57.9	35	29	7	51.7	57	35	177	37.1
14	15	6	80.0	36	34	2	32.4	58	28	183	50.0
15	30	4	43.3	37	39	9	30.8	59	23	86	52.2
16	14	5	92.9	38	16	22	62.5	60	27	43	40.7
17	38	338	42.1	39	24	2	41.7	61	34	177	55.9
18	22	10	63.6	40	32	463	40.6	62	36	13	41.7
19	22	2	59.1	41	46	1,091	41.3	63	31	562	45.2
20	38	15	39.5	42	47	43	38.3	64	33	91	42.4
21	41	236	34.1	43	38	347	36.8	65	24	14	54.2
22	18	1	50.0	44	36	12	38.9	66	38	558	44.7

3.4.2 *Evaluating the pre-loading policy*

In the FTPDP application of our interest, a set of orders that will be served tomorrow, denoted by V_t, is usually known a few days ahead. We now allow for pre-loading trucks with orders $i \in V_t$ on their returning trip to the depot and call this a "pre-loading policy". Specifically, a truck is allowed to perform all pickups at the designated plants $p \in P_i$ for an order $i \in V_t$ on its returning trip "today" and will be scheduled to deliver a full truckload of product to the client c^i "tomorrow" right after it leaves the depot. In this section, we aim to quantify the economic effect of adopting the pre-loading policy. To achieve this, we first define three types of orders: pre-loaded orders V_p, optional orders V_o, and tomorrow orders V_t.

- Pre-loaded orders were picked up the day before, but have not yet been delivered; they will be delivered "today", right after trucks leave the depot.
- Optional orders have a "today" due date; they are ready to be picked up and delivered; if not picked up and delivered, they will be outsourced for a cost ("bid").
- Tomorrow orders have a "tomorrow" due date; they can be picked up right before a truck returns to the depot to end its shift; if so, they will become "pre-loaded" orders once we roll the horizon

To accommodate the above three types of orders into the MILP formulation proposed in Section 3.3, we make the following modifications.

- For order $i \in V_p$, we define $P_i := \emptyset, M_i := \{1\}$ and $S_{i1} := \langle c^i \rangle$. We impose constraints (3.24) to ensure that order $i \in V_p$ will always be the first served order by a truck of type h^i, which has picked up this order the day before.

$$\sum_{m \in M_i} x^m_{h^i 0 i} = 1 \qquad \forall i \in V_p \tag{3.24}$$

- For order $i \in V_t$, we change its client to be the depot (i.e., $c^i := 0$) and we fix the delivery time to be 0 (i.e., $\beta_{hi} := 0$). Constraints (3.25) enforce that order i will always be the last served order on a returning trip by some truck.

$$\sum_{h \in H} \sum_{m \in M_0} x^m_{hi0} = z_i \qquad \forall i \in V_t \tag{3.25}$$

- To encapsulate both pre-loaded orders and tomorrow orders, we redefine $V_c := V_p \cup V_o \cup V_t$.
- The last term of the objective (3.23) is modified to be $\sum_{i \in V_o} \Delta_i (1 - z_i) - \alpha \sum_{i \in V_t} \Delta_i z_i$, where $\alpha \geq 0$ is a hyper-parameter that will be tuned. Deducting the term $\alpha \sum_{i \in V_t} \Delta_i z_i$ from the objective is to incentivize the action of picking up tomorrow orders.

The resulting MILP formulation is to minimize the objective (3.23) with decision variables being subject to constraints (3.1) - (3.22), (3.24) and (3.25).

We now quantify the economic effect of adopting the pre-loading policy by choosing $\alpha \in \{0, 0.25, 0.50, 0.75, 1\}$. When $\alpha = 0$, the pre-loading policy is not allowed; we call this the base case. For each work day, a subset of today's orders are pre-loaded orders V_p and the remaining ones are optional orders V_o; orders from the next day become tomorrow orders V_t. In total, we can generate 66 instances. Note that $V_p := \emptyset$ for the first instance and $V_t := \emptyset$ for the last instance.

We impose a time limit of $3,600$ seconds for each instance and present the consolidated results in Table 3.2. The column "# inst." denotes the number of instances. We report in columns "# opt.", "Avg. t (sec)" and "Avg. gap (%)" the number of instances that were solved to optimality, the geometric mean solution time (rounded to the nearest integer) for those solved instances, and the average residual gap for those unsolved ones, respectively. We report in column "Served (%)" the percentage of served orders over 3 months. The column "Cost" reports the accumulative cost (excluding the incentivizing term) for 3 months. The cost is normalized with respect to the base case.

From Table 3.2, the vast majority of benchmark instances were solved to optimality while a small gap was returned to the unsolved ones. Hence, we can claim that allowing for pre-loading trucks does not complicate the solution of our proposed MILP formulation. When α increases, the percentage of served orders first increases and then decreases; the decrease was caused by the fact that optional orders are less likely to be served when the pickup of tomorrow orders is heavily incentivized (e.g., α is large). Correspondingly, the total cost first decreases and then increases. Choosing $\alpha = 0.25$ helps to save 2% of the total cost. Comparing with $\alpha = 0.25$, the order serving rate is higher but the cost saving is surprisingly lower when $\alpha = 0.50$, which was owing to the distinct bid costs for different orders. From the above analysis, we can claim that adopting the pre-loading policy can help to increase the percentage of served orders and hence reduce the operational cost.

Table 3.2: Computational results for evaluating the pre-loading policy

α	# inst.	# opt.	Avg. t (sec)	Avg. gap (%)	Served (%)	Cost
0	66	66	9	–	47.5	1.00
0.25	66	64	10	2.6	50.5	0.98
0.50	66	63	12	1.9	51.2	0.99
0.75	66	65	8	4.5	41.5	1.05
1	66	66	4	–	38.1	1.07

3.5 CONCLUSIONS

In this chapter, we study the full truckload pickup and delivery problem that is featured by a heterogeneous fleet, optional orders, multiple pickup points and loading dock capacity constraints. We propose a novel mixed-integer linear programming formulation to model this problem. The computational studies on 66 real-life benchmark instances demonstrate the effectiveness of our proposed model. We further incorporate the pre-loading policy into the proposed model and demonstrate the economic benefit of allowing for pre-loading trucks.

4

ESTIMATION OF MARGINAL COST TO SERVE INDIVIDUAL CUSTOMERS

In this chapter, we proposes a scenario-sampling framework to estimate the expected incremental routing cost required so as to incorporate a target customer into the stochastic supply chain network. The cost estimate is shown to converge to its true value with statistical guarantee as the number of samples increases, while the Hoeffding's inequality can be utilized to determine a sufficient sample size for a desired estimation accuracy. Inspired from a real-life setting arising in distribution of industrial gases, we sample instances of multi-depot vehicle routing problems with inter-depot routes, and we use these as scenarios towards a demonstration of the marginal cost estimation framework and towards a detailed study to elucidate the quality of our estimates. In order to solve such rich routing problems exactly, we also develop a tailored branch-price-and-cut algorithm, which is shown to be able to solve to optimality instances of up to 70 customers within reasonable time, significantly outperforming the previous state-of-the-art exact method.

4.1 INTRODUCTION

Supply chain logistics contribute a significant portion of total cost for many businesses, and thus receive a lot of effort for optimization. In the literature, lots of attention has been paid to optimize supply chain systems in strategic, tactical and operational levels, usually with the objective of obtaining designs that are the most economical in the long run. The most classic example is the one where, given a distribution center, a fleet of vehicles and a group of customers to be served, we aim to identify the minimum-cost routes for vehicles to traverse, such that customer demands are satisfied and relevant system constraints, e.g., vehicle capacities, time windows, and route duration limits, among others, are respected. A wide variety of vehicle routing problems have been addressed in the literature by both exact and heuristic approaches [167]. However, little effort has to-date

been devoted to understanding the marginal cost of serving an individual customer, i.e., the incremental cost of delivering to an extra customer on top of the current supply chain design. This problem is of great significance in the following business situations: (i) the distributor wants to estimate the customer lifetime value, in order to distinguish profitable customers; (ii) the distributor wants to understand the expected marginal cost of serving a new customer for pricing purposes. We highlight that our focus in this chapter is on *cost estimation*, which can be informative in the context of selecting customer portfolios and/or designing updates to the distribution network; we do not consider *cost allocation* of past operations, which is often sought for purposes of fair accounting.

Estimating the incremental distribution cost of serving a specific customer is challenging, mainly because of the following two reasons. First, the extra cost emanates from the change in routing due to the additional service that must be performed, and thus, identifying the marginal cost in a deterministic setting usually entails solving two $\mathcal{NP}$-hard vehicle routing problems (one accounting for the customer in question, and one not involving that customer). Second, the supply chain network of interest is intrinsically stochastic. For example, not all potential customers request a visit on the same day. Furthermore, the amount of goods to be delivered to customers usually varies on a daily basis. Hence, on different days the distributor might face completely different delivery challenges, as it aims to implement least-cost routing plans that often possess little geographical similarity. In this setting, the addition of a new customer may on certain days be "almost free" because the new customer "fits well" with otherwise optimal routes, but may be completely cost-prohibitive on other days when, for example, the new customer is far away from the existing customers or when the demand of the new customer cannot be served as "balance load" of existing routes. This leads to the situation where cost allocation to a specific customer in a long term becomes much more perplexing, having to account for such intrinsic stochasticity.

Cost estimation in a stochastic supply chain system is an interesting problem and was approached in the literature mainly from a machine learning perspective. The work of [72] studied approximations to the average travel distance of vehicle routing problems with varying numbers of customers, demands and locations. The authors proposed six approximations with the total number of customers, the customer dispersion area, the average distance between customers and the depot, and the number of vehicles being input variables, and then utilized linear regression to estimate parameters. In [168], the authors focused on analyzing the effect of the geographical dispersion of potential customers on the expected routing cost of the distribution system and derived four estimates using

continuous distance approximation [59]. The work of [107] considered the prediction of the cost to serve new customers in the industrial gas business and addressed this problem by first applying a supervised learning technique to group customers based on their features and then utilizing a linear regression model to forecast the distribution cost. A similar approach was adopted in the work of [162]. The authors first identified seven important customer features including neighborhood density, latitude and longitude, among others, and then utilized those to build a predictive model. The work of [129] considered the inventory routing cost allocation problem in a deterministic setting from a cooperative game theory perspective and proposed four cost allocation mechanisms to determine a cost-to-serve for customers.

In this chapter, we propose a novel scenario-sampling approach that aims to rigorously account for the intrinsic stochasticity in the supply chain system. The basic idea is that the expected value of a given function can be approximated by a sample average estimate derived from random samples. This approach has been widely used for histogram construction [48], function estimation [80], and stochastic optimization [106], among many other settings. In particular, the work of [106] studied a stochastic optimization problem with its objective being an expected function. The authors proposed to approximate the expected objective by its corresponding sample average function and then showed that, with probability approaching one exponentially fast with increase of the sample size, an optimal solution of the sample average approximation problem provides an exact optimal solution of the "true" problem. In our context, the objective is to identify the expected marginal cost of serving an extra customer in a stochastic supply chain network. Our scenario-sampling framework first generates representative daily delivery scenarios that can simulate the stochasticity in customer presence and demands, and it then quantifies the incremental cost of delivering to the target customer in each such deterministic scenario. The sample average is then returned as an estimate of the expected marginal cost. We estimate the number of required scenarios that need to be sampled by utilizing the Hoeffding's inequality [92], which statistically guarantees that we identify a good estimate of the true marginal routing cost. We highlight that, in contrast with previous works on distribution cost estimation which are mainly based on machine learning principles and lack statistical guarantees [72, 107, 162, 168], our approach combines classic probability theory with exact vehicle routing techniques and thus provides statistical insurances for the estimation accuracy.

In order to quantify the extra cost of serving a specific customer in each sampled scenario, we need to solve routing problems. The routing application we face is a distribution

problem arising in the industrial gas business where trucks can be replenished at intermediate production plants and each truck usually performs several trips during a day shift. This setting can be mapped to the *multi-depot vehicle routing problem with inter-depot routes* (MDVRPI) [58]. In our context, we define a *trip* to be a sequence of nodes with "depots" only being its starting and ending nodes, and we define a *route* to be a feasible combination of trips assigned to a vehicle. The MDVRPI is a variant of the *multi-depot vehicle routing problem* in which vehicles are allowed to be replenished at intermediate depots along their routes, while the route duration limit is respected. This problem was formally defined in the work of [58], in which the authors encountered a real-life application of the MDVRPI in a grocery distribution problem. Other important applications arise in municipal service, especially in waste collection problems [12, 43, 105], where vehicles have to renew their capacities by uploading the waste at one of the treatment plants and return to the depot only when the work shift is over. Interested readers are referred to the recent work of [150] for a comprehensive review on routing problems with intermediate stops and their applications.

In the literature, the scientific research on solving the MDVRPI is dominated by heuristic methods [43, 58, 164]. In the work of [58], the authors proposed a three-phase methodology: (i) adaptive memory and tabu search are applied to generate a set of routes; (ii) an integer program based on the set-partitioning formulation is executed to determine a minimum-cost routing design; (iii) a post-optimization phase is performed in an attempt to improve the solution. The authors of [164] renamed the MDVRPI to be the *vehicle routing problem with intermediate replenishment facilities* to emphasize the use of a single central station for the fleet, and they proposed a three-step meta-heuristic algorithm in which tabu search and variable neighborhood search are combined to improve the solution quality after the construction heuristic and a guided local search, combined with a customer removal and reinsertion procedure, is applied to produce the final solution. Their approach was tested on the set of 12 benchmark instances used in [58] and generated 6 new best known solutions. Finally, in [43], the authors considered a waste collection vehicle routing problem with time windows and proposed an adaptive large neighborhood search method.

To the best of our knowledge, the only exact approach for solving the MDVRPI is a branch-and-price algorithm proposed in [127]. The authors modeled the MDVRPI as a *set covering* formulation in which variables are routes corresponding to feasible combinations of trips, and they proposed two pricing subproblems to generate routes. The first one generates routes directly by solving an *elementary shortest path problem with resource constraints* (ESPPRC), while the second one first enumerates all non-dominated trips by

solving an ESPPRC and then combines trips into routes by exploiting the relationship between the sets of trips and routes. Their computational studies showed that the second pricing mechanism performs better and their branch-and-price algorithm could solve MDVRPI benchmark instances involving up to 50 customers to optimality.

Whereas satisfactory for instances of the demonstrated sizes, the enumeration step becomes too prohibitive to execute when the number of customers grows larger, and hence, in this chapter we adopt the first pricing mechanism, which we evolve with several modifications. Furthermore, we develop a *branch-price-and-cut* (BPC) algorithm for exactly solving the MDVRPI. Our BPC algorithm has the following distinctive features from the work of [127]: (i) we enforce only partial elementarity and solve a *shortest path problem with resource constraints* (SPPRC) to generate routes with negative reduced costs, given the fact that the ESPPRC is strongly $\mathcal{NP}$-hard, while the relaxed one is still $\mathcal{NP}$-hard but in a weak sense [68, 138]; (ii) we enhance it with several state-of-the-art pricing techniques, including *ng*-routes, variable fixing and routing enumeration; and (iii) we incorporate rounded capacity inequalities and limited-memory subset row cuts as strengthening constraints.

Finally, we remark that the MDVRPI generalizes another well-studied problem, namely the *multi-trip vehicle routing problem* (MTVRP).[1] The MTVRP extends the classical vehicle routing problem inasmuch as each vehicle is allowed to perform several trips, subject to route duration constraints. Clearly, the MDVRPI reduces to the MTVRP when the number of depots is equal to one, and consequently, our proposed BPC algorithm can also be applied to solve the MTVRP as well as a number of its variants.

In summary, the distinct contributions of our work are as follows.

- We propose a scenario-sampling framework to estimate the expected marginal cost of serving individual customers. Specifically, we obtain independent scenarios by sampling customer demands from their distribution and consider the sample average as an estimate of the cost. We prove that our proposed framework provides statistical guarantee of the estimation accuracy, provided that a sufficiently large–informed by Hoeffding's inequality–sample size has been obtained.

- We model the MDVRPI as a set partitioning formulation and propose a branch-price-and-cut algorithm that incorporates several state-of-the-art techniques, including *ng*-routes, variable fixing, route enumeration, and limited-memory subset row cuts, among others.

1 In the literature, the MTVRP is also named as the *vehicle routing problem with multiple use of vehicles* and it has been well studied by both heuristic and exact approaches. Interested readers are referred to [47] for a comprehensive review on the MTVRP and its variants.

- We conduct computational studies to show that our branch-price-and-cut algorithm significantly outperforms the state-of-the-art exact approach for the MDVRPI. In particular, our algorithm proves optimality for all previously open MDVRPI benchmark instances involving up to 40 customers. We further push the envelope by extending the literature benchmarks with 70-customer instances and solving to proven optimality the vast majority of those as well.
- We demonstrate the scenario-sampling framework and the quality of its cost estimates, utilizing thousands of MDVRPI instance samples in each case, and we elucidate the effect on the marginal routing cost of factors such as customer locations and demand levels.

The remainder of this chapter is organized as follows. In Section 7.2, we provide a formal problem definition. In Section 4.3, we propose our scenario-sampling framework and prove its validity through the lens of probability theory. In Section 4.4, we discuss the implementation details of our proposed BPC algorithm. Section 7.5 presents computational results on the BPC algorithm's performance as well as a detailed analysis of the marginal cost estimation framework. Finally, we conclude our work with some remarks in Section 7.6.

4.2 PROBLEM DEFINITION

Consider a supply chain network where n customers with stochastic demands might be served on a daily basis. Let $\xi \in \mathbb{R}^n_{\geq 0}$ denote the corresponding demand vector and we assume that ξ is drawn from some given probability distribution, e.g., $\xi \sim \Xi$. We emphasize that a customer's demand can be zero on a given day, which indicates that this customer does not request a visit. Now, let us assume that the distributor wants to serve an "additional" customer[2] whose demand, denoted by $\vartheta \in \mathbb{R}_{>0}$, is also stochastic and is drawn from some given probability distribution, e.g., $\vartheta \sim \Theta$. We further assume that ξ and ϑ are independent and that, for any $(\xi, \vartheta) \sim \Xi \times \Theta$, the resulting routing problem is always feasible; that is, there always exists a feasible routing plan that can be implemented, such that customer demands are satisfied and applicable system constraints are respected. The distributor aims to figure out how much marginal cost will be incurred on average when serving the additional customer with demand ϑ. In other words, the objective is to quantify the expected incremental cost of incorporating the additional customer into the current supply chain network, while all other customer demands are stochastic. We do not

2 This customer may be either an existing or a prospective one.

describe the specific routing setting here, since our proposed framework is generic and remains valid irrespective of this aspect. A detailed definition of the routing problem of interest to us is deferred to Section 4.4.

4.3 PROPOSED FRAMEWORK

In this section, we first properly define the marginal cost and then present our scenario-sampling framework to estimate it. We also prove the validity of this framework using probabilistic arguments, as well as we present a detailed procedure for its deployment.

4.3.1 *Marginal Cost Estimation*

Consider a scenario $(\xi, \vartheta) \sim \Xi \times \Theta$. The marginal cost under this specific scenario is defined as

$$\texttt{MC}\,(\xi, \vartheta) := \frac{\texttt{VRP}\,(\xi, \vartheta) - \texttt{VRP}(\xi, 0)}{\vartheta}. \tag{4.1}$$

Here, for a given supply chain network with customer demands fixed to be ξ, $\texttt{VRP}(\xi, 0)$ and $\texttt{VRP}\,(\xi, \vartheta)$ denote the optimal routing costs before and after serving an extra customer with demand ϑ, respectively. Note that the marginal cost in our definition is normalized by ϑ, representing the incurred incremental cost per unit of extra demand served.[3] Also note that, with the above definition, we make no assumption regarding the sign of the marginal cost. In fact, although in the vast majority of settings the marginal cost is non-negative, i.e., one cannot save costs by performing more service, it is conceivable that, due to some special contract activation or some other synergy between the extra customer with the rest of customers to be served, the marginal cost attains a negative value.

In any case, it is important to emphasize that the optimal routing costs in the numerator of equation (4.1) have to be computed from exact approaches to appropriately formulated vehicle routing problems. Despite being much cheaper to compute via efficient (meta-)heuristic algorithms, heuristic solutions cannot be used in the definition of the marginal cost, since their lack of optimality guarantee might result in a comparison between optimal and suboptimal costs, causing $\texttt{MC}\,(\xi, \vartheta)$ to attain a misleading value.

The goal of this chapter is to estimate $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\,(\xi, \vartheta)\right]$ numerically, which can be very challenging to compute exactly. First, there is no closed-form formula for $\texttt{MC}\,(\xi, \vartheta)$ for a given (ξ, ϑ), since it entails solving two $\mathcal{NP}$-hard routing problems, in general. Second,

3 A sampled ϑ value is always strictly positive, since a zero demand would be meaningless in this context.

(ξ, ϑ) is a high-dimensional vector, and thus, computing $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right]$ exactly would entail high-dimensional integration, which is impractical.

Let $(\xi^1, \vartheta^1), (\xi^2, \vartheta^2), \ldots, (\xi^N, \vartheta^N)$ be an independent and identically distributed random sample of N realizations of the demand vector $(\xi, \vartheta) \sim \Xi \times \Theta$. For each sample $\left(\xi^i, \vartheta^i\right)$, we first solve two $\mathcal{NP}$-hard vehicle routing problems exactly to obtain $\texttt{VRP}\left(\xi^i, 0\right)$ and $\texttt{VRP}\left(\xi^i, \vartheta^i\right)$ and then compute the specific sample's marginal cost, $\texttt{MC}\left(\xi^i, \vartheta^i\right)$, using equation (4.1). We consider the average of $\texttt{MC}\left(\xi^i, \vartheta^i\right)$ over N samples as our estimate of the expected marginal cost. The sample average estimate, $\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right)$, coincides with the true value, $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right]$, with very high probability, when the sample size N is large enough. See Proposition 4.1.

Proposition 4.1. *Given any $\delta > 0$,*

$$\lim_{N\to+\infty} \text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right) - \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right]\right| > \delta\right) = 0. \tag{4.2}$$

Proof. Since $(\xi^1, \vartheta^1), (\xi^2, \vartheta^2), \ldots, (\xi^N, \vartheta^N)$ are independent and identically distributed, and since $\texttt{MC}\left(\xi, \vartheta\right)$ is a function of (ξ, ϑ), then $\texttt{MC}\left(\xi^1, \vartheta^1\right), \texttt{MC}\left(\xi^2, \vartheta^2\right), \ldots, \texttt{MC}\left(\xi^N, \vartheta^N\right)$ are also independent and identically distributed. By the *weak law of large numbers*, the average of independent and identically distributed random variables converges in probability to the expectation; that is,

$$\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right) \xrightarrow{p} \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right], \quad \text{when } N \to +\infty.$$

The above implies that (4.2) holds for any specified margin $\delta > 0$, no matter how small. □

We remark that, since all N samples are randomly chosen, the sample average converges to the true value in a probabilistic sense, but not in a deterministic one. Whereas the convergence within a permissible deviation, δ, is only guaranteed to occur when the sample size N approaches infinity, in practice one has to settle with using finitely many samples. In the following section, we seek to determine a finite sample size that would be deemed sufficient to yield an estimate of high quality.

4.3.2 *Bounding the Sample Size*

The immediate question after Proposition 4.1 is how fast is the convergence, namely how fast does the probability defined in equation (4.2) decrease with the increase in sample size N. We address this question in Proposition 4.2.

Before we present it, we introduce two quantities, $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$, to denote valid lower and upper bounds for $\texttt{MC}(\xi, \vartheta)$, respectively; that is, $\texttt{MC}(\xi, \vartheta) \in [\underline{\texttt{MC}}, \overline{\texttt{MC}}]$, for any $(\xi, \vartheta) \sim \Xi \times \Theta$. The boundedness of $\texttt{MC}(\xi, \vartheta)$ is ensured by the following two observations: (i) $\texttt{VRP}(\xi, \vartheta)$ and $\texttt{VRP}(\xi, 0)$ are both finite, due to the feasibility assumption from Section 7.2; and (ii) in practice, ϑ is bounded from below by some positive value, since the target customer has to order some minimum amount of product each time it requires service.

We remark that the tightest bounds for $\texttt{MC}(\xi, \vartheta)$ correspond to the solution of the following optimization problems:

$$\min/\max_{(\xi,\vartheta)\sim\Xi\times\Theta} \texttt{MC}(\xi, \vartheta) \tag{4.3}$$

However, since $\texttt{MC}(\xi, \vartheta)$ does not assume an a-priori known form, solving (4.3) is generally intractable. Fortunately, one can usually deduce practically-relevant, valid bounds for $\texttt{MC}(\xi, \vartheta)$. As we pointed out earlier, in the vast majority of applications, the marginal cost is always non-negative. In that case, one can choose $\underline{\texttt{MC}} = 0$ as a safe lower bound. On the other hand, a safe upper bound $\overline{\texttt{MC}}$ can be the normalized cost of performing as many dedicated round-trips as necessary to deliver the least allowable amount of product to the target customer. In this calculation, penalties could also be incorporated for the use of more-than-available vehicles, to reflect overtime and/or outsourcing costs usually incurred in such situations.

Assume now $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$ can be deduced for the routing problem of interest. Proposition 4.2 applies.

Proposition 4.2. *Given any $\delta > 0$, then*

$$\text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right) - \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}(\xi, \vartheta)\right]\right| > \delta\right) \leq 2\exp\left(-\frac{2N\delta^2}{\left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)^2}\right). \tag{4.4}$$

Proof. Since $\texttt{MC}\left(\xi^i, \vartheta^i\right)$ are independent random variables such that $\texttt{MC}\left(\xi^i, \vartheta^i\right) \in [\underline{\texttt{MC}}, \overline{\texttt{MC}}]$, for all $i = 1, 2, \ldots, n$, applying *Hoeffding's inequality* [92] yields

$$\text{Prob}\left(\left|\frac{1}{N}\sum_{i=1}^{N}\texttt{MC}\left(\xi^i, \vartheta^i\right) - \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\xi^i,\vartheta^i}\left[\texttt{MC}\left(\xi^i, \vartheta^i\right)\right]\right| > \delta\right) \leq 2\exp\left(-\frac{2N^2\delta^2}{\sum_{i=1}^{N}\left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)^2}\right),$$

noting that the right hand side can be simplified by a factor of N. Furthermore, since $\texttt{MC}\left(\xi^i, \vartheta^i\right)$ are also identically distributed, we have that

$$\mathbb{E}_{\xi^i,\vartheta^i}\left[\texttt{MC}\left(\xi^i, \vartheta^i\right)\right] = \mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right] \text{ for all } i = 1, 2, \ldots, n.$$

Hence, the above inequality results into (4.4), which completes the proof. □

Proposition 4.2 shows us that, as the sample size N increases, the probability that our estimate deviates from the true value more than some permissible value, δ, decreases. In fact, the probability that the sample average estimate approaches the "true" value of $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right]$ increases exponentially fast with the sample size N.

Proposition 4.2 also implies a way to derive the sample size for a desired accuracy. Let us choose a probability threshold, $\alpha \in (0, 1)$, and use (4.4) to deduce the sample size N necessary for the probability to be at most α. By requiring that the right-hand side of (4.4) be less than or equal to α, we obtain that

$$N \geq \frac{\left(\overline{\texttt{MC}} - \underline{\texttt{MC}}\right)^2}{2\delta^2}\ln\left(\frac{2}{\alpha}\right). \tag{4.5}$$

A key observation from (4.5) is that the sufficient sample size depends logarithmically on the probability threshold α, causing N to increase only mildly as α decreases. In contrast, we observe that the sample size should grow polynomially as a stricter permissible deviation δ is required.

4.3.3 *A General Framework*

Our scenario-sampling framework works as follows. We first choose a desired permissible deviation, $\delta > 0$, and a probability threshold, $\alpha \in (0, 1)$. Then, we deduce valid values for $\underline{\texttt{MC}}$ and $\overline{\texttt{MC}}$, and using inequality (4.5), we identify a bound on the sample size, denoted by

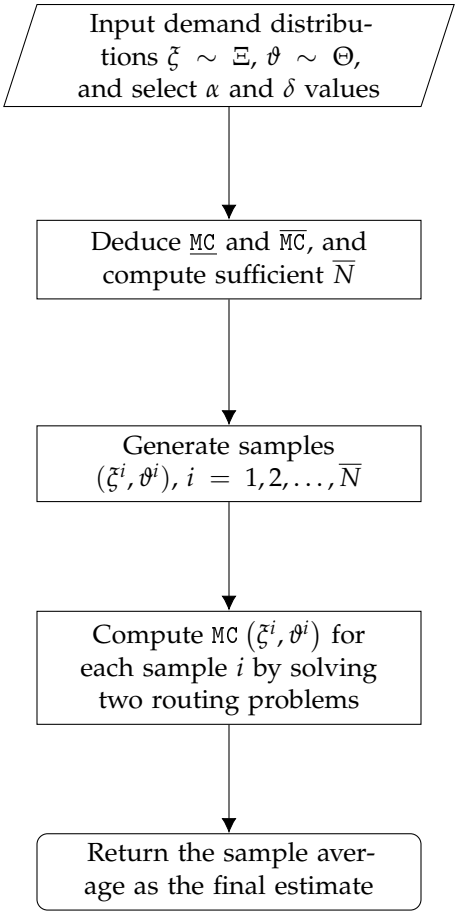


Figure 4.1: A scenario-sampling framework

$\overline{N}$. We then sample $\overline{N}$ scenarios, each having customer demands (ξ, ϑ) from the demand distribution $\Xi \times \Theta$.[4]

For each deterministic scenario, we solve two routing problems exactly to obtain $\mathtt{VRP}(\xi, 0)$ and $\mathtt{VRP}(\xi, \vartheta)$, and then calculate the marginal cost $\mathtt{MC}(\xi, \vartheta)$ using the formula (4.1). We finally compute the sample average $\frac{1}{\overline{N}} \sum_{i=1}^{\overline{N}} \mathtt{MC}(\xi^i, \vartheta^i)$ as the estimate of the true value $\mathbb{E}_{\xi,\vartheta}[\mathtt{MC}(\xi, \vartheta)]$. The overall framework is illustrated in Fig. 4.1.

Our proposed framework possesses a number of notable features. Firstly, the estimated value will always converge to the true marginal cost with probabilistic guarantee. Secondly, the framework only assumes that the customer demand distributions Ξ and Θ are given, but it does not require that they belong to any particular probability distribution family,

4 We remark that a perfect description of the demand distribution Ξ is seldom available in practice, but distributors often keep historical delivery data for each customer. This data can be used to extract the distribution using many well-known parametric and non-parametric approaches. In this chapter, we do not explore these techniques but refer interested readers to [170] for details.

such as Guassian or any other distribution commonly used in similar contexts. Finally, the framework is generic in terms of the type of routing problems one faces, and its validity does not depend on the type of algorithm one uses to obtain provably optimal cost evaluations.

4.4 SOLVING ROUTING PROBLEMS

We now turn our attention to discussing the exact approach for solving the routing problems of interest. In this chapter, we focus on the *multi-depot vehicle routing problem with inter-depot routes*, which extends the multi-depot vehicle routing problem inasmuch as the vehicles are allowed to stop at intermediate depots for replenishment. The examined problem was first introduced in the work of [58]. Note that, in the work of [164], the authors proposed an alternative name, the *vehicle routing problem with intermediate replenishment facilities*, to emphasize both the replenishment role of the intermediate facilities and the use of a single central station for the fleet of vehicles. We follow the work of [58] and use MDVRPI to denote the routing problem of our interest.

Before presenting the exact algorithm, we first formally define the MDVRPI. Let a directed graph $G = (V, A)$, where $V := V_c \cup V_d$ denotes the set of nodes, consisting of customers $V_c := \{1, 2, \ldots, n\}$ and depot facilities $V_d := \{n+1, n+2, \ldots, n+m\}$, and $A := \{(i,j) : i \in V_d, j \in V_c\} \cup \{(i,j) : i \in V_c, j \in V \setminus \{i\}\}$ denotes the set of arcs. Let $c_{ij} \in \mathbb{R}_{\geq 0}$ and $t_{ij} \in \mathbb{R}_{\geq 0}$ represent the cost and time, respectively, for a vehicle to traverse arc $(i,j) \in A$. We assume that the depot facilities have an unlimited supply of goods. A homogeneous fleet of K vehicles of capacity $Q \in \mathbb{R}_{>0}$ is available, with the exact allocation of these vehicles to the depots to be determined by the optimizer. Each customer $i \in V_c$ is associated with a demand $q_i \in \mathbb{R}_{>0}$ and a service duration $s_i \in \mathbb{R}_{\geq 0}$, while a vehicle docking time $\tau_i \in \mathbb{R}_{\geq 0}$ applies at each depot $i \in V_d$.[5] In the MDVRPI, a feasible vehicle route starts from and ends at the same depot, but vehicles are allowed to stop at any depot to replenish and continue with another trip. The vehicle capacity Q has to be respected during each trip, while the total duration of the route cannot exceed a predefined limit $T \in \mathbb{R}_{\geq 0}$. The objective is to identify the minimum-cost set of routes for the fleet of available vehicles to traverse so that each customer is visited exactly once with its demand served, while vehicle capacities and route duration limits are respected.

5 Without loss of generality, customer service durations and depot docking times can be suitably incorporated into the travel times along the arcs; moving forward, we always do so for convenience.

4.4.1 *Set Partitioning Model*

A sequence of vertices $\{i_1, i_2, \ldots, i_p\}$ visited by a vehicle is called a *feasible route*, if $i_1 = i_p \in V_d$, $(i_l, i_{l+1}) \in A$ for $1 \leq l \leq p-1$, and the following conditions are satisfied: (i) each customer vertex is visited at most once (elementary); (ii) the vehicle capacity Q is respected in each trip; (iii) the total route duration does not exceed T. Let R denote the set of all feasible routes and let c_r denote the cost of traversing route $r \in R$ by a vehicle. Let the parameter δ_{ir} denote the number of times customer $i \in V_c$ is covered in route $r \in R$. Let λ_r be a binary variable indicating whether route $r \in R$ is selected in the optimal solution. The MDVRPI can be formulated as the following *set partitioning* model (5.8)–(5.11).

$$\underset{\lambda_r}{\text{minimize}} \quad \sum_{r \in R} c_r \lambda_r \tag{4.6}$$

$$\text{subject to} \quad \sum_{r \in R} \delta_{ir} \lambda_r = 1 \qquad \forall i \in V_c \tag{4.7}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{4.8}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R \tag{4.9}$$

The objective function (5.8) is to minimize the total cost for selected routes. The degree constraints (5.9) guarantee that every customer is served exactly once, while the fleet size constraint (4.8) enforces that at most K vehicles are used. Finally, constraints (5.11) enforce binarity of variables.

It is well-known that one can relax the feasible space of the above set partitioning model by including non-elementary vehicle routes in R without sacrificing optimality. In our implementation, we replace R with the set of so-called *ng*-routes, $R^{ng} \supseteq R$, which are not necessarily elementary [26]. Since there exist exponentially many *ng*-feasible routes, the formulation (5.8)–(5.11) is a mixed-integer linear programming model with a very large number of binary variables. As a result, the linear programming (LP) relaxations at each node of the branch-and-bound tree are of large sizes such that they have to be tackled via an efficient *column generation* method. Valid inequalities are dynamically separated and added so as to strengthen the LP relaxations, yielding a *branch-price-and-cut* algorithm to solve the set-partitioning model. Details on this algorithm are presented below.

4.4.2 *Branch-Price-and-Cut Algorithm*

In the BPC algorithm, the LP relaxations at every node in the branch-and-bound tree are solved via column generation, while cutting planes are added to strengthen the relaxations. Our implementation incorporates several elements of the algorithm described in the work of [131], including *ng*-routes [26], variable fixing [98], route enumeration [25], and limited-memory subset row cuts [133], among other features. In what follows, we highlight only the most important of these ingredients; for more details, we refer readers to the works of [130], [131], and [147].

We first replace the binarity constraints (5.11) in the set partitioning model by non-negativity constraints and obtain its LP relaxation, which is usually referred to as the *master problem*. The master problem has a very large number of variables, as exponentially (to number of customers) many feasible routes are typically available. Generating all of these routes to explicitly define the master problem is obviously impractical, thus we resort to column generation. Given the premise that most of the variables will be non-basic, and assuming non-zero values in the optimal solution, only a subset of variables need to be considered in practice when solving the problem. Column generation leverages this idea and aims to generate only the variables that have the potential to improve the objective function, i.e., variables with negative reduced costs. In our context, columns correspond to *ng*-feasible routes, and we will use these terms interchangeably. We first consider a *restricted master problem* (RMP) defined by a subset of *ng*-routes $\tilde{R}^{ng} \subseteq R^{ng}$. After optimizing the RMP, columns with negative reduced costs will be appended to $\tilde{R}^{ng}$ and the resulting RMP will be reoptimized. This procedure iterates until no such column exists. In that case, we say that column generation converges and the master problem has achieved optimality. If at that point the solution to the master problem is fractional, we separate and add valid inequalities, or resort to branching. The overall BPC algorithm is illustrated in Fig. 6.6.

4.4.2.1 *Initialization*

To start the algorithm, we utilize single-customer routes to build $\tilde{R}^{ng}$, which are used to define the initial RMP. We highlight that, by constructing $\tilde{R}^{ng}$ in this way, the initial RMP is not necessarily guaranteed to be feasible, e.g., due to the fleet size constraint (4.8). If that is the case, we can seek for a feasible RMP via the "feasibility-recovery" step [71], which will be discussed in Section 4.4.2.5.

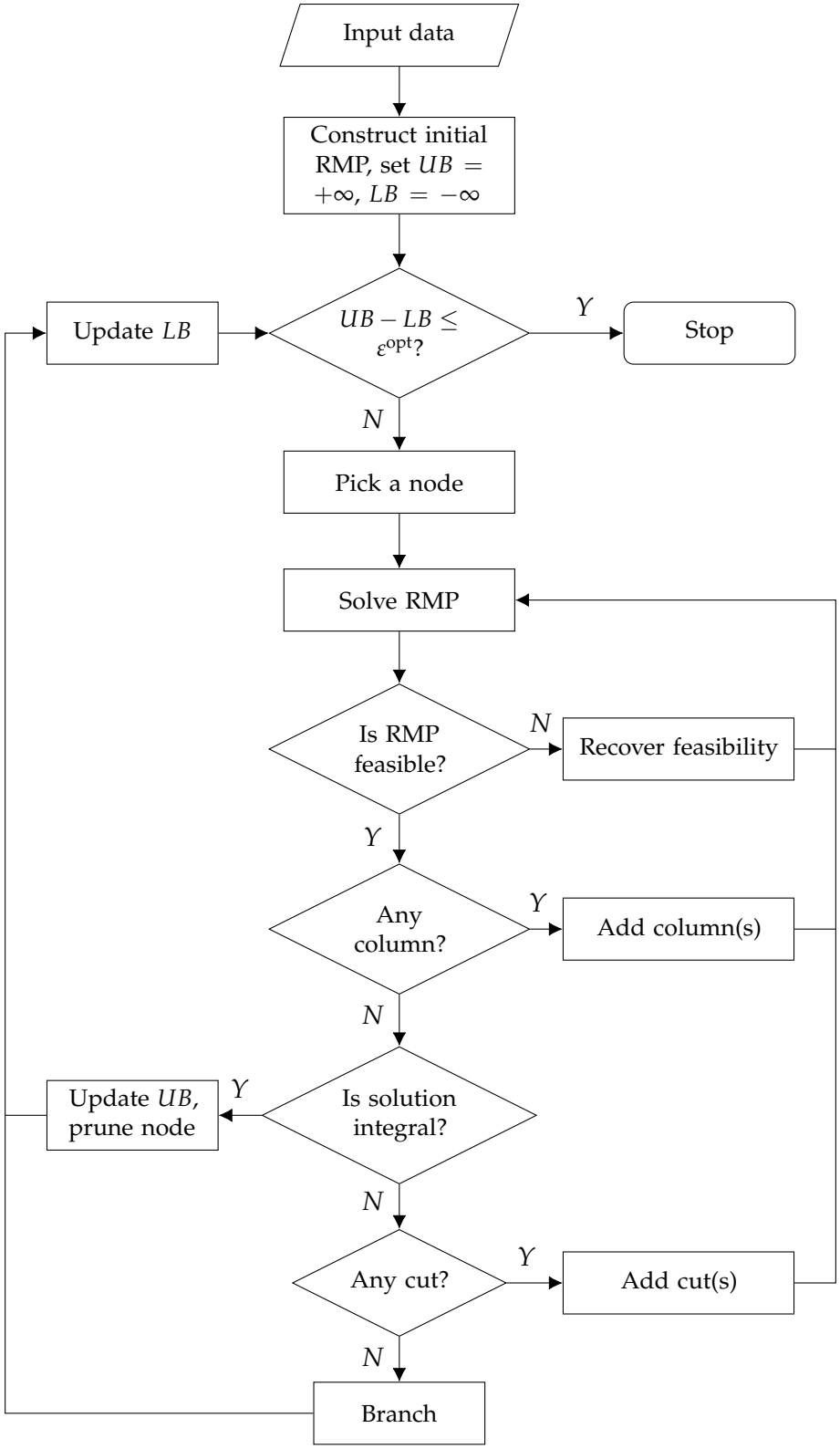


Figure 4.2: The Branch-Price-and-Cut algorithm.

4.4.2.2 *Solving Pricing Subproblems*

After the RMP is solved, we need to check whether it is necessary to enlarge $\tilde{R}^{ng}$ to include some neglected *ng*-feasible routes so as to improve the objective value of the RMP. This entails identifying columns with negative reduced costs, which is achieved by solving *pricing subproblems*. Since vehicle routes in R^{ng} can be categorized by their start depots, we consider to solve a pricing subproblem for each depot $i \in V_d$. Without loss of generality, let vertex $n+1$ denote the depot of interest.

In our context, the pricing subproblem can be modeled as a *shortest path problem with resource constraints* (SPPRC) [138]. The SPPRC can be defined on a directed graph $\overline{G} = (\overline{V}, \bar{A})$, where $\overline{G}$ is obtained by inserting into G two copies of the chosen depot, 0 and $n+m+1$, to represent a virtual start depot and a virtual destination depot, respectively, and by adding edges $(0, n+1)$ and $(i, n+m+1)$ for $i \in V_c$. Clearly, in such a configuration, a feasible route always starts from vertex 0 and ends at $n+m+1$, the chosen depot vertex $n+1$ being the immediate vertex after the starting node, and all depot vertices $i \in V_d$ used as intermediate replenishment facilities. Associated with each arc $(i,j) \in \bar{A}$ are the cost $\bar{c}_{ij}$, travel time t_{ij} and demand q_j. Note that $q_j = 0$, if vertex j corresponds to a depot. The cost $\bar{c}_{ij}$ is obtained by properly modifying c_{ij} in order to account for the contribution from current dual values to constraints (5.9) and (4.8). We associate each vertex $i \in \overline{V}$ with the load limit Q and the route duration limit T. While arc $(i,j) \in \bar{A}$ is traversed by a path, denoted by P, time resource t_{ij} and capacity resource q_j are consumed. A resource's accumulated consumption until a vertex visited along a path should not exceed its limit. We highlight that, if vertex j represents some intermediate depot facility, then the load of path P is reset to 0 to effectuate the replenishment process. The goal is to determine the minimum-cost path among all paths that start from the vertex 0 and end at the vertex $n+m+1$, such that all resource constraints are respected.

EXACT PRICING: It is well-known that the SPPRC is $\mathcal{NP}$-hard [68]. The most successful solution approach is a dynamic programming method called the *labeling algorithm*, which has a pseudo-polynomial time complexity [138]. The labeling algorithm works as follows. We associate an *ng*-feasible partial path P with a label, $L(P) = (\bar{c}(P), v(P), d(P), t(P), \Pi(P), pred(P))$, which stores the reduced cost, end vertex, load, duration, a set of forbidden vertices, and a pointer to its predecessor label. For a given pair (i, t') with $i \in \overline{V}$ and integer t' such that $0 \leq t' \leq T$, we define a bucket $B(i, t')$. A label $L(P)$ is stored in bucket $B(v(P), t(P))$. We remark that, if any entry of the travel time matrix t_{ij} is not integral, we then enforce an extra resource constraint in the SPPRC. In

particular, we associate arc $(i, j) \in \bar{A}$ and vertex $i \in \overline{V}$ with the integral travel time $\lfloor t_{ij} \rfloor$ and route duration limit $\lfloor T \rfloor$, respectively. We then use the integral travel time duration as our "reference" resource to store labels.

The labeling algorithm is initialized by storing a label $(0, 0, 0, 0, \emptyset, null)$ in bucket $B(0, 0)$. From bucket $B(0, 0)$, we use dynamic programming to fill other buckets, starting with lower values of t'. For each integer t' with $0 \leq t' \leq T$, the algorithm goes through $i \in \overline{V}$ and, for each neighbor j such that $(i, j) \in \bar{A}$, evaluates the extension of the walk represented by $B(i, t')$ to j. Given a label $L(P)$ from bucket $B(v(P), t(P))$, we now attempt to extend it to vertex $j \in \overline{V} \setminus \Pi(P)$ with $(v(P), j) \in \bar{A}$. We first check whether this is a feasible extension, i.e., whether the resource consumption constraints are respected at vertex j. If that is the case, we obtain a new ng-feasible label and store it in bucket $B(j, t(P) + t_{v(P)j})$. As we have discussed above, the load will be reset to 0, if vertex j denotes a depot.

The set of forbidden vertices should be updated in a similar way as the work of [26], but with some modifications due to the existence of intermediate depot vertices. In our implementation, we associate each customer vertex $j \in V_c$ with an ng-set $NG(j)$ composed of 8 nearest customer neighbors. When extending a label $L(P)$ to a vertex $j \in \overline{V} \setminus \Pi(P)$, its successor's forbidden set is updated to be $(\Pi(P) \cap NG(j)) \cup \{j\}$, if vertex j denotes a customer; otherwise, the successor maintains the same forbidden set as the label itself. The idea behind this is that enforcing forbidden sets and thus ng-routes is to seek for partial elementarity of routes generated in the SPPRC. Since the elementarity of a given route is not affected by the existence of depot vertices, those vertices should be excluded from forbidden sets. When no more label extensions can be made, we collect all columns stored in buckets $B(n + m + 1, t')$ with $0 < t' \leq T$ and return all those with negative reduced costs.

To accelerate the labeling algorithm, it is crucial to check for dominance relationships so as to avoid some unnecessary label extensions. More specifically, before storing a new ng-feasible label, we first check whether it dominates or is dominated by existing labels stored in buckets. We say that a label $L(P_1)$ dominates another label $L(P_2)$ if, for any feasible path

extended from P_2, we can always find a feasible path extension from P_1 and this extended path is not more costly. Sufficient conditions for this are given by (5.18)–(5.20).

$$\bar{c}(P_1) \leq \bar{c}(P_2), \tag{4.10}$$
$$v(P_1) = v(P_2), \tag{4.11}$$
$$d(P_1) \leq d(P_2), \tag{4.12}$$
$$t(P_1) \leq t(P_2), \tag{4.13}$$
$$\Pi(P_1) \subseteq \Pi(P_2). \tag{4.14}$$

A newly generated label $L(P)$ is saved only if it is not dominated, and existing labels that are dominated by $L(P)$ will be removed. In our implementation, the dominance rule is only tested between a label $L(P)$ and labels stored in buckets $B(v(P), t')$ with $\max\{0, t(P) - 10\} \leq t' \leq t(P)$.

We solve a pricing subproblem for each depot $i \in V_d$ and append into $\tilde{R}^{ng}$ the up to 20 columns with the most negative reduced costs. Let $\bar{c}_{min}$ denote the minimum cost among the returned columns. Clearly, $\bar{c}_{min}$ is the minimum reduced cost of all ng-feasible routes.

VARIABLE FIXING: Let z^* denote the current RMP optimal value. It is well-known that one can obtain a valid lower bound, $z^{LB} := z^* + \min\{K, |V_c|\}\bar{c}_{min}$, also known as the *Lagrangian bound* [119]. We emphasize that the fleet size bound K is updated, whenever branching on the fleet size is executed (see later for details on branching rules). Besides pruning branch-and-bound nodes, we also use Lagrangian bounds for variable fixing.

Let us define the primal-dual gap, $z^{gap} := UB - z^{LB}$. One can easily show that a column with a reduced cost larger than $z^{gap} + \bar{c}_{min}$ will not be part of any feasible solution better than the incumbent [147]; thus, such columns can be excluded from consideration. In our context, we employ a technique called arc fixing [98]. Specifically, given arc $(i, j) \in \bar{A}$, if every feasible column traversing this arc has a reduced cost larger than $z^{gap} + \bar{c}_{min}$, then this arc can be omitted from consideration. To reach such a conclusion and omit an arc, we need to apply the labeling algorithm we discussed above in a backward fashion. The backward labeling algorithm works in a similar way as the regular (forward) labeling, but with the following minor modifications: (i) it is initialized by storing the first label $\{0, n + m + 1, Q, T, \emptyset, null\}$ and starts with higher values of t' for label extension; (ii) the inequality conditions (5.21) and (5.22) in the dominance check are reversed; (iii) the dominance relationship is tested between a label $L(P)$ and labels stored in buckets $B(v(P), t')$ with $t(P) \leq t' \leq \min\{t(P) + 10, T\}$.

We separately perform a full run of both forward and backward labeling. The minimum reduced cost of a column passing by an arc $(i, j) \in \bar{A}$, denoted by $\bar{C}_{ij}$, can be obtained by concatenating the labels in $B(i, t')$ from the forward run with the labels in $B(j, t' + t_{ij})$ from the backward run for $0 \leq t' \leq T$. If $\bar{C}_{ij}$ is larger than $z^{gap} + \bar{c}_{min}$, then one can eliminate the arc (i, j) from $\bar{A}$. Since arc fixing is typically effective when z^{gap} becomes small, we begin to fix arcs by reduced costs after the initial convergence of column generation and when z^{gap}/UB is below 5%. Thereafter, arc fixing is performed only if z^{gap} decreases by at least 10% since the last call.

ROUTE ENUMERATION: As [25] suggested, one can enumerate all elementary columns with reduced costs less than $z^{gap} + \bar{c}_{min}$, since only these columns may contribute to a solution better than the incumbent. We adopt the idea from the work of [56] and attempt to enumerate all possible routes when the primal-dual gap becomes small. In particular, after each call of variable fixing, if the primal-dual gap is below 1%, we then attempt to enumerate all columns that have the potential to improve the objective function. In order to enumerate elementary routes using the forward labeling algorithm we discussed above, $NG(i)$ is defined to be the customer set V_c. As a result, $\Pi(P)$ is simply the set of visited customer vertices along path P. Furthermore, as suggested by [131], the dominance rule should be updated in the following way. First, the condition (5.20) is modified to $\Pi(P_1) = \Pi(P_2)$. Second, we define for each label $L(P)$ an extra member $c(P)$ to indicate the original accumulated travel cost along path P, and then, the condition (5.18) is revised to be $c(P_1) \leq c(P_2)$.

The route enumeration procedure can be accelerated by using the so-called *completion bound* [56], which can be computed for free from the variable fixing step. For the sake of brevity, we refer readers to the work of [130, 131] for details.

The enumeration procedure is interrupted when the number of labels exceeds 1.0×10^6, or when the number of enumerated routes exceeds 1.0×10^5. After a successful enumeration, generated columns are not immediately appended to $\tilde{R}^{ng}$ but saved in a pool. At this point, one can also erase non-elementary columns from $\tilde{R}^{ng}$. Finally, once the total number of columns from $\tilde{R}^{ng}$ and the pool is less than 1.0×10^4, we append integrality constraints (5.11) to the RMP and solve the resulting integer program. Otherwise, in the coming iterations, we can simply inspect columns saved in the pool and return at most 150 of them with negative reduced costs.

HEURISTIC PRICING: The column generation procedure can be further accelerated by using a fast heuristic pricing methods called "bucket pruning," which was introduced in the work of [77]. Simply put, this method relaxes the dominance check by only considering the condition (5.18). The trade-off to this efficiency is that non-dominated labels may be dropped, though it is expected that those would be less likely to lead to optimal solutions. In our implementation, we opted to first apply the heuristic labeling algorithm to generate columns with negative reduced costs, and to resort to the exact algorithm only if the heuristic method fails to identify any column. In particular, in each iteration, we return the up to 30 columns with the most negative reduced costs from heuristic pricing.

4.4.2.3 *Cutting Planes*

When column generation converges, the master problem achieves its optimality. If the current LP optimal solution is not integral, we try to identify strengthening inequalities to exclude the fractional solution from the relaxed feasible space. We consider two types of valid inequalities, namely *Rounded Capacity Inequalities* (RCI) [110] and *Subset Row Cuts* (SRC) [99].

The separation routine for RCI is synopsized as follows. Let $G^* = (V, A)$ be a support graph corresponding to the current solution to an RMP. We first shrink the set of depot vertices, V_d, to be a single vertex, which results in a support graph for the classic capacitated vehicle routing problem. We then apply a tabu search algorithm inspired from [21] to identify violated RCI. In the context of a branch-price-and-cut algorithm, RCI are so called "robust" cuts [15] because their corresponding dual values can be properly accommodated into the modified arc cost in the SPPRC; thus, adding RCI into the set partitioning model as strengthening inequalities will not complicate the solution of pricing subproblems.

With regards to SRC, it is well-known that they usually help to close the primal-dual gap significantly; however, these are known to be "non-robust" cuts, since their dual values cannot be easily combined into the arc cost in the SPPRC, changing the pricing subproblem structure. In essence, each additional SRC makes pricing harder. To mitigate the negative effect of SRC on solving SPPRC subproblems, we adopt a variant of SRC, called *limited-arc-memory subset row cuts* (lm-SRC), which was proposed in the work of [133]. The lm-SRC are still non-robust cuts but have much less impact on solving pricing problems [131]. We use the procedure proposed in the work of [132] to separate lm-SRC. In particular, for every $i \in V_c$, we define $N(i) \subseteq V_c$ as the set containing the 15 other customers closest to i and then test the violation of all SRC obtained by applying multipliers to all base sets S with cardinalities 3, 4 and 5, and for each i and j in S, $j \in N(i)$. After violated SRC are

identified, we add their corresponding limited-arc-memory version. To accommodate the use of lm-SRC, the labeling algorithm is modified in the following aspects: (i) we define a state for each active cut; (ii) the contributions from dual values to lm-SRC are taken into consideration when computing the reduced cost for a new label; (iii) the condition (5.18) in the dominance check is updated to account for the dual values to lm-SRC. Readers are referred to [132, 147] for implementation details.

We now discuss our overall cut separation protocol. At the root node, we first attempt to identify violated RCI and resort to lm-SRC only if no RCI is found or if "tailing-off" is detected for RCI. We define tailing-off when the node primal-dual gap z^{gap} has not improved by at least 0.1% during the past 3 iterations. In all other nodes of the branch-and-bound tree, both RCI and lm-SRC are separated in every round. If the tailing-off condition is satisfied, we stop the cut separation and perform branching. Since, as discussed, the lm-SRC have a negative effect on our ability to keep solving SPPRC subproblems, we add into the set partitioning model only the 5 most violated lm-SRCs per round and add at most 300 lm-SRCs in total. We also remove non-active lm-SRC from the model before each cut separation round. After route enumeration at a given node is completed, all lm-SRC are lifted to their SRC counterparts, since we no longer need to solve the SPPRC.

4.4.2.4 *Branching Strategy*

When the optimal solution to the master problem is not integral and no violated cuts can be identified, or if we have decided to stop adding cuts due to the tailing-off issue, we then perform branching. We first attempt to branch on the number of used vehicles, since from our experience it often has a more profound impact on the branch-and-bound node lower bound; if this is not applicable, then branching on edges performed by choosing the edge with value closest to 0.5. Readers are referred to [71] and [146] for details. We emphasize that the branching rules we adopt do not change the pricing structure and, thus, do not add complexity when solving the SPPRC.

4.4.2.5 *Other implementation details*

FEASIBILITY-RECOVERY: When the RMP becomes infeasible (e.g., due to branching or because the initial RMP itself is infeasible), we resort to the feasibility-recovery step as suggested by [71], which corresponds to the Phase I step in the classic simplex algorithm. In particular, we introduce a dummy variable into the RMP with its coefficient at each constraint being the right-hand side and its objective coefficient being $UB + \epsilon$, where $\epsilon > 0$ (we used $\epsilon = 0.1$). Clearly, the modified linear program becomes feasible. The

resulting linear program is then solved and its dual values are used to generate new columns with negative reduced costs so as to recover the RMP feasibility. When column generation converges, i.e., the LP optimality is achieved, and assuming the dummy variable attains a value of zero, the primal feasibility of the RMP is recovered; otherwise, if the dummy variable attains a non-zero value, the LP optimal value must be $UB + \epsilon$ (due to complementary slackness), resulting in the node being pruned by bound in the branch-and-bound tree.

PRIMAL HEURISTICS: Whenever the RMP is solved, we apply a simple rounding heuristic in an attempt to find better feasible solutions than the incumbent. In our implementation, elementary columns with their current corresponding LP optimal values above 0.5 are collected and we check whether the routing plan composed of these columns is feasible. If this leads to a feasible solution with an objective value better than the incumbent, we then update the upper bound UB.

4.5 COMPUTATIONAL STUDIES

Our algorithm was implemented in C++ and all subordinate linear and mixed-integer linear programs were solved using the IBM ILOG CPLEX Optimizer 12.9.0 through the C application programming interface, with all settings being default except that the relative optimality gap tolerance was set to the value of 0. In the branch-price-and-cut algorithm, the absolute optimality gap tolerance was set to $\varepsilon^{\text{opt}} = 1.0 \times 10^{-4}$, and the best-bound node selection strategy was chosen. All computations were performed on an Intel Xeon CPU E5-2689 v4 server running at 3.10 GHz. A total of 128GB of available RAM was shared among 10 copies of the algorithm running in parallel on the server. In Section 4.5.1, each instance was solved by one copy of the algorithm using a single thread. The results presented in Section 4.5.2 were obtained with OpenMP by using up to 10 threads in parallel.

4.5.1 *Evaluation of BPC Implementation Performance*

We first evaluate the performance of our BPC algorithm on solving MDVRPI benchmark instances and compare our algorithm with the state-of-the-art branch-and-price algorithm proposed in the work of [127]. The differences between these two algorithms can be synopsized as follows:

1. our BPC algorithm models each pricing subproblem as an SPPRC and generates routes with negative reduced costs in a single stage; in contrast, the branch-and-price algorithm considers a two-phase approach where trips between each pair of depots are first constructed and then routes with negatived reduced costs are obtained by combining these trips;

2. our BPC approach incorporates certain state-of-the-art pricing techniques, such as *ng*-routes, variable fixing, route enumeration, among others;

3. we add valid inequalities to strengthen the LP relaxations at each node of the branch-and-bound tree.

We consider the instances from the work of [58] and we adapt them in the same way as done in [127]. There are 12 `random` (a1-l1) and 10 `CGL` (a2-j2) instances. In these instances, which are available at http://chairelogistique.hec.ca/data/mdvrpi/, the number of depots ranges from 3 to 7. For each instance, we consider the first 25, 40 and 70 customers with the total number of vehicles limited to 4, 6 and 10, respectively. Note that datasets a1, d1, and a2 from [58] were not adapted to 70-customer instances because they only feature 48 customers.

In each instance, the vehicle capacity Q and the route duration limit T is set to be 50 and 450, respectively, as done in [127]. Also following that work, the travel time is calculated from the coordinates and we do not round these values. Thus, each entry of the travel time matrix is a double-precision non-negative number. As discussed earlier, in order to use our labeling algorithm in such setting, we need to round down all entries of the matrix and to define an extra resource as the "reference" to store labels in an SPPRC.

Finally, as it is common practice in the VRP literature using BPC algorithms (see, e.g., [77, 131, 133, 135, 147, 169]), tight initial upper bounds are provided so that the various pricing techniques like variable fixing and route enumeration are activated in the early stage of the algorithm. Since we did not have at hand sophisticated heuristics for solving the MDVPRI, we consider the following two strategies to mitigate this. The first strategy is to guess an initial upper bound via utilizing the BPC root node lower bound (denoted by $RootLB$) as a reference and to adjust it if necessary. In particular, we choose the initial upper bound to be $RootLB \times \alpha$ with α being 1.01; if the resulting mathematical model is deemed infeasible by our BPC algorithm, we then increase α by 0.005 and iterate. The second strategy is to use as the initial upper bound the optimal (or best known) value of each instance modified upwards by an offset of 1.0×10^{-2}. Note that, since we do not round the travel costs, and since 1.0×10^{-2} is larger than the branch-and-bound absolute

optimality gap tolerance ε^{opt}, our BPC algorithm always has to locate by itself a feasible solution with a value better than the initial upper bound provided. For convenience, we call the former and the latter an *iterative strategy* and a *BKS strategy*, respectively.

Our computational results are presented in Tables 4.1 and 4.2. In the instance name field, "X" is a placeholder for the number of customers. For our three different values of "X", the columns "Opt [UB]" then report the corresponding optimal objective values, while the columns "t (sec) [LB]" provide the time to solve the instance to optimality; if an instance could not be solved within the allotted time limit of 2 hours, these columns report (in brackets) the best upper and lower bounds (rounded to 3 decimal places) found within this time limit. If an instance was solved to optimality by both aforementioned strategies, we then report two solution times that are separated by "/" in the column "t (sec) [LB]"; otherwise, we only report the computational results under the BKS strategy.

Focusing first on the 25- and 40-customer instances, the tables show that our BPC algorithm solved all 44 instances to optimality in both strategies, with the solution time ranging from 4 to 1,733 seconds. The performance of our iterative strategy significantly improves upon the current state-of-the-art exact approach from [127], which could only solve 23 instances to proven optimality within a time limit of 10 hours, noting however that, according to www.cpubenchmark.net/singleThread.html, our computer's computational speed was approximately double that of the machine used in [127]. When utilizing best known solutions as prior information for determining initial upper bounds, the BKS strategy solved these benchmark instances on average 2.7 times faster than the iterative strategy. Focusing on the larger, 70-customer instances, the iterative strategy solved 5 out of a total of 19 to optimality with an average solution time of 4,149 seconds, while the BKS strategy could solve to optimality 17 instances with an average solution time of 2,201 seconds, leaving an average residual gap of 0.90% to the remaining 2 instances at the time limit of 2 hours. The disparity strengthens the belief that BPC algorithms' performance is sensitive to the initial upper bound [135].

Overall, our computational results demonstrate that, with tight upper bounds provided, our BPC algorithm is able to routinely solve to provable optimality MDVRPI instances with up to 40 customers within a mere couple of minutes, while it can solve most MDVRPI instances involving up to 70 customers within 2 hours. We report our optimal and best known solutions to all these benchmark instances in the following link: http://gounaris.cheme.cmu.edu/bks/mdvrpi/.

Table 4.1: Computational results for our BPC algorithm on 25-, 40-, and 70-customer `random` instances

Instances	X = 25		X = 40		X = 70	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
a1-X-50-450	693.810	9 / 4	998.431	35 / 22	–	–
b1-X-50-450	731.414*	29 / 5	1,059.370	61 / 27	1,559.669*	4,968
c1-X-50-450	855.831*	71 / 23	1,148.669*	87 / 23	1,692.074*	1,951
d1-X-50-450	763.303*	41 / 12	1,056.868*	283 / 74	–	–
e1-X-50-450	803.715	13 / 5	1,236.620	111 / 32	1,806.888*	4,888 / 937
f1-X-50-450	551.828	24 / 13	854.108*	563 / 85	1,365.147*	4,760
g1-X-50-450	654.016	74 / 15	1,034.936*	1,130 / 191	1,405.974*	4,979
h1-X-50-450	558.646	74 / 21	875.552	179 / 76	[1,380.122]	[1,364.404]
i1-X-50-450	823.775*	17 / 14	1,222.845	389 / 54	1,804.683*	609
j1-X-50-450	777.701*	289 / 79	904.281*	244 / 109	1,330.820*	3,177
k1-X-50-450	868.604*	14 / 10	1,255.505*	912 /164	1,803.937*	2,042
l1-X-50-450	818.280	12 / 9	1,085.320	295 / 62	1,643.854*	1,129

*Instances solved to optimality for the first time are indicated with an asterisk.

Table 4.2: Computational results for our BPC algorithm on 25-, 40-, and 70-customer `CGL` instances

Instances	X = 25		X = 40		X = 70	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
a2-X-50-450	725.157*	55 / 13	1,010.607	287 / 87	–	–
b2-X-50-450	912.429	72 / 21	1,238.944	295 / 42	1,636.112*	403
c2-X-50-450	683.188	65 / 15	1,159.053*	308 / 38	1,859.551*	3,981
d2-X-50-450	876.113	60 / 17	1,199.848*	186 / 36	[1,864.038]	[1,851.572]
e2-X-50-450	699.363*	15 / 5	1,087.712*	248 / 54	1,801.178*	3,014 / 1,959
f2-X-50-450	781.176	13 / 8	1,046.454*	138 / 43	1,617.750*	983
g2-X-50-450	793.633*	30 / 19	1,035.008	208 / 44	1,521.121*	2,399 / 702
h2-X-50-450	716.220	58 / 15	940.924	406 / 114	1,528.441*	5,451 / 793
i2-X-50-450	910.505	45 / 10	1,171.579*	225 / 74	1,696.617*	2,381
j2-X-50-450	609.378	26 / 20	887.270*	1,733 / 279	1,488.898*	4,990 / 1,656

*Instances solved to optimality for the first time are indicated with an asterisk.

4.5.2 *Marginal Cost Analysis*

The previous section demonstrated the effectiveness of our BPC implementation in terms of its ability to solve MDVRPI instances. Being equipped with this state-of-the-art code, we now turn our focus to utilizing the latter as the VRP solver in our framework to estimate the marginal cost to serve individual customers. Since there is no available benchmark instance in the literature for this setting, we generated those as follows. We consider the first five `random` instances we used in Section 4.5.1, namely instances a1-25-50-450 through e1-25-50-450, each having 25 customers, and in every instance we inserted a 26^{th} customer as our target customer to perform the marginal cost analysis. The target customer's coordinates, fixed service time and nominal demand are chosen to be the rounded average (rounded to the nearest integer) of the 25 original customers, respectively.

The stochastic customer demands are designed in the following way. Firstly, since not all 25 original customers generally request a visit on the same day, we assign each of them a probability to indicate how likely the customer is to request an order on any given day. In particular, we assign 75%, 50% and 25% to the first 6, intermediate 12 and last 7 customers, respectively. The target customer always requests an order on any given day, and is thus given a probability of 100%. Secondly, for any customer i who requests an order, we consider the demand from the instance data as its nominal value, denoted by $\bar{q}_i$, and assume that the stochastic demand q_i is an integer value that always falls into the set $\mathcal{J}_i := \left[\left\lceil 0.7\bar{q}_i \right\rceil, \left\lfloor 1.3\bar{q}_i \right\rfloor\right] \bigcap \mathbb{N}$. Let p_{ij} denote the probability that q_i takes a specific value j from this set. We set p_{ij} to be proportional to the probability density function of a normal distribution with mean $\bar{q}_i$ and standard deviation $0.1\bar{q}_i$, evaluated at q_i, and normalized such that $\sum_{j\in\mathcal{J}_i} p_{ij} = 1$. We highlight that, although the demand distribution $\Xi \times \Theta$ was designed in the way described above so as to allow us to easily implement the sampling process in our computational studies, our proposed scenario-sampling framework is generally valid for any given demand distribution. Furthermore, in order to ensure that every possibly sampled routing scenario remained feasible, the fleet size was set to be unlimited.

We now discuss how to conduct scenario-sampling from the given demand distribution and how to perform the marginal cost calculation. When generating a scenario, we first draw from a Bernoulli distribution with the order-requesting frequency as the parameter, to decide whether a given customer requests an order or not. If it does, we then determine the customer's demand in this scenario by sampling from its distribution. After this process is repeated for each customer, a routing scenario with customers having fixed demands has

been defined. In order to computing the marginal cost of serving the target customer in this scenario, we utilize our BPC algorithm to solve to optimality two MDVRPI instances, one with and another one without the target customer, and then apply the formula (4.1). The average marginal cost across all sampled scenarios is returned as the estimate. We emphasize that, since each scenario can can be solved independently, we utilize OpenMP to parallelize our code using up to 10 threads in the following experiments and report the CPU wall time.

4.5.2.1 *Marginal Cost Estimation*

In our context, $\underline{\texttt{MC}} = 0$ is a valid lower bound to the marginal cost, since serving an additional customer never reduces the routing cost. Let C denote the cost for the direct round-trip route from the depot that is closest to the target customer. The value $\overline{\texttt{MC}} = C / \left\{ \min_{\vartheta \in \Theta} \vartheta \right\}$ is a safe upper bound. The probability threshold is chosen to be $\alpha = 5\%$, while the permissible deviation is chosen as $\delta = 2\% \times (\overline{\texttt{MC}} - \underline{\texttt{MC}})$. Consequently, from equation (4.5), the minimum number of samples that is required for the specified accuracy is $\overline{N} = 4,612$. Fig. 4.3 shows the progress of the marginal cost average, as the sample size increases, for the benchmark instance d1-25-50-450, which is selected here as a representative of all instances.

There are several noteworthy observations. Firstly, the marginal cost average initially fluctuates strongly, but then gradually stabilizes around the value of 0.501, which empirically confirms the conclusion made in Proposition 4.1, i.e., that the sample average $\frac{1}{N} \sum_{i=1}^{N} \texttt{MC}\left(\xi^i, \vartheta^i\right)$ converges to a constant value (specifically, the value $\mathbb{E}_{\xi,\vartheta}\left[\texttt{MC}\left(\xi, \vartheta\right)\right]$) when the sample size N is large enough. The claim that the estimate is of high quality is further strengthened by the fact that the chosen value of the permissible deviation ($\delta = 0.020$ in this example) ended up being nearly negligible when compared to this estimate. Secondly, the stabilized estimate was produced relatively early in the execution of the algorithm, but many additional scenarios had to be sampled until the probability of a sufficiently accurate estimate was lowered below the required threshold to warrant stopping the process (pre-computed sample size $\overline{N}$). We remark that, from a practical point of view, one can keep track of the progress of the marginal cost average and terminate the algorithm when the sample average satisfactorily stabilizes, which might occur significantly before $\overline{N}$ samples have been collected and processed.

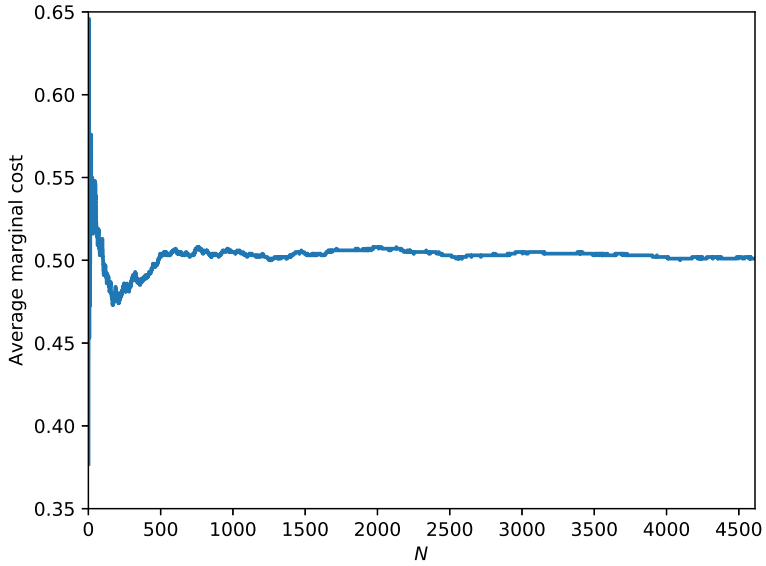


Figure 4.3: Example progress of the average marginal cost, as the sample size N increases

4.5.2.2 *Customer Locations*

We now investigate the effect of a customer's location on the marginal cost of serving this customer. Generally speaking, when a customer is located at a place far away from the depot facilities, or far away from the other customers, a vehicle has to drive a longer distance for deliveries to this customer, which causes more routing costs. In order to quantify this effect in the context of a stochastic supply chain network, we consider two additional locations for the target customer in each of the five proposed instances: (i) the first location, dubbed "Central," having coordinates at the rounded (to the nearest integer) average of the depots' coordinates, and (ii) the second location, dubbed "Remote," having coordinates $(\max_i x_i, \max_i y_i)$, where (x_i, y_i) represent the i-th customer's coordinates in this instance. Along with the original placement of the target customer, a total of three customer locations are considered. We shall again choose $\alpha = 5\%$ and $\delta = 2\% \times (\overline{\texttt{MC}} - \underline{\texttt{MC}})$ for our estimation parameters, while we shall again generate 4,612 representative scenarios and compute the average marginal cost.

Table 4.3 shows the computational results for our five instances under the three customer location cases. The column "δ" denotes the applicable value of the permissible deviation parameter, as determined for the each specific instance, while the column "MC" reports the marginal cost estimate. The relative estimation error is reported in the column "$\delta/\texttt{MC}$ (%)".

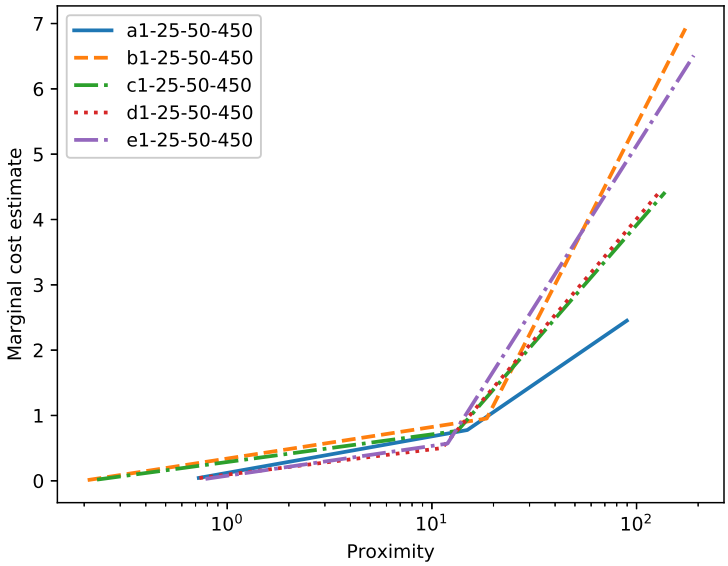


Figure 4.4: Effect of a customer's proximity to the depot facilities on the marginal routing cost. The proximity is defined as the length of the direct round-trip route from the depot that is closest to the target customer.

Finally, we also report the total CPU wall time for running each instance in the column "t (sec)".

The entries in this table reveal that, when the target customer is located further away from the depot facilities, the marginal cost of serving this customer increases, as also plotted in Fig. 4.4 for all five benchmark datasets. Notably, we can claim in each case that the probability that the expected marginal cost deviates from the returned estimate by *at most* the corresponding relative estimation error is above $1 - \alpha = 95\%$. In particular, this demonstrates a satisfactory estimation accuracy, since the average (among all instances) deviation turns out to be merely 4.5% of the respective estimates. Finally, we remark that the computational time in these experiments ranges from approximately $1,200$ to $2,500$ seconds, which is quite short considering that a total of $9,224$ MDVRPI instances have to be solved to proven optimality in each case.

4.5.2.3 *Demand Levels*

In this section, we consider the effect of a customer's demand level on the marginal routing cost. Let $\bar{q}$ denote the nominal demand of the target customer in each of the five proposed instances, and let us now also consider two additional demand levels, namely $\bar{q} \pm 5$. The

Table 4.3: Marginal cost estimation under three customer location cases

Instances	Central				Original				Remote			
	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)
a1-25-50-450	0.001	0.044	2.3	1,566	0.030	0.777	3.9	1,883	0.179	2.451	7.3	1,707
b1-25-50-450	4×10^{-4}	0.011	3.8	1,431	0.037	0.953	3.9	1,221	0.347	6.924	5.0	1,425
c1-25-50-450	5×10^{-4}	0.018	2.6	1,811	0.027	0.764	3.5	1,417	0.276	4.418	6.2	1,463
d1-25-50-450	0.001	0.042	2.4	2,099	0.020	0.501	4.0	2,509	0.237	4.435	5.3	2,277
e1-25-50-450	0.002	0.029	6.9	1,266	0.026	0.569	4.6	1,668	0.423	6.510	6.5	1,324

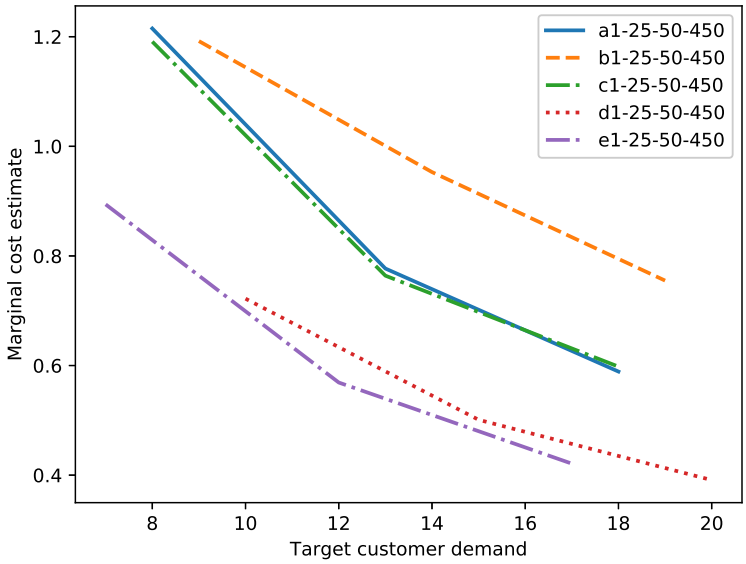


Figure 4.5: Effect of a customer's demand level on the marginal routing cost

three demands are classified, based on their values, as "Low," "Medium," and "High." The customer demand distribution is created in the same way as before. Again, $\alpha = 5\%$ and $\delta = 2\% \times (\overline{\texttt{MC}} - \underline{\texttt{MC}})$. Table 4.4 shows the relevant computational results. As expected, when the customer demand increases, the marginal routing cost decreases, as also plotted in Fig. 4.5. The actual deviation is on average within 4.1% of the marginal cost estimate, which is considered acceptable from a practical point of view. Again, the computational times are deemed quite affordable.

4.6 CONCLUSIONS

We considered the problem of estimating the marginal routing cost of serving an extra customer on top of a given distribution network. In this context, the main challenge stems from the intrinsic stochasticity in customer demands, and to counteract this, we proposed a systematic scenario-sampling framework that can rigorously quantify the marginal routing cost. In particular, we used probability theory to demonstrate that the estimate returned by our framework will converge to the true value with sufficient increase of the sample size. For purposes of practicality, we proposed to utilize the well-known Hoeffding's inequality to bound the sample size for a desired accuracy.

Table 4.4: Marginal cost estimation under three customer demand levels

Instances	Low				Medium				High			
	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)	δ	MC	δ/MC (%)	t (sec)
a1-25-50-450	0.050	1.215	4.1	1,623	0.030	0.777	3.9	1,883	0.023	0.589	3.9	1,354
b1-25-50-450	0.053	1.192	4.4	1,224	0.037	0.953	3.9	1,221	0.026	0.755	3.4	1,482
c1-25-50-450	0.044	1.191	3.7	1,810	0.027	0.764	3.5	1,417	0.021	0.598	3.5	1,209
d1-25-50-450	0.032	0.722	4.4	2,437	0.020	0.501	4.0	2,509	0.016	0.391	4.1	1,948
e1-25-50-450	0.048	0.894	5.4	1,016	0.026	0.569	4.6	1,668	0.020	0.421	4.8	1,298

In our computational studies, considering each scenario entailed solving two $\mathcal{NP}$-hard multi-depot vehicle routing problems with inter-depot routes. To this end, we develop a tailored branch-price-and-cut algorithm that incorporates several state-of-the-art techniques, including *ng*-routes, route enumeration, and limited-memory subset row cuts, among others. Our implementation was shown to perform significantly better than the previous state-of-the-art exact approach on solving MDVRPI benchmark instances. Specifically, within a time limit of 2 hours, our algorithm was able to close all 21 MDVRPI benchmark instances involving up to 40 customers that remained opened in the literature, while it was additionally able to solve 17 out of a total of 19 newly-defined benchmark instances with 70 customers.

Utilizing our branch-price-and-cut code, we conducted extensive computational studies to demonstrate the deployment of our proposed framework for estimating the incremental routing cost, elucidating the accuracy of our estimates as well as quantifying the effect of two important factors, namely customer location and demand, on the marginal cost estimate.

5

A BRANCH-PRICE-AND-CUT ALGORITHM FOR ROBUST VEHICLE ROUTING PROBLEMS UNDER UNCERTAINTY

In this chapter, we focus on solving vehicle routing problems under uncertainty from a robust optimization perspective. Given postulated uncertainty sets for customer demands and vehicle travel times, one aims to identify a set of cost-effective routes for vehicles to traverse such that along these routes vehicle capacity constraints and customer time window constraints are respected under any anticipated demand and travel time realization, respectively. To tackle this problem, we propose a novel approach that combines cutting-plane techniques with the advanced deterministic branch-price-and-cut algorithm. In particular, we apply a deterministic pricing procedure to dynamically generate vehicle routes and then utilize robust rounded capacity inequalities and infeasible path elimination constraints to guarantee robust feasibility of routing designs against demand and travel time uncertainty. We conduct extensive computational studies on benchmark instances and compare our approach against the existing research efforts of adapting branch-price-and-cut algorithms for addressing robust vehicle routing problems under uncertainty. Our computational studies demonstrate that our proposed approach is not only comparable to against the state-of-the-art approaches but also able to efficiently handle a variety of commonly used uncertainty sets.

5.1 INTRODUCTION

The *vehicle routing problem* (VRP) is one of the most highly studied combinatorial optimization problems in the area of supply chain and logistics. The classic setting is the one where, given a distribution center, a fleet of vehicles and a group of customers to be served, one aims to identify a set of minimum-cost routes for vehicles to traverse, such that customer demands are satisfied and relevant system constraints, e.g., vehicle capacities, time windows, and route duration limits, among others, are respected. For the

past a few decades, lots of research efforts have been dedicated to the development of exact and heuristic approaches for solving VRPs [167]. The majority of these studies have been conducted under the assumption that all the information necessary to formulate the relevant routing problems is known and readily available. In practical applications, this assumption is usually not verified due to the presence of uncertainty affecting the parameters of the problem. For example, vehicle travel times can vary due to unexpected events such as bad weather, vehicle breakdown or traffic congestion. Making routing decisions while ignoring parameter variability can potentially cause either cost-prohibitive routing designs or failure to satisfy service commitments. Hence, it is of great importance for the distributor to take uncertainty into account at the route planning stage.

The work of [78] identified three types of uncertainty sources that are commonly found in VRP applications: customer demands, travel/service times, and customer orders. The customer demand uncertainty indicates a situation where the amount of product either delivered or picked up at customers are random. The travel/service times are subject to change due to unpredicted events, e.g., weather conditions, and hence they are often not known with precision at the moment the distributor is designing routes. Customer order uncertainty often arises when the distributor may receive unanticipated orders from customers and at the same time he has to take these orders into consideration before routes are to be committed [161]. In this chapter, we focus on customer demand and vehicle travel time uncertainty, since they are the most popular ones in practical applications.

Most contributions for tackling VRPs with uncertainty in demands and travel times come from the *stochastic programming* [40] paradigm. It assumes that customer demands/vehicle travel times are random variables that follow a given probability distribution. The work of [78] surveyed existing research efforts using this paradigm and classified them into three main modeling frameworks. The most common one is to model the routing problem under uncertainty as a two-stage optimization problem: routing decisions are made *a priori* in the first stage, and then in the second stage uncertainty is gradually revealed and recourse actions are taken so as to fulfill service commitments that otherwise will not be delivered upon due to the presence of uncertainty, hence it is usually called *a priori optimization* approach. The second framework is a *re-optimization* approach. Its key feature is that routing decisions are made dynamically in order to benefit from the fact that the information related to uncertainty parameters is revealed over time. The third one is a *chance-constrained* approach which ensures that the feasibility of a routing design is achieved above a prespecified probability threshold. We refer interested readers to [78] for a comprehensive discussion on this topic.

The stochastic programming paradigm is a natural mindset when dealing with uncertainty, but it suffers from two main shortcomings: (i) information availability, that is, the probability distribution of uncertain parameters is usually either not available or not known precisely at the decision-making moment; (ii) tractability, that is, when uncertain parameters are random continuous (discrete) variables, the resulting mathematical model usually entails both optimization and numerical integration (many scenarios), and is thus prohibitive to solve. In contrast, *robust optimization* (RO) [33] can be a promising approach for the treatment of uncertainty in VRP applications. This approach only assumes that uncertain parameters are random variables falling into a postulated set and it seeks for the optimal routing design that is immunized against all anticipated parameter realizations from that set. Usually, this uncertainty set can be constructed with probabilistic confidence using historical data. Additionally, computational tractability is often preserved when compared with its deterministic counterpart [33]. Consequently, the aforementioned two issues are now circumvented. In this chapter, we opt for the RO paradigm to *a priori* design cost-effective routes that are immunized against infeasibility caused by variability in customer demands and vehicle travel times. For convenience, we refer to the routing problems of our interest as *robust vehicle routing problems*.

In recent years, there has been a steadily growing interest in both formulating and solving robust VRPs. The work of [163] considered a RO approach for the *capacitated vehicle routing problem* (CVRP) with uncertainty in demand and anticipated that each customer's demand can attain its maximum value simultaneously, leading to a overly conservative formulation. A distinguished work is [82], in which the authors considered the robust CVRP with customer demands being supported on a generic polyhedron and they derived and compared *robust counterparts* of several deterministic CVRP formulations. The computational studies show that embedding their proposed *robust rounded capacity inequalities* into the classic *vehicle flow formulation* [110] is the most effective method to address robust VRPs under demand uncertainty. To deal with travel time uncertainty, the work of [5] applied the classic *dualization* technique for RO, yielding a large formulation that is hard to solve for instances of more than 20 customers. Prominent contributions have been made by [6], in which the authors proposed two novel techniques to handle travel time variability: *column-and-constraint generation* [176] and *cutting-plane*. The former method initially only considers a subset of anticipated scenarios and then gradually selects and introduces some neglected extreme scenarios via appending both variables and constraints. This process iterates until the returned routing design is robust feasible with respect to all anticipated scenarios. The cutting-plane method ensures the immunity of a routing

solution by dynamically separating and enforcing *infeasible path elimination constraints*. The computational studies in [6] show that these two techniques have comparable performance in terms of solving robust VRPs with time windows. The column-and-constraint generation approach was also employed in the work of [7] for the treatment of travel time uncertainty in a maritime inventory routing application.

All the aforementioned works[5–7, 82, 163] are utilizing the popular *arc-based formulation*, which includes polynomially many binary variables and is thus solved by the *branch-and-cut* method. In the realm of deterministic problems, the *route-based formulation* is becoming more and more popular due to its tight linear programming (LP) relaxation. This formulation has exponentially many binary variables, each of them representing the selection of a feasible vehicle route. The resulting LP relaxation at each branch-and-bound node is thus solved by *column generation* [119], which simply starts with a subset of feasible vehicle routes and entails the solution of *pricing subproblems* to dynamically introduce neglected feasible routes with the potential of reducing the objective value. The pricing subproblem is commonly modeled as a *shortest path problem with resource constraints* (SPPRC) and can be solved efficiently via *dynamic programming*. Strengthening constraints are usually incorporated to tighten the LP relaxation, hence the solution approach for route-based formulations is referred to as a *branch-price-and-cut* (BPC) method. For the past decade, considerable progress has been made in the development of BPC algorithms [57]. To date, the BPC algorithm has emerged as the state-of-the-art exact approach for addressing deterministic VRPs [167]. In this chapter, we aim to explore the possibilities of extending the BPC framework to solve robust VRPs.

Whereas lots of literature has been devoted on how to address deterministic VRPs via BPC algorithms, there have been only a few works combining RO with BPC to tackle VRPs under uncertainty. To the best of our knowledge, the first attempt was made in [113]. The authors focused on the robust VRP with deadlines under demand and travel time uncertainty and proposed a novel approach that is able to directly encapsulate demand/travel time variability into pricing subproblems. This approach was recently revisited and improved by [126] for solving the robust VRP with time windows under demand/travel time uncertainty. In both works, the pricing subproblem is formulated as a *robust shortest path problem with resource constraints* (RSPPRC) [11], for generating routes that are robust feasible with respect to demand/travel time variability. The authors of the above works demonstrated that the resulting RSPPRC could be efficiently solved when uncertain parameters are supported on the commonly used *cardinality-constrained uncertainty set* [37]. To handle robust VRPs with demand uncertainty, another approach was proposed in [136]

and [118], where the authors transformed a RSPPRC into polynomially many deterministic SPPRCs, avoiding the burden of solving the RSPPRC directly. The former work showed that the transformation could be achieved for two popular polyhedral uncertainty sets from the RO literature.

Though the above two approaches from the literature seem quite promising, they exploit the structures of uncertainty sets and hence can not be generalized to many commonly used uncertainty sets (e.g., ellipsoidal sets) from the RO literature. Therefore, it is of great necessity to develop a new approach that can work for solving robust VRPs under various, general types of uncertainty sets. [82] and [6] proposed the cutting-plane idea to enforce robust feasibility and embedded it into a branch-and-cut framework for addressing robust VRPs with uncertainty in demands and travel times, respectively. The authors have demonstrated their success in handling several classes of uncertainty sets. The work of [65] integrated the cutting-plane idea into a BPC framework for solving the *chanced-constrained vehicle routing problem* with stochastic demands. This motivates us to develop a generic approach of combining cutting-plane techniques with the deterministic BPC algorithm, so as to address robust VRPs under a variety of uncertainty sets.

The distinct contributions of our work can be summarized as follows.

- We consider robust VRPs with uncertainty in customer demands and vehicle travel times. We consider five popular classes of uncertainty sets: cardinality-constrained sets, budget sets, factor models, ellipsoids, and discrete sets. We perform polyhedral studies on the cardinality-constrained set and the factor model, and reduce each set to its equivalent discrete set.

- We propose a novel BPC algorithm to address robust VRPs under demand and travel time uncertainty. Our algorithm embeds cutting-plane techniques into the deterministic BPC framework. In particular, we utilize a deterministic pricing engine to generate partially robust feasible routes and then dynamically enforce robust rounded capacity inequalities and infeasible path elimination constraints as necessary constraints to ensure the immunity of a routing design against infeasibility caused by variability in demands and travel times, respectively. To that end, we demonstrate that separating these inequalities can be done efficiently for the aforementioned uncertainty sets.

- We synopsize the existing methods from the literature that have demonstrated success in extending BPC algorithms to the solution of robust VRPs. We then make a detailed comparison between the literature approach and our cutting-plane approach

in the following aspects: uncertainty sources, uncertainty sets, the time complexity of pricing subproblems and the tightness of LP relaxations, so as to demonstrate the applicability and limitations of each approach.

- We conduct comprehensive computational studies to evaluate our proposed algorithm on solving robust VRPs under the aforementioned uncertainty sets. Through computational experiments, we show that the robust cutting-plane algorithm is versatile for the uncertainty sets of our interest. We compare its performance against those from the literature and demonstrate that our proposed algorithm's effectiveness and efficiency over the state-of-the-art approaches.

The remainder of the chapter is organized as follows. In Section 7.2, we give a formal problem definition. The uncertainty sets of our interest are presented in Section 5.3. In Section 5.4, we study several polyhedral uncertainty sets from a geometric perspective and then reduce them to their equivalent discrete sets. A brief overview of the BPC algorithm for solving deterministic VRPs is given in Section 5.5. In Section 5.6, we summarize the existing approaches from the literature and propose a new one for the adaption of BPC algorithms to address robust VRPs. Section 7.5 presents computational results on the BPC algorithm's performance. Finally, we conclude our work in Section 7.6.

5.2 PROBLEM DEFINITION

The *robust vehicle routing problem with time windows* (RVRPTW) is defined on a directed graph $G = (V, A)$, where $V := V_c \cup \{0, n+1\}$ denotes the set of nodes that is itself composed of a set of customers $V_c := \{1, 2, ..., n\}$, the origin depot 0 and the destination depot $n+1$, and $A := \{(i,j) \in V \times V : i \neq j, i \neq n+1, j \neq 0\} \setminus \{(0, n+1)\}$ is the set of arcs. We consider a fleet of K identical vehicles of capacity $Q \in \mathbb{R}_{>0}$, initially located at the origin depot. Every used vehicle can only leave the origin depot after time 0 and has to return to the destination depot by time H. For convenience, we associate a time window $[e_i, l_i]$ where $e_i = 0$ and $l_i = H$ with node $i \in \{0, n+1\}$. Customer $i \in V_c$ has a demand $q_i \in \mathbb{R}_{>0}$ that has to be delivered during a given time window indicated by $[e_i, l_i]$. A vehicle is allowed to wait if it arrives at customer i before e_i, while arriving after l_i is prohibited. Service at customer $i \in V_c$ must start during the corresponding time window and takes time of $s_i \in \mathbb{R}_{\geq 0}$. Let $c_{ij} \in \mathbb{R}_{\geq 0}$ and $t_{ij} \in \mathbb{R}_{\geq 0}$ represent the cost and time for traversing arc $(i,j) \in A$ by a vehicle, respectively. In this work, we consider the demand vector q and/or the travel time vector t are uncertain and can independently take any value from postulated uncertainty sets $\mathcal{Q}$ and $\mathcal{T}$, respectively. A *feasible vehicle route*

$r = (0, v_1, v_2, ..., v_p, n+1)$ starts from the origin depot and ends at the destination depot such that:

(C1) each customer is visited at most once (i.e., an *elementary* route);

(C2) the vehicle capacity constraint is respected under any customer demand realization $q \in \mathcal{Q}$, that is, $\sum_{i=1}^{p} q_{v_i} \leq Q$ for all $q \in \mathcal{Q}$;

(C3) the time window constraints are satisfied under any travel time realization $t \in \mathcal{T}$, that is, $a_i(t) \leq l_i$ for all $i \in \{v_1, v_2, ..., v_p, n+1\}$ and for all $t \in \mathcal{T}$, where $a_i(t)$ denotes the earliest service start time at node i under scenario t.

We call **(C2)** and **(C3)** *robust capacity feasibility* and *robust time window feasibility* conditions, respectively. The objective is to determine the cost-effective *feasible routes* for vehicles to traverse such that each customer is visited exactly once and no more than K vehicles are used.

The considered problem is reduced to (i) a *vehicle routing problem with time windows* (VRPTW) if $|\mathcal{Q}| = 1$ and $|\mathcal{T}| = 1$; (ii) a *robust capacitated vehicle routing problem* (RCVRP) if time window constraints are neglected.

5.3 UNCERTAINTY SETS

In this section, we review uncertainty sets that are commonly used in the RO literature to model demand and travel time uncertainty in VRP applications. In particular, we consider five demand uncertainty sets and two travel time time uncertainty sets. These sets can be categorized into five groups: cardinality-constraint sets, budget sets, factor models, ellipsoidal sets and discrete sets.

5.3.1 *Cardinality-constrained sets*

The first and foremost set for modeling demand uncertainty is the *cardinality-constrained set*, as shown in equation (5.1).

$$\mathcal{Q}_G := \left\{ q \in \mathbb{R}^n : q_i = q_i^0 + \hat{q}_i \xi_i, \xi_i \in [0,1], \forall i \in V_c, \sum_{i \in V_c} \xi_i \leq \Gamma^q \right\} \tag{5.1}$$

Here, $q^0 \in \mathbb{R}^n_{>0}, \hat{q} \in \mathbb{R}^n_{\geq 0}$ and $\Gamma^q \in \mathbb{N}$ are parameters that need to be specified by the modeler. This uncertainty set stipulates that customer demand q_i can deviate upward from

its nominal value q_i^0 by up to $\hat{q}_i$ and that the number of positive deviations is bounded from above by Γ^q. This uncertainty set was originally proposed in [37] and is also popularly referred to as a "budgeted" or "gamma" uncertainty set. This set has been widely used to model customer demand uncertainty in many VRP applications [62, 96, 113, 126, 136, 159].

The cardinality-constrained set is also often used to model travel time uncertainty, as shown in equation (5.2).

$$\mathcal{T}_G := \left\{ t \in \mathbb{R}^{|A|} : t_{ij} = t_{ij}^0 + \hat{t}_{ij}\xi_{ij}, \xi_{ij} \in [0,1], \forall (i,j) \in A, \sum_{(i,j)\in A} \xi_{ij} \leq \Gamma^t \right\} \tag{5.2}$$

Here, $t^0 \in \mathbb{R}_{\geq 0}^{|A|}, \hat{t} \in \mathbb{R}_{\geq 0}^{|A|}$ and $\Gamma^t \in \mathbb{N}$ are parameters that need to be specified by the user. This set stipulates that the time t_{ij} for traversing arc $(i,j) \in A$ can deviate upward from the nominal value t_{ij}^0 by up to $\hat{t}_{ij}$, and the number of deviations is bounded from above by Γ^t. This set has been widely used to model travel time uncertainty in VRPs [5–7, 42, 96, 113, 126].

5.3.2 *Budget sets*

Consider the demand uncertainty set of the following form (5.3).

$$\mathcal{Q}_B := \left\{ q \in \left[\underline{q}, \overline{q}\right] : \sum_{i\in B_l} q_i \leq b_l, \forall l \in \{1,2,...,L\} \right\} \tag{5.3}$$

Here, $\underline{q} \in \mathbb{R}_{>0}^n, \overline{q} \in \mathbb{R}_{>0}^n$, $B_l \subseteq V_c$ and $b_l \in \mathbb{R}_{>0}$ for $l \in \{1,2,...,L\}$ are parameters given by the modeler such that (i) the customer subsets B_l are pairwise disjoint, i.e., $B_l \cap B_{l'} = \emptyset$ for all $l \neq l'$; (ii) $b_l \leq \sum_{i\in B_l} \bar{q}_i$ for $l \in \{1,2,...,L\}$; (iii) $\mathcal{Q}_B \neq \emptyset$. This uncertainty set (5.3) stipulates that customer demand q_i can deviate within an interval $[\underline{q}_i, \overline{q}_i]$ and that the total demand in every customer set B_l is bounded from above by b_l. This uncertainty set was originally proposed in the work of [82] to model demand uncertainty and then it was used in [83, 136, 159].

5.3.3 *Factor models*

Consider the demand uncertainty set of the following form (5.4).

$$\mathcal{Q}_F := \left\{ q \in \mathbb{R}^n : q = q^0 + \Psi\xi, \xi \in [-1,1]^F, \left| \sum_{f=1}^{F} \xi_f \right| \leq \beta F \right\} \tag{5.4}$$

Here, $q^0 \in \mathbb{R}^n_{\geq 0}, F \in \mathbb{N}, \Psi \in \mathbb{R}^{n \times F}_{\geq 0}$ and $\beta \in [0,1]$ are parameters that need to be specified by the modeler. This uncertainty set (5.4) stipulates that the customer demand vector q is distributed around a nominal demand vector q^0, subject to an additive disturbance of $\Psi\xi$. This disturbance is a linear combination of independent factors $\xi_1, \xi_2, ..., \xi_F$ that reside in the F-dimensional hypercube. This set was originally proposed in the work of [82] to model demand uncertainty and then it was used in [83, 159].

5.3.4 *Ellipsoidal sets*

Consider the demand uncertainty set of the following form (5.5).

$$\mathcal{Q}_E := \left\{ q \in \mathbb{R}^n : q = q^0 + \Sigma^{1/2}\xi, \xi^T\xi \leq 1 \right\}, \tag{5.5}$$

where $\Sigma^{1/2}$ denotes the square root of matrix Σ. Here, $q^0 \in \mathbb{R}^n_{\geq 0}$ and $\Sigma \in \mathbb{S}^n_{\geq 0}$ are parameters that need to be specified by the modeler, and $\mathbb{S}^n_{\geq 0}$ denotes the positive semi-definite cone. This uncertainty set (5.5) stipulates that the customer demand vector q can only attain values in an ellipsoid centered at the nominal demand vector q^0. This uncertainty set was originally proposed by [159] to model customer demand uncertainty in the context of routing a heterogeneous fleet. We emphasize that $\mathcal{Q}_E$ is not a polyhedral set but represents a feasible region dictated by nonlinear constraints. To see this, one can reformulate (5.5) when Σ is not singular and obtain

$$\mathcal{Q}_E := \left\{ q \in \mathbb{R}^n : \left(q - q^0\right)^\top \Sigma^{-1} \left(q - q^0\right) \leq 1 \right\}.$$

5.3.5 *Discrete sets*

Consider the demand uncertainty set of the following form (5.6).

$$\mathcal{Q}_D := \text{conv}\left(\left\{ q^d : d = 1, 2, ..., D^q \right\} \right), \tag{5.6}$$

where conv(·) denotes the convex hull of a finite set of points. Here, $q^1, q^2, ..., q^{D^q} \in \mathbb{R}^n_{\geq 0}$ are $D^q \in \mathbb{N}$ distinct realizations of the uncertain customer demands that need to be specified by the modeler. This uncertainty set (5.6) stipulates that the uncertain customer demand vector can take any value from the convex hull of D^q *a priori* specified demand vectors. This set was originally proposed in the work of [159] to model uncertain customer demands in the context of routing a heterogeneous fleet.

We can define a discrete support for the travel time vector t, as shown by (5.7).

$$\mathcal{T}_D := \text{conv}\left(\left\{t^d : d = 1, 2, ..., D^t\right\}\right). \tag{5.7}$$

Here, $t^1, t^2, ..., t^{D^t} \in \mathbb{R}^{|A|}_{\geq 0}$ are $D^t \in \mathbb{N}$ distinct realizations of the uncertain travel times that need to be specified by the modeler. The uncertainty set (5.7) was originally proposed in [155].

5.4 POLYHEDRAL STUDIES

In this section, we study the extreme points of polyhedral sets $\mathcal{Q}_G$ and $\mathcal{Q}_F$ and then reduce each set to its equivalent discrete set $\mathcal{Q}_D$. We do not consider the polyhedral set $\mathcal{Q}_B$ because the resulting discrete set will be of an exponential size. Before proceeding, we first present Observation 5.1, 5.2 and Definition 5.1.

Observation 5.1. *If the demand uncertainty set $\mathcal{Q}$ is compact and convex, then $\mathcal{Q}$ can be replaced by* Ext$(\mathcal{Q})$*, where* Ext$(\mathcal{Q})$ *denotes the set of extreme points of $\mathcal{Q}$.*

Proof. Clearly, Ext$(\mathcal{Q}) \subseteq \mathcal{Q}$. We only need to show that, for route $r = (0, v_1, v_2, ..., v_p, n+1)$, if $\sum_{i=1}^{p} q_{v_i} \leq Q$ for all $q \in$ Ext$(\mathcal{Q})$, then $\sum_{i=1}^{p} q'_{v_i} \leq Q$ for all $q' \in \mathcal{Q}$. This is clearly true, owing to the fact that each inequality from the latter can be obtained as a consequence of a convex combination of the former. □

Observation 5.1 simply implies that the demand uncertainty set $\mathcal{Q}$ can be reduced to the set of its extreme points, Ext$(\mathcal{Q})$. Note that if $\mathcal{Q}$ is a polytope, Ext$(\mathcal{Q})$ is a finite set. Furthermore, if $\mathcal{Q}$ has a favorable structure, one may be able to pinpoint Ext$(\mathcal{Q})$.

Definition 5.1. *A demand scenario q dominates another one q' if*

$$\sum_{i=1}^{p} q_{v_i} \geq \sum_{i=1}^{p} q'_{v_i} \text{ for any route } r = (0, v_1, v_2, ..., v_p, n+1).$$

The immediate implication from Definition 5.1 is that if the vehicle capacity constraint along a route r is respected under demand scenario q, then it is also respected under any scenarios dominated by q. It is obvious that the sufficient and necessary condition for this dominance relationship to happen is $q \geq q'$.

Observation 5.2. *If the demand uncertainty set $\mathcal{Q}$ is compact and convex, then $\mathcal{Q}$ can be replaced by $\overline{\text{Ext}}(\mathcal{Q})$, where $\overline{\text{Ext}}(\mathcal{Q})$ denotes the set of extreme points of $\mathcal{Q}$ that are not dominated by others from $\mathcal{Q}$.*

Proof. Combining Observation 5.1 with Definition 5.1, we can claim this observation is valid. □

One can replace the travel time uncertainty set $\mathcal{T}$ by the set of its extreme points, $\text{Ext}(\mathcal{T})$, and define the similar dominance relationship between t and t'. For the sake of interest, we skip this part. Now we perform polyhedral studies on $\mathcal{Q}_G$ and $\mathcal{Q}_F$, and identify $\overline{\text{Ext}}(\mathcal{Q}_G)$ and $\overline{\text{Ext}}(\mathcal{Q}_F)$.

Proposition 5.1. *$\overline{\text{Ext}}(\mathcal{Q}_G)$ is $\mathcal{Q}_D$ with at most $\binom{n}{\Gamma^q}$ scenarios.*

Proof. For the cardinality-constrained set $\mathcal{Q}_G$, the demand vector q is resulted from an affine mapping of ξ, that is, $q = q^0 + \text{diag}(\hat{q}_1, \hat{q}_2, ..., \hat{q}_n)\,\xi$, where $\text{diag}(\cdot)$ denotes a diagonal matrix. In order to locate the extreme points of $\mathcal{Q}_G$, we consider the following polyhedral set

$$\Xi_G := \left\{ \xi \in \mathbb{R}^n : 0 \leq \xi_i \leq 1, \forall i \in \{1, 2, ..., n\}, \sum_{i=1}^{n} \xi_i \leq \Gamma^q \right\}.$$

In what follows, we aim to identify all the extreme points of the set Ξ_G via *polyhedral theory*[1]. Among them, we neglect those points which will result in dominated demand scenarios, using the observation that $q \geq q'$ whenever $\xi \geq \xi'$ since $\hat{q}_1, \hat{q}_2, ..., \hat{q}_n$ are all non-negative.

The set Ξ_G is defined by n pairs of variable-bounding inequalities and an extra budget constraint. It is obvious that the coefficient vectors for any n inequalities are *linearly independent*. Let ξ^* denote an extreme point of Ξ_G. We can claim that the budget constraint must be active at the extreme points of our interest. Otherwise, one can always increase some elements of ξ^* that are currently binding at their lower bounds until the budget constraint becomes active. As a result, we have (i) for all but one $i \in \{1, 2, ..., n\}$, one direction of the bounding inequalities $-1 \leq \xi_i \leq 1$ is chosen to be active at ξ^* (e.g.,

1 For a polyhedral set $X := \left\{ x \in \mathbb{R}^n : a_i^\top x \leq b_i \;\; \forall i \in I \right\}$, a point $x^* \in X$ is an extreme point if and only if among all inequalities that are active at x^* (i.e., $a_i^\top x^* = b_i$), there are n linearly independent a_i's.

either $\xi_i^* = -1$ or $\xi_i^* = 1$); (ii) $\sum_{i=1}^{n} \xi_i^* = \Gamma^q$. This yields an extreme point ξ^* with its Γ^q elements being 1 and others being 0. Considering all permutations, we have $\binom{n}{\Gamma^q}$ unique extreme points ξ^*, each of them having Γ^q 1's and $(n - \Gamma^q)$ 0's as its elements. Let $\overline{\text{Ext}}(\Xi_G)$ denote the set of extreme points. Hence, the polytope $\mathcal{Q}_G$ can be reduced to the subset of its extreme points, each of them resulting from an affine mapping of an extreme point $\xi \in \overline{\text{Ext}}(\Xi_G)$[2]. Therefore, we can conclude that $\overline{\text{Ext}}(\mathcal{Q}_G)$ is a discrete set $\mathcal{Q}_D$ with at most $\binom{n}{\Gamma^q}$ demand scenarios. □

Proposition 5.2. *$\overline{\text{Ext}}(\mathcal{Q}_F)$ is $\mathcal{Q}_D$ with at most $\binom{F}{\kappa^*+1}$ scenarios, where $\kappa^* = (F - \lfloor \beta F \rfloor - 1)/2$ if $F - \lfloor \beta F \rfloor$ is an odd number and $\kappa^* = (F - \lfloor \beta F \rfloor - 2)/2$ otherwise.*

Proof. It is easy to show that $\overline{\text{Ext}}(\mathcal{Q}_F)$ is a singleton when $\beta \in \{0, 1\}$. Now we consider $\beta \in (0, 1)$. In the definition of $\mathcal{Q}_F$, the demand vector q is resulted from an affine mapping of ξ, that is, $q = q^0 + \Psi\xi$. In order to locate the extreme points of $\mathcal{Q}_F$, we consider the following polyhedral set

$$\Xi_F := \left\{ \xi \in \mathbb{R}^F : -1 \leq \xi_f \leq 1 \; \forall f \in \{1, 2, ..., F\}, -\beta F \leq \sum_{f=1}^{F} \xi_f \leq \beta F \right\}.$$

In what follows, we aim to identify all the extreme points of the set Ξ_F as we did for Ξ_G. Among them, we disregard those points which will result in dominated demand scenarios, using the observation that $q \geq q'$ whenever $\xi \geq \xi'$ since every entry of the matrix Ψ is non-negative.

The set Ξ_F is defined by $F + 1$ pairs of linear inequalities: F pairs of variable-bounding inequalities and one pair of summation-bounding constraints. It is obvious that the coefficient vectors for any F pairs of inequalities are linearly independent. Since only non-dominated demand scenarios are of interest, we just need to focus on the budget constraint, $\sum_{f=1}^{F} \xi_f \leq \beta F$. Furthermore, we can claim that the budget constraint must be active at the extreme points of our interest. Otherwise, one can always increase some elements of ξ^* that are currently binding at their lower bounds until the budget constraint becomes active. Let ξ^* denote an extreme point of Ξ_F, hence we have (i) for all but one $f \in \{1, 2, ..., F\}$, one direction of the bounding inequalities $-1 \leq \xi_f \leq 1$ is chosen to be active at ξ^* (e.g., either $\xi_f^* = -1$ or $\xi_f^* = 1$); (ii) $\sum_{f=1}^{F} \xi_f^* = \beta F$. Let $\xi_{f'}^*$ be the element that is not chosen to be binding at bounds and let κ denote the number of ξ_f^*'s which are fixed at their lower bounds. We consider the following two cases:

2 Each extreme point of $\mathcal{Q}_G$ must correspond to at least one extreme point of Ξ_G.

- If $F - \lfloor \beta F \rfloor$ is an odd number, then $\xi^*_{f'} = \beta F - \lfloor \beta F \rfloor$ and other $\kappa + \lfloor \beta F \rfloor$ elements of ξ^* are at their upper bounds;
- If $F - \lfloor \beta F \rfloor$ is an even number, then $\xi^*_{f'} = \beta F - \lfloor \beta F \rfloor - 1$ and other $\kappa + \lfloor \beta F \rfloor + 1$ elements of ξ^* are at their upper bounds.

Correspondingly, $\kappa = (F - \lfloor \beta F \rfloor - 1)/2$ in the former case, and $\kappa = (F - \lfloor \beta F \rfloor - 2)/2$ in the latter. Consider all permutations, we have at most $\binom{F}{\kappa^*+1}$ extreme points ξ^*, where κ^* denotes the value of κ in the applicable case. Let $\overline{\text{Ext}}(\Xi_F)$ denote the set of identified extreme points. Hence, the polytope $\mathcal{Q}_F$ can be reduced to the subset of its extreme points, each of them resulting from an affine mapping of an extreme point $\xi \in \overline{\text{Ext}}(\Xi_F)$[3]. Therefore, we can conclude that $\overline{\text{Ext}}(\mathcal{Q}_F)$ is a discrete set $\mathcal{Q}_D$ with at most $\binom{F}{\kappa^*+1}$ demand scenarios. □

Proposition 5.1 implies that, in theory, $\mathcal{Q}_G$ can be reduced to an equivalent discrete set. However, the resulting set has a size that increases polynomially with the number of customers n, such that the reduction procedure will become impractical very quickly when n increases. In contrast, Proposition 5.2 indicates that the equivalent discrete set for $\mathcal{Q}_F$ has a size that does not depend on n but on the number of factors F. Hence, if F is small, the equivalent set $\mathcal{Q}_D$ will be of a manageable size. This implies that, one can convert a robust VRP with the demand uncertainty set $\mathcal{Q}_F$ to the one with an equivalent discrete set $\mathcal{Q}_D$.

5.5 BRANCH-PRICE-AND-CUT

Before we present the solution approaches to the RVRPTW, we first discuss the state-of-the-art exact algorithm for solving its deterministic counterpart, VRPTW. The BPC method has been gradually accepted as the most efficient exact approach for solving the VRPTW and its variants. In this section, we only highlight the most important ingredients of the BPC algorithm and refer readers to [131, 135] for details.

5.5.1 *Set Partitioning Model*

Let R denote the set of feasible routes for a VRPTW and let c_r denote the cost of traversing route $r \in R$ by a vehicle. Let the parameter δ_{ir} denote the number of times customer $i \in V_c$ is covered in route $r \in R$. Let λ_r be a binary variable indicating whether route $r \in R$

3 Each extreme point of $\mathcal{Q}_F$ must correspond to at least one extreme point of Ξ_F.

is selected in the optimal solution. The VRPTW can be formulated as the following *set partitioning* model (5.8) - (5.11).

$$\min_{\lambda_r} \quad \sum_{r \in R} c_r \lambda_r \tag{5.8}$$

$$\text{s.t.} \quad \sum_{r \in R} \delta_{ir} \lambda_r = 1 \qquad \forall i \in V_c \tag{5.9}$$

$$\sum_{r \in R} \lambda_r \leq K \tag{5.10}$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in R \tag{5.11}$$

The objective function (5.8) is to minimize the total cost for selected routes. The degree constraints (5.9) guarantee that every customer is served exactly once and the fleet size constraint (5.10) enforces that at most K vehicles are used. Constraints (5.11) are simply to enforce binarity of variables. We emphasize that both capacity feasibility and time window feasibility are ensured via the definition of "feasible routes".

It is well-known that one can relax the feasible space of the above set partitioning model by including non-elementary vehicle routes (i.e., relaxing condition **C1**) in R without sacrificing optimality. In particular, we replace R with the set of so-called *ng*-routes that are not necessarily elementary [26]. In the reminder of our chapter, we use R to denote the set of *ng*-feasible routes. Since there exists exponentially many *ng*-feasible routes, the formulation (5.8) - (5.11) is a mixed-integer linear programming model with a huge number of binary variables. To address this issue, the LP relaxation of an exponential size at each node of the branch-and-bound tree will be tackled via *column generation* [119]. Valid inequalities are dynamically separated and added so as to strengthen the LP relaxations, thus the set-partitioning model is solved via the *branch-price-and-cut* algorithm.

5.5.2 *A Branch-Price-and-Cut Algorithm*

In the BPC algorithm, the LP relaxations at every node in the branch-and-bound tree are solved via column generation, while cutting planes are added to strengthen the relaxations. We first replace the binarity constraints (5.11) in the set partitioning model by non-negativity constraints and obtain the LP relaxation, which is usually called a *master problem*. The master problem has a huge number of variables as exponentially many feasible routes are typically available. Generating all of these routes to explicitly define the master problem is obviously impractical, thus we resort to column generation. In our

context, columns denote *ng*-feasible routes and we will use them interchangeably. We first consider a *restricted master problem* (RMP) defined by a subset of *ng*-routes $\bar{R} \subseteq R$. After optimizing the RMP, columns with negative reduced costs will be appended to $\bar{R}$ and the resulting RMP is reoptimized. This procedure iterates until no such columns exist. In that case, we say that column generation converges and the master problem has achieved its optimality. If the solution to the master problem is fractional, we then turn to valid inequalities or branching. In particular, *rounded capacity inequalities* [110] and *limited-memory subset row cuts* [99, 131, 133] are often considered to tighten the LP relaxations; branching on the number of used vehicles is usually prioritized over branching on edges [71, 146]. Readers are referred to Chapter 4 for a flowchart representation of the BPC algorithm.

5.5.3 *Pricing Subproblems*

After the RMP is solved, we need to check whether it is necessary to enlarge $\bar{R}$ by including some neglected *ng*-feasible routes that may potentially improve the RMP's objective value. This entails identifying columns with negative reduced costs, which is achieved by solving a *pricing subproblem*.

In our context, the pricing subproblem can be modeled as a *shortest path problem with resource constraints* (SPPRC) [138]. The SPPRC is defined on a directed graph $G = (V, A)$. We associate demand q_i, capacity Q and time window $[e_i, l_i]$ with vertex $i \in V$. Note that $q_i = 0$ if vertex i denotes a depot. Associated with arc $(i, j) \in A$ are the cost $\bar{c}_{ij}$, travel time t_{ij}. The cost $\bar{c}_{ij}$ is obtained by properly modifying c_{ij} in order to account for the contribution from current dual values to constraints (5.9) and (5.10). While arc $(i, j) \in A$ is traversed by a path, time resource $(s_i + t_{ij})$ and capacity resource q_j are consumed. The resource accumulated consumption until a vertex visited along a path should not exceed its limit. The goal is to determine the minimum-cost path among all paths that start from the vertex 0 and end at the vertex $n + 1$ such that resource constraints are respected.

It is well-known that the SPPRC is weakly $\mathcal{NP}$-hard [68]. The most successful solution approach is a *dynamic programming* method called the *labeling algorithm*, which has a pseudo-polynomial time complexity [138]. The labeling algorithm works as follows. We associate an *ng*-feasible partial path P with a label $L(P) := (pred(P), \bar{c}(P), v(P), \Pi(P), d(P), a(P))$ that stores a pointer to its predecessor label, reduced cost, end vertex, a set of forbidden vertices, total vehicle load and earliest time to start servicing. We initialize the labeling algorithm by storing the first label $(null, 0, 0, \emptyset, 0, 0)$ into a pool. Choosing a label $L(P)$

from the pool, we attempt to extend it to vertex $j \in V \setminus \Pi(P)$ with $(v(P), j) \in A$ for generating a new label P', using the following extension procedure:

$$pred(P') \leftarrow L(P), \tag{5.12}$$
$$\bar{c}(P') \leftarrow \bar{c}(P) + \bar{c}_{v(P)j}, \tag{5.13}$$
$$v(P') \leftarrow j, \tag{5.14}$$
$$\Pi(P') \leftarrow \{\Pi(P) \cap NG(j)\} \cup \{j\}, \tag{5.15}$$
$$d(P') \leftarrow d(P) + q_j, \tag{5.16}$$
$$a(P') \leftarrow \max\left\{e_j, a(P) + s_{v(P)} + t_{v(P)j}\right\}, \tag{5.17}$$

where $NG(j) \subseteq V_c$ denote the *ng*-set for node j. Usually, the *ng*-set is chosen to be the set of nearest neighbors as [26] suggested. Before storing label $L(P')$ to the pool, we check whether this is a feasible extension, namely, whether the resource consumption constraints are respected, i.e., $d(P') \leq Q$ and $a(P') \leq l_j$. If that is the case, we then obtain a new *ng*-feasible label and store it in the pool. When no more label extension can be made, we collect all paths that end at vertex $n+1$ and return those with negative reduced costs.

To accelerate the labeling algorithm, it is crucial to utilize the dominance relationship to avoid some label extensions. Before storing a new *ng*-feasible label, we first check whether it dominates or is dominated by existing labels stored in the pool. We say that label $L(P_1)$ dominates another label $L(P_2)$ if for any feasible path extended from P_2, we can always find a feasible path extension from P_1 and this extended path is not more costly. Sufficient conditions for this are given by (5.18) - (5.20).

$$\bar{c}(P_1) \leq \bar{c}(P_2), \tag{5.18}$$
$$v(P_1) = v(P_2), \tag{5.19}$$
$$\Pi(P_1) \subseteq \Pi(P_2), \tag{5.20}$$
$$d(P_1) \leq d(P_2), \tag{5.21}$$
$$a(P_1) \leq a(P_2). \tag{5.22}$$

A newly generated *ng*-feasible label $L(P)$ is saved only if it is not dominated, and existing labels that are dominated by $L(P)$ will be removed from the pool. Since checking dominance relationship is computationally expensive, we do not perform it too aggressively. Usually, labels of proximity in resource consumption levels are saved into a bucket and dominance check is only applied among labels residing in the same or neighboring buckets[131, 147].

Whenever a pricing subproblem is solved exactly, one can always obtain a *Lagrangian dual bound* [119]. As [25, 56] suggested, when the primal-dual gap becomes small, one can enumerate all elementary columns with reduced costs less than the primal-dual gap, since only these columns may contribute to a solution better than the incumbent. This step is called *route enumeration*. It works in a similar fashion as the aforementioned labeling algorithm except that: (i) the extension rule (5.15) is updated to be $\Pi(P') \leftarrow \Pi(P) \cup \{j\}$, so as to produce elementary routes; (ii) sufficient conditions (5.18) and (5.20) are replaced by (5.23) and (5.24), respectively.

$$c(P_1) \leq c(P_2), \tag{5.23}$$

$$\Pi(P_1) = \Pi(P_2), \tag{5.24}$$

where $c(P)$ denote the actual monetary cost for traversing path P.

In order to expedite the labeling algorithm, various advanced pricing techniques such as heuristic pricing [77], bidirectional labeling [142], and variable fixing [98], among others, have been proposed. If strengthening inequalities are incorporated to the RMP, then proper modifications have to be made when solving the SPPRC [131]. To further expedite the BPC algorithm, primal heuristics [148], stabilized column generation [134], dynamic ng-set [143], and strong branching [147] are also considered. We refer readers to [135] for details.

5.6 SOLUTION APPROACHES

In this section, we first review existing approaches in the literature for adapting the BPC algorithm to solve robust VRPs under uncertainty; we then propose a new approach that can tackle these problems under various types of uncertainty sets we discussed in Section 5.3. These approaches can be categorized into two groups: a *robust pricing* approach and a *robust cutting-plane* approach. The robust pricing approach aims to ensure robust feasibility of routing designs in pricing subproblems, while the robust cutting-plane approach seeks for robust feasibility via dynamically enforcing necessary constraints in master problems. Clearly, robust feasibility of a solution is warranted from two intrinsically different perspectives. For notation convenience, let $\mathcal{R}$ denote the set of *robust ng-feasible* routes with respect to any realization $(q, t) \in \mathcal{Q} \times \mathcal{T}$. That is, for route $r \in \mathcal{R}$, robust feasibility conditions **(C2)** and **(C3)** are satisfied.

5.6.1 *Robust Pricing Approach*

The robust pricing approach is based on the set partitioning model (5.8) - (5.11) but with a set of robust *ng*-feasible routes $\mathcal{R}$, rather than the set of deterministic *ng*-feasible routes R. Therefore, the BPC algorithm remains the same except that, in order to dynamically introduce robust *ng*-feasible routes, the resulting pricing subproblem becomes a *robust shortest path problem with resource constraints*. Since robust feasibility of optimal routes is ensured when these routes were generated in pricing subproblems, we call this a *robust pricing* approach. We review two methods proposed in the literature that have demonstrated success in solving the RSPPRC with a few uncertainty sets from Section 5.3.

5.6.1.1 *Direct Pricing*

As [11] pointed out, the RSPPRC under resource uncertainty is strongly $\mathcal{NP}$-hard for arbitrary uncertainty sets. In this section, we first focus on cardinality-constrained uncertainty sets $\mathcal{Q}_G$ and $\mathcal{T}_G$. To tackle the RVRPTW in which the uncertain demand vector q and uncertain travel time vector t fall into respective cardinality-constrained sets $\mathcal{Q}_G$ and $\mathcal{T}_G$, the authors of [126] proposed a modified labeling algorithm to directly solve the resulting RSPPRC. Hence, we call this a *direct pricing* method. Their proposed labeling procedure is similar to its deterministic counterpart we discussed in Section 5.5.3 but with the following modifications. First, for a given path P, the labeling notation is redefined as $L(P) := (pred(P), \bar{c}(P), v(P), \Pi(P), d_0(P), d_1(P), ..., d_{\Gamma^q}(P), a_0(P), a_1(P), ..., a_{\Gamma^t}(P))$, where the first four terms keep their same meanings as before while $d_\gamma(P)$ for $\gamma \in \{0, 1, ..., \Gamma^q\}$ denotes the the maximum total vehicle load along path P when up to γ customers' demands attain their maximum values, and $a_\gamma(P)$ for $\gamma \in \{0, 1, ..., \Gamma^t\}$ denotes the worst-case earliest time to start servicing node i, considering that up to γ travel times attain their maximum values. Second, when extending path P to node j for creating a new path P', the extension procedures (5.16) - (5.17) are replaced by (5.25) - (5.28).

$$d_0(P') \leftarrow d_0(P) + q_j, \tag{5.25}$$

$$d_\gamma(P') \leftarrow \max\{d_\gamma(P) + q_j, d_{\gamma-1}(P) + q_j + \hat{q}_j\} \qquad \forall \gamma \in \{1, 2, ..., \Gamma^q\}, \tag{5.26}$$

$$a_0(P') \leftarrow \max\left\{e_j, a_0(P) + s_{v(P)} + t_{v(P)j}\right\}, \tag{5.27}$$

$$a_\gamma(P') \leftarrow \max\left\{e_j, \max\left\{a_\gamma(P), a_{\gamma-1}(P) + \hat{t}_{v(P)j}\right\} + s_{v(P)} + t_{v(P)j}\right\} \quad \forall \gamma \in \{1, 2, ..., \Gamma^t\}. \tag{5.28}$$

Hence, the resulting label P' can be accepted as a feasible extension only if $d_{\Gamma^q}(P') \leq Q$ and $a_{\Gamma^t}(P') \leq l_j$. Last, sufficient conditions (5.21) and (5.22) for checking dominance should be replaced by conditions (5.29) and (5.30), respectively.

$$d_\gamma(P_1) \leq d_\gamma(P_2) \qquad \forall \gamma \in \{0, 1, ..., \Gamma^q\}, \tag{5.29}$$

$$a_\gamma(P_1) \leq a_\gamma(P_2) \qquad \forall \gamma \in \{0, 1, ..., \Gamma^t\}. \tag{5.30}$$

The direct pricing method could effectively solve the RSPPRC due to the following observation: for demand uncertainty set $\mathcal{Q}_G$ (travel time uncertainty set $\mathcal{T}_G$), one can efficiently compute the maximum vehicle load (worst-case earliest service start time) along a path by introducing Γ^q (Γ^t) extra relevant resources into a deterministic SPPRC and keeping track of their consumption. In particular, every path-extending operation can be achieved in $\mathcal{O}(\Gamma^q + \Gamma^t)$ time. The immediate implication from dominance rules (5.29) and (5.30) is that when compared with the deterministic case, path P_1 is "less likely" to dominate path P_2 due to more restrictive sufficient conditions. As a result, more labels will be kept and processed in the modified labeling algorithm, causing an increase in its time complexity.

One can simply apply the direct pricing idea when the demand/travel time vector falls into the discrete support $\mathcal{Q}_D/\mathcal{T}_D$. Correspondingly, the pricing subproblem entails the enforcement of $\left(D^q + D^t\right)$ resource constraints. Similarly, the path-extending operation could be achieved in $\mathcal{O}\left(D^q + D^t\right)$ time. Meanwhile, as we have pointed out, we will expect a time complexity increase of the modified labeling algorithm. This implies that for a discrete set $\mathcal{Q}_D$ that consists of a large number of scenarios, solving the corresponding RSPPRC via a direct pricing method will become prohibitive.

5.6.1.2 *Transformed Pricing*

In this section, we consider that the travel time vector t is constant (i.e., $|\mathcal{T}| = 1$) and that the customer demand vector q can take any value from a non-empty knapsack set, $\mathcal{Q}_K$, given by (5.31).

$$\mathcal{Q}_K := \left\{ q \in \left[\underline{q}, \overline{q}\right] : \sum_{i \in V_c} a_{li} q_i \leq b_l, \forall l \in \{1, 2, ..., L\} \right\}, \tag{5.31}$$

where $a_{li} \geq 0$ for $i \in V_c$ and $l \in \{1, 2, ..., L\}$. Note that this knapsack set is a more generalized uncertainty set and can be reduced to (i) $\mathcal{Q}_G$ when $\underline{q} = q^0, \overline{q} = q^0 + \hat{q}, L = 1, a_{li} = 1/\hat{q}_i$, and $b_l = \Gamma^q + \sum_{i \in V_c} q_i^0/\hat{q}_i$; (ii) $\mathcal{Q}_B$ when $a_{li} = 1$ if $i \in B_l$ and 0 otherwise.

The work of [136] was focused on solving a robust VRP with knapsack uncertainty of form (5.31) via the BPC approach and the authors presented the following Proposition 5.3.

Proposition 5.3. *If $\mathcal{Q} = \mathcal{Q}_K$ and $|\mathcal{T}| = 1$, then solving the RSPPRC is equivalent to solving at most $\sum_{l=0}^{L} \binom{L}{l}\binom{n}{L-l}$ deterministic SPPRCs.*

Proof. For an elementary route $r \in \mathcal{R}$, robust capacity feasibility condition (**C2**) is satisfied, as shown by (5.32).

$$\max_{q \in \mathcal{Q}_K} \sum_{i \in V_c} \delta_{ir} q_i \ \leq Q \tag{5.32}$$

Plugging in set $\mathcal{Q}_K$ and replacing q by $\underline{q} + \eta$, where $0 \leq \eta \leq \bar{q} - \underline{q}$, we expand the left hand side of inequality (5.32) into an LP problem (5.33).

$$\begin{aligned} \max_{\eta \geq 0} \quad & \sum_{i \in V_c} \delta_{ir} \left(\underline{q}_i + \eta_i \right) & & \\ \text{s.t.} \quad & \eta_i \leq \bar{q}_i - \underline{q}_i & & \forall i \in V_c \\ & \sum_{i \in V_c} a_{li} \eta_i \leq b_l - \sum_{i \in V_c} a_{li} \underline{q}_i & & \forall l \in \{1, 2, ..., L\} \end{aligned} \tag{5.33}$$

Since $\mathcal{Q}_K$ is a nonempty and bounded set, strong duality holds for problem (5.33). We introduce dual variables $z \in \mathbb{R}^n_{\geq 0}$ and $\theta \in \mathbb{R}^L_{\geq 0}$ and obtain its LP dual, denote by (5.34).

$$\begin{aligned} \min_{z \geq 0, \theta \geq 0} \quad & \sum_{i \in V_c} \delta_{ir} \underline{q}_i + \sum_{i \in V_c} \left(\bar{q}_i - \underline{q}_i \right) z_i + \sum_{l=1}^{L} \left(b_l - \sum_{i \in V_c} a_{li} \underline{q}_i \right) \theta_l & & \\ \text{s.t.} \quad & z_i + \sum_{l=1}^{L} a_{li} \theta_l \geq \delta_{ir} & & \forall i \in V_c \end{aligned} \tag{5.34}$$

Reformulating problem (5.34) by eliminating z_i and considering $\delta_{ir} \in \{0, 1\}, a_{li} \geq 0$, we obtain an equivalent problem (5.35).

$$\min_{\theta \geq 0} \left\{ \sum_{i \in V_c} \left(\underline{q}_i + \left(\bar{q}_i - \underline{q}_i \right) \max \left\{ 0, 1 - \sum_{l=1}^{L} a_{li} \theta_l \right\} \right) \delta_{ir} + \sum_{l=1}^{L} \left(b_l - \sum_{i \in V_c} a_{li} \underline{q}_i \right) \theta_l \right\} \tag{5.35}$$

Let $f(\theta; r)$ denote the objective function of problem (5.35). A key observation is that $f(\theta; r)$ is a piece-wise linear function on domain $\mathbb{R}^L_{\geq 0}$; thus $f(\theta; r)$ achieves its minimum at some breaking point $\theta' \in \mathbb{R}^L_{\geq 0}$. Let $\Theta \subseteq \mathbb{R}^L_{\geq 0}$ denote the set of breaking points, each of which

can be identified as a solution to a subsystem of L linearly independent equations among the following $L + n$ equations:

$$\begin{aligned} \theta_l &= 0 && \forall l \in \{1, 2, ..., L\}, \\ 1 - \sum_{l=1}^{L} a_{li}\theta_l &= 0 && \forall i \in V_c. \end{aligned}$$

Note that $|\Theta| \leq \sum_{l=0}^{L} \binom{L}{l}\binom{n}{L-l}$. For $\theta \in \Theta$, let R^θ denote a set of routes such that for any route $r \in R^\theta$: (i) ng-feasibility is satisfied [4]; (ii) time window constraints are respected along this route; (iii) $f(\theta; r) \leq Q$ [5]. Then one can easily show $\mathcal{R} = \cup_{\theta\in\Theta} R^\theta$. □

Proposition 5.3 indicates that solving a RSPPRC can be transformed into solving *polynomially many* deterministic SPPRCs [6]. Hence, we call this a *transformed pricing* method. We make a few remarks here. First, the resulting SPPRC has roughly the same time complexity as the one we discussed in Section 5.5.3, since condition (iii) for the definition of R^θ is enforced exactly like a capacity constraint. Second, the number of transformed SPPRCs increases polynomially but not mildly with the number of customers n. If the demand vector q is supported on a general knapsack support $\mathcal{Q}_K$, solving the RSPPRC via the transformed approach will still be impractical when n increases. Third, for uncertainty sets of our interest, if $\mathcal{Q}_K = \mathcal{Q}_G$, $|\Theta| \leq n$[7]; if $\mathcal{Q}_K = \mathcal{Q}_B$, $|\Theta| \leq 2^L$. This indicates that the RSPPRC under $\mathcal{Q}_G / \mathcal{Q}_B$ is weakly $\mathcal{NP}$-hard and that one can efficiently solve the relevant RSPPRC via the transformed pricing method, as [118, 136] did.

5.6.2 *Robust Cutting-Plane Approach*

Another perspective to ensure robust feasibility of routing designs is through cutting planes. Let $\widetilde{\mathcal{Q}} \subseteq \mathcal{Q}$ and $\widetilde{\mathcal{T}} \subseteq \mathcal{T}$ denote finite sets, respectively. Let $\widetilde{\mathcal{R}}$ denote the set of

4 Relaxing the elementary condition (**C1**) is again permitted due to degree constraints (5.9).

5 $f(\theta; r) \leq Q$ can be properly mapped as a capacity constraint in the resulting SPPRC defined by θ. In particular, a demand of $\left(\underline{q}_i + \left(\overline{q}_i - \underline{q}_i\right) \max\left\{0, 1 - \sum_{l=1}^{L} a_{li}\theta_l\right\}\right)$ is assigned to customer $i \in V_c$ and the vehicle capacity becomes $Q - \sum_{l=1}^{L}\left(b_l - \sum_{i\in V_c} a_{li}\underline{q}_i\right)\theta_l$.

6 The number of deterministic SPPRCs that have to be solved might be reduced, e.g., one can deduce that $R^{\theta'} = \emptyset$ for some $\theta' \in \Theta$. Readers are referred to [136] for details.

7 This bound can be further tightened to be $\lceil (n - \Gamma^q)/2 \rceil + 1$, see [114]

ng-feasible routes with respected to any realization $(q, t) \in \widetilde{\mathcal{Q}} \times \widetilde{\mathcal{T}}$. That is, for route $r = \{0, v_1, v_2, ..., v_p, n+1\} \in \widetilde{\mathcal{R}}$:

$$\sum_{i=1}^{p} q_{v_i} \leq Q \text{ for all } q \in \widetilde{\mathcal{Q}},$$

$$a_i(t) \leq l_i \text{ for all } i \in \{v_1, v_2, ..., v_p, n+1\} \text{ and for all } t \in \widetilde{\mathcal{T}},$$

where $a_i(t)$ denotes the earliest service start time at node i under scenario t. Based on this definition, we have two observations: (i) $\mathcal{R} \subseteq \widetilde{\mathcal{R}}$, that is, all robust feasible routes are included within $\widetilde{\mathcal{R}}$; (ii) since capacity feasibility and time window feasibility are only ensured against a subset of anticipated scenarios, robust feasibility conditions **(C2)** and **(C3)** are not guaranteed yet. In order to forbid the selection of routes $r \in \widetilde{\mathcal{R}} \setminus \mathcal{R}$ in the optimal solution, we rely on the enforcement of necessary constraints in the master problem. In particular, we utilize *robust rounded capacity inequalities* (robust RCI) and *infeasible path elimination constraints* (IPEC) to ensure robust capacity feasibility and robust time window feasibility, respectively. Since the number of necessary constraints is exponentially many, these constraints are separated and introduced dynamically at every branch-and-bound node. Hence, we call this a *robust cutting-plane* approach. We emphasize that adding these necessary constraints to a RMP will not complicate the solution of pricing subproblems because their corresponding dual values can be properly accommodated into the modified arc cost in the SPPRC. Furthermore, these dual values will progressively guide the pricing engine to generate robust feasible routes. To summarize, our BPC algorithm starts with a subset of vehicle routes (not necessarily robust feasible) and iteratively introduces some neglected routes via solving the deterministic SPPRC under finitely many resource constraints, each corresponding to an element from $\widetilde{\mathcal{Q}}$ or $\widetilde{\mathcal{T}}$; when column generation converges, IPEC and robust RCI are separated and added to prohibit accepting solutions that are not robust feasible. Next, we will discuss two key steps: (i) the choice of $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{T}}$; (ii) separating IPEC and robust RCI.

5.6.2.1 *Choosing $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{T}}$*

Since both $\mathcal{Q}$ and $\mathcal{T}$ are compact and convex sets, we can replace them by $\text{Ext}(\mathcal{Q})$ and $\text{Ext}(\mathcal{T})$, respectively. Thus, our goal is to choose $\widetilde{\mathcal{Q}} \subseteq \text{Ext}(\mathcal{Q})$[8] and $\widetilde{\mathcal{T}} \subseteq \text{Ext}(\mathcal{T})$. When $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{T}}$ contain more scenarios, routes generated in pricing subproblems are "more likely" to be robust feasible, and consequently less effort will be made to separate necessary

8 One may be only interested in $\widetilde{\mathcal{Q}} \subseteq \overline{\text{Ext}}(\mathcal{Q})$. However, given an extreme point $q \in \text{Ext}(\mathcal{Q})$, it is generally not convenient to determine whether q is dominated by any other scenarios from $\text{Ext}(\mathcal{Q})$ or not.

constraints; however, pricing subproblems correspondingly become computationally more expensive since the time complexity of an SPPRC generally increases with the number of resource constraints that are enforced. Next, we encapsulate a proper number of selected demand and travel time scenarios (denoted by $\mathtt{NS}^q$ and $\mathtt{NS}^t$) into $\widetilde{\mathcal{Q}}$ and $\widetilde{\mathcal{T}}$, respectively, so as to balance the efforts of solving pricing subproblems and separating necessary constraints.

- $\mathcal{Q}_G$. We use the *k-means clustering* method to partition the customer set V_c into $\mathtt{NS}^q$ clusters based on their coordinates. Customers from each cluster are sorted by their worst-case demands in descending order. Assign $\xi_i = 1$ to the first Γ^q customers from a selected cluster and $\xi_i = 0$ to other customers, yielding a demand scenario q^*. This scenario denotes the case where only the demands for Γ^q geographically close customers achieve their maximum upward deviations and other demands are kept nominal. Clearly, $q^* \in \mathrm{Ext}\,(\mathcal{Q}_E)$. Let $\widetilde{\mathcal{Q}}$ be a set of such demand scenarios.

- $\mathcal{Q}_B$. We sort customers from each partitioned set B_l by their worst-case demands in descending order. We fix the demands for higher-ranking and lower-ranking customers at their upper and lower bounds, respectively, and allow exactly one customer demand to take a value within its bounds, such that the the budget constraint is active. This yields a scenario q^*. One can easily show that $q^* \in \mathrm{Ext}\,(\mathcal{Q}_B)$. Taking the reverse direction, we fix the demands for lower-ranking and higher-ranking customers at their upper and lower bounds, respectively. Considering all possible combinations among different customer subsets B_l, we have in total 2^L scenarios. We sort them, in descending order, by the number of customer demands that have achieved their upper bounds. Let $\widetilde{\mathcal{Q}}$ be a set of the first $\mathtt{NS}^q$ scenarios.

- $\mathcal{Q}_F$. In the proof of Proposition 5.2, we have identified the set $\overline{\mathrm{Ext}}\,(\mathcal{Q}_F)$. We sort the elements from $\overline{\mathrm{Ext}}\,(\mathcal{Q}_F)$ by total customer demands in descending order. Let $\widetilde{\mathcal{Q}}$ be the set of the first $\mathtt{NS}^q$ scenarios.

- $\mathcal{Q}_E$. We consider the following optimization problem: the objective is to maximize a weighted sum of customer demands q_i that is subject to $q \in \mathcal{Q}_E \cap \{q \in \mathbb{R}^n : q \geq q^0\}$. We choose the corresponding weight for q_i to be $1/q_i^0$. Optimizing this problem will yield a extreme demand vector q^* that dominates q^0. Let $\widetilde{\mathcal{Q}}$ be a singleton with q^*. Hence, $\mathtt{NS}^q = 1$.

- $\mathcal{Q}_D$. We first eliminate from $\mathcal{Q}_D$ any scenarios that are either dominated or not extreme points (e.g., check each one via solving a linear program) and then sort the

remaining scenarios by the total customer demands in descending order. Let $\widetilde{\mathcal{Q}}$ be a set of the first $\texttt{NS}^q$ scenarios. Clearly, $\widetilde{\mathcal{Q}} \subseteq \text{Ext}\,(\mathcal{Q}_D)$.

- $\mathcal{T}_G$. We assign $\xi_{0j} = 1$ to the first Γ^t minimum-cost arcs $(0, j) \in A$ such that $t^0_{0j} + \hat{t}_{0j} > e_j$, and assign $\xi_{ij} = 0$ to the remaining arcs, yielding a travel time vector t^*. This represents the case where the travel times for some out-going arcs from the depot achieve their maximum upward deviations and others are kept nominal. Clearly, $t^* \in \text{Ext}\,(\mathcal{T}_G)$. Let $\mathcal{T}$ be a singleton with t^*. Hence, $\texttt{NS}^t = 1$.
- $\mathcal{T}_D$. We first eliminate from $\mathcal{T}_D$ any scenarios that are either dominated[9] or not extreme points and then sort the remaining scenarios by the total arc traversal time in descending order. Let $\widetilde{\mathcal{T}}$ be a set of the first $\texttt{NS}^t$ scenarios. Clearly, $\widetilde{\mathcal{T}} \subseteq \text{Ext}\,(\mathcal{T}_D)$.

We can now empirically adjust the time complexity of pricing subproblems by simply controlling $\texttt{NS}^q$ and $\texttt{NS}^t$, the number of resource constraints that have to be enforced in the pricing step.

5.6.2.2 *Robust Rounded Capacity Inequalities*

For notional convenience, we introduce arc-based variable x_{ij} for $(i, j) \in A$ and relate it to route-based variables λ_r, as shown by (5.36).

$$x_{ij} = \sum_{r \in \widetilde{\mathcal{R}}} \tau_{ijr} \lambda_r \quad \forall (i, j) \in A \tag{5.36}$$

To ensure robust feasibility of selected routes against demand uncertainty, the authors of [82, 159] proposed to enforce robust RCI (as shown by (5.37)) in a branch-and-cut framework.

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq \left\lceil \frac{1}{Q} \max_{q \in \mathcal{Q}} \sum_{i \in S} q_i \right\rceil \quad \forall S \subseteq V_c \tag{5.37}$$

In this work, we adopt the same idea but enforce it in the context of BPC. The robust RCI for robust vehicle routing are natural extensions of classic RCI for deterministic VRPs, hence their validity is obvious. We now focus on the separation routine. When $\bar{x}_{ij}$ variables are integral, one can simply loop through all vehicle routes and check whether robust capacity feasibility is satisfied or not. In contrast, the exact separation of robust RCI at a fractional solution $\bar{x}$ is a $\mathcal{NP}$-complete problem, since separating its deterministic

9 We use the fact that t dominates t' if $t \geq t'$.

counterpart itself is $\mathcal{NP}$-complete [20]. Hence, solving the separation problem exactly is computationally prohibitive and we resort to heuristic methods.

The work of [82] proposed a tabu search algorithm to identify violated robust RCI when $\mathcal{Q} = \mathcal{Q}_B, \mathcal{Q}_F$. The authors of [159] extended this to all demand uncertainty sets presented in Section 5.3. In this work, we adopt the separation routine from [159]. In particular, the separation procedure starts with a randomly selected customer $S \subseteq V_c$ and then iteratively perturbs this set through a sequence of operations in which individual customers are added or removed. In each iteration, the separation algorithm greedily chooses a customer whose inclusion or removal maximizes the slack of the robust RCI (5.37). Computing this slack requires the computation of the right hand side which, in turn, requires the efficient evaluation of the worst-case load over the current candidate set of customers S, i.e., $\max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$. Generally speaking, maximizing a linear objective over a convex set has a polynomial time complexity [34]. Fortunately, for those convex uncertainty sets of our interest, this problem can be optimized analytically in roughly linear time by exploiting the structure of a given uncertainty set. Furthermore, since the set S is resulted from perturbing another set S' by adding or removing a single customer, we can then apply a quick incremental or decremental update on the worst-case load for S' and obtain the counterpart for S, which is more efficient than computing it from scratch. Table 5.1 presents the closed-form expressions, time and storage complexities for computing the worst-case load of a given customer set S under each of aforementioned demand uncertainty sets in Section 5.3. Our algorithm also maintains a tabu lists of customers that have recently been added or removed to avoid cycling and to escape local optima. Readers are referred to [159] for more implementation details.

We remark that when the travel time vector t is deterministic, i.e., $|\mathcal{T}| = 1$, sufficient conditions (5.19) and (5.21) - (5.24) for claiming dominance of path P_1 over path P_2 during route enumeration is still valid[10]. Their certification comes from the fact that if P_1 dominates P_2, then both paths cover exactly the same customer set (i.e., condition (5.24)) and hence the vehicle loads are equal under any demand realization $q \in \mathcal{Q}$. After route enumeration, only those robust feasible routes will be kept for consideration.

5.6.2.3 *Infeasible Path Elimination Constraints*

In applications of routing with time windows, *infeasible path elimination constraints* [101] are often introduced to forbid routes along which time window conditions are overridden.

10 This was also pointed out in [135], in which capacity inequalities were enforced as necessary constraints when solving the VRPTW via a BPC algorithm.

Table 5.1: Closed-form expressions, time and storage complexities for computing the worst-case load of a vehicle route

$\mathcal{Q}$	Closed-form expression for $\max_{q \in \mathcal{Q}} \sum_{i \in S} q_i$	Update	
		Time	Storage
$\mathcal{Q}_G$	$\sum_{i \in S} q_i^0 + \sum_{l=1}^{\min\{\lvert S \rvert, \Gamma^q\}} \hat{q}_{g_l}$, where $g_1, ..., g_{\lvert S \rvert}$ represents an ordering of the customers in S such that $\hat{q}_{g_1} \geq ... \geq \hat{q}_{g_{\lvert S \rvert}}$	$\mathcal{O}(\log \lvert S \rvert)$	$\mathcal{O}(n)$
$\mathcal{Q}_B$	$\sum_{i \in S} \bar{q}_i - \sum_{l=1}^{L} \max \left\{ 0, \sum_{i \in S \cap B_l} \left(\bar{q}_i - \underline{q}_i \right) - \left(b_l - \sum_{i \in B_l} \underline{q}_i \right) \right\}$	$\mathcal{O}(1)$	$\mathcal{O}(L)$
$\mathcal{Q}_F$	$\sum_{i \in S} q_i^0 - \min \left\{ \sum_{f=1}^{F} \sum_{i \in S} \Psi_{if} - \lambda + \beta F \lvert \lambda \rvert : \lambda \in \left\{ 0, \sum_{i \in S} \Psi_{if_{l^+}}, \sum_{i \in S} \Psi_{if_{l^-}} \right\} \right\}$, where $f_1, ..., f_F$ represents an ordering of the factors such that $\sum_{i \in S} \Psi_{if_1} \geq ... \geq \sum_{i \in S} \Psi_{if_F}$, and $l^+ = \lceil (1+\beta)F/2 \rceil, l^- = \max\{1, \lceil (1-\beta)F/2 \rceil\}$	$\mathcal{O}(F \log F)$	$\mathcal{O}(F)$
$\mathcal{Q}_E$	$\sum_{i \in S} q_i^0 + \left\| \sum_{i \in S} \Sigma_i^{1/2} \right\|_2$ where $\Sigma_i^{1/2}$ denotes the i^{th} column of $\Sigma^{1/2}$ and $\lvert\lvert \cdot \rvert\rvert_2$ denotes the l_2-norm of a vector	$\mathcal{O}(n)$	$\mathcal{O}(n)$
$\mathcal{Q}_D$	$\max \left\{ \sum_{i \in S} q_i^d : d \in \{1, 2, ..., D^q\} \right\}$	$\mathcal{O}(D^q)$	$\mathcal{O}(D^q)$

Let a sequence of vertices $P = (v_1, v_2, ..., v_p)$ denote an elementary path, and let $A_P \subseteq A$ denote the set of arcs that are traversed by path P. Note that path P does not necessarily start or end at a depot. Let $\mathcal{P}$ denote the set of paths that are deemed infeasible with respect to time window constraints. One can simply forbid these paths via enforcing IPEC given by (5.38).

$$\sum_{(i,j)\in A_P} x_{ij} \leq |A_P| - 1 \quad \forall P \in \mathcal{P} \tag{5.38}$$

It is well-known that one can lift IPEC to *tournament inequalities* [19], due to degree constraints (5.9). In particular, we replace the subscript of the summation with $(i, j) \in tr.cl.(P)$, where $tr.cl.(P)$ denotes the *transitive closure* of path P. This so-called tournament form of the inequality is stronger than the version presented above, hence we always do so in the remainder of this chapter. To tackle a RVRPTW, the work of [6] applied IPEC in the branch-and-cut framework to forbid those routes along which time window constraints are violated. In this work, we adopt the same idea but enforce them in the BPC framework.

Separating IPEC at an integral solution $\bar{x}$ is trivial. Suppose we are given a fractional solution $\bar{x}$, it can be shown that there are polynomially many paths P for which inequalities (5.38) are violated [19]. These paths can be easily detected by a simple enumeration procedure. In particular, the algorithm starts with choosing $i \in V$ as a root node and then runs a depth-first search for extending paths. If a violated tournament inequality is found for some path P', we then check whether time window feasibility is respected (i.e., $P' \in \mathcal{P}$) under any travel time realization $t \in \mathcal{T}_G$. This can be achieved in $\mathcal{O}\left(\Gamma^t\right)$ time, using proper data structures similar to the ones in Section 5.6.1.1. In our implementation, we use the enumeration procedure to identify all violated infeasible paths and only forbid those if they are *minimal infeasible*[11].

One can simply extend the above separation routine to the case of a discrete support $\mathcal{T}_D$. Using proper data structures, checking the time window feasibility along a path takes $\mathcal{O}(D^t)$ time. We emphasize that if travel times are variable, sufficient conditions (5.19) and (5.21) - (5.24) for claiming dominance are no longer certified [12], since they cannot suffice the dominance relationship between P_1 and P_2 for some realization $t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}$. Thus, route enumeration is turned off in this case.

11 An infeasible path $P = (v_1, v_2, ..., v_p)$ is said to be minimal infeasible if the truncated subpaths defined by $A_P \setminus \{(v_1, v_2)\}$ and $A_P \setminus \{(v_{p-1}, v_p)\}$ are feasible [101].

12 Our previous work [160] also pointed it out in a similar situation where path inequalities were dynamically enforced as necessary constraints in the context of solving the VRPTW variant via a BPC algorithm.

5.6.3 *Comparison*

We have discussed two distinct perspectives for adapting the BPC algorithm to solve robust VRPs under demand and travel time uncertainty. The robust pricing perspective guarantees robust feasibility of routing designs in pricing subproblems, while the robust cutting-plane perspective seeks for robust feasibility via dynamically enforcing necessary constraints in master problems. To solve the resulting pricing subproblem in the former, we have synopsized two existing methods from the literature: a directing pricing method and a transformed pricing method. To apply cutting planes dynamically in the latter, we have discussed efficient separation routines. In this section, we compare these two perspectives in the following aspects.

- Uncertainty sources. Both perspectives can deal with robust VRPs with uncertainty in demands and travel times. Note that for both uncertainty sources, ensuring robust feasibility of a routing design can be decomposed into ensuring that of every single route. Stated differently, if every vehicle route in the returned solution is robust feasible, so is the whole routing design. However, for other types of uncertainty sources such as customer order uncertainty we discussed in Section 7.1, one may not be able to encapsulate the robust feasibility condition into pricing subproblems but have to enforce it explicitly in master problems (e.g., see [161]). In such a case, the robust cutting-plane perspective might become the only choice.

- Uncertainty sets. Compared with the robust pricing perspective, the robust cutting-plane perspective is applicable to more general, various types of uncertainty supports. We present in Table 5.2 the applicability of three BPC algorithms to different uncertainty sets. A check mark "✓" indicates that a specific approach is applicable to the corresponding uncertainty set, while a cross mark "✗" denotes the case where this approach is not fit. To the best of our knowledge, our robust cutting-plane perspective can handle all popular classes of uncertainty sets listed in Table 5.2, while the robust pricing perspective has limited applicability. Specifically, the transformed pricing method only works for two types of demand uncertainty sets while the direct pricing method is limited to three types of demand uncertainty sets. The work of [65] has showed that the RSPPRC with an ellipsoidal support $\mathcal{Q}_E$ for demand uncertainty is strongly $\mathcal{NP}$-hard, hence the robust pricing approach is not suitable. As we have pointed out, the direct pricing method will quickly become computationally prohibitive in practice when the budget size Γ^q/Γ^t in $\mathcal{Q}_G/\mathcal{T}_G$ or the number of scenarios D^q/D^t in $\mathcal{Q}_D/\mathcal{T}_D$ increases.

Table 5.2: Applicability of BPC algorithms to different uncertainty sets

Approaches	$\mathcal{Q}$					$\mathcal{T}$	
	$\mathcal{Q}_G$	$\mathcal{Q}_B$	$\mathcal{Q}_F$	$\mathcal{Q}_E$	$\mathcal{Q}_D$	$\mathcal{T}_G$	$\mathcal{T}_D$
Direct Pricing	✓		✓	✗	✓	✓	✓
Transformed Pricing	✓	✓		✗			
Robust Cutting-plane	✓	✓	✓	✓	✓	✓	✓

- Time complexity of pricing subproblems. The robust pricing perspective always has to solve the RSPPRC as pricing subproblems via either a direct or transformed method. The former solves the RSPPRC directly in each pricing iteration, while the latter transformed the RSPPRC into polynomially many deterministic SPPRCs. In either case, the RSPPRC has a larger time complexity than the deterministic SPPRC. In contrast, the robust cutting-plane perspective always solves a deterministic SPPRC of a controlled time complexity. Specifically, one can choose a proper number of demand/travel time scenarios to define the SPPRC so as to generate partially robust feasible routes.

- Tightness of LP relaxations. One can also apply robust RCI and IPEC as strengthening constraints in the robust pricing perspective, then both perspectives are using exactly the same formulation (i.e., the backbone set partitioning model (5.8) - (5.11) + robust RCI + IPEC) except that they are based on two different route sets, $\mathcal{R}$ and $\widetilde{\mathcal{R}}$. Given that $\mathcal{R} \subseteq \widetilde{\mathcal{R}}$, the LP relaxation in the robust pricing perspective is always stronger. This implies that the robust pricing approach might solve a robust VRP more effectively on the condition that the RSPPRC could be solved efficiently.

To summarize, our robust cutting-plane algorithm only entails the solution of deterministic pricing subproblems to generate vehicle routes and relies on the enforcement of necessary constraints in master problems to ensure robust feasibility of routing designs. Compared with the robust pricing algorithm, our proposed approach has the following two distinctive features:

- **Versatility:** it can deal with the parameter variability related to customer demands, vehicle travel times, and even customer orders[13]; furthermore, it works for solving robust VRPs with various types of uncertainty supports in, as shown in Table 5.2.

13 The workf of [161] has demonstrated this in a branch-and-cut framework, hence we can deduce its applicability in the context of BPC.

- **Flexibility:** pricing subproblems that have to be solved are adjustable; in particular, one can enforce a desired number of scenarios in pricing subproblems so as to balance the efforts of generating routes (i.e., pricing step) and separating cutting planes (i.e., cut generation step). This feature becomes crucial in the following case: if the resulting RSPPRC is strongly $\mathcal{NP}$-hard (e.g., $\mathcal{Q}_E$), the robust pricing method will not be practically viable.

5.7 COMPUTATIONAL STUDIES

In this section, we test our proposed BPC algorithm on RCVRP and RVRPTW benchmark instances and compare it against the existing methods from the literature. Our algorithm relies on a deterministic BPC engine to generate vehicle routes and a cut generation routine to enforce necessary constraints. In particular, we utilize VRPSolver 0.3 [135] as our BPC engine via its Julia interface, in which all subordinate linear and mixed-integer linear programs were solved using the IBM ILOG CPLEX Optimizer 12.9.0. The separation routines for IPEC and robust RCI were implemented in C++ and compiled into a C library for use in Julia. The VRPSolver provides a callback function for users to separate and add necessary constraints. The experiments were run on an Intel Xeon E5-2689 v4 server running at 3.10 GHz. The 128 GB of available RAM was shared among 10 copies of the algorithm running in parallel on the server. Each instance was solved by one copy of the algorithm using a single thread. We compare our proposed algorithm against the direct pricing approach from [126] (denoted by `MMVAGM19`) and the transformed pricing approach from [136] (denoted by `PPSV18`) for solving RVRPTW and RCVRP benchmark instances, respectively. The authors of [126] ran their experiments on an Intel Xeon E5-2680 2.70 GHz, while [136] did it on an Intel Core i7-3770 3.40 GHz. According to `https://www.cpubenchmark.net/singleThread.html`, our machine runs as 1.5 times fast as the former and as 2.1 times fast as the latter.

5.7.1 *Computational Results on RCVRP Instances*

In this section, we evaluate our proposed algorithm on solving RCVRP instances under demand uncertainty. We adapt the classic CVRP benchmark instances for generating RCVRP instances in which the customer demands are supported on aforementioned demand uncertainty sets in Section 5.3. We consider five classes of CVRP instances: A, B, E, F, M, and P. These benchmark instances are available at `http://vrp.galgos.inf.puc-rio.`

br/index.php/en/. To be consistent with the literature, when $\mathcal{Q} \in \{\mathcal{Q}_B, \mathcal{Q}_F\}$, we consider 26 instances from class A, 23 instances from B, 11 instances from E, 3 instances respectively from F and M, and 24 instances from P, as [82] did; while $\mathcal{Q} = \mathcal{Q}_G$, we consider one extra instance from class A and two extra instances from E, discarding one instance from P and two large-size instances from F and M, as [136] did. In the case of $\mathcal{Q} \in \{\mathcal{Q}_E, \mathcal{Q}_D\}$, no benchmarks are available in the literature, and we then choose the same set of CVRP benchmarks as $\mathcal{Q} = \mathcal{Q}_F$ to generate RCVRP instances. For each uncertainty set, there are in total 90 benchmarks. The number of customers ranges from 13 to 150. Following the convention in the literature, every entry of the travel cost matrix is calculated from coordinates and then rounded to the nearest integer. The customer demands specified in the benchmark are taken to be their nominal values q^0. For each deterministic CVRP instance, we construct the following five types of uncertainty sets if applicable. As [82] did, we partition the customer set V_c into four geographic quadrants, *NE*, *NW*, *SW*, and *SE*, based on the coordinates in the benchmark instance.

(a) Cardinality-constrained set

$$\mathcal{Q}_G := \left\{ q \in \mathbb{R}^n : q_i = q_i^0 + \alpha q_i^0 \xi_i, \xi_i \in [0,1], \forall i \in V_c, \sum_{i \in V_c} \xi_i \leq \Gamma^q \right\}.$$

This set stipulates that each customer's demand can deviate upward from the nominal value by at most $\alpha \cdot 100\%$, but the total number of customer demands that can simultaneously deviate is bounded from above by Γ^q. As [136] did, we choose $\alpha = 0.3, \Gamma^q = \lfloor 0.75n/K \rfloor$ and modify the vehicle capacity to be $Q = \lfloor 0.3C_{\max} + 0.7C_{\min} \rfloor$, where $C_{\max}$ and $C_{\min}$ values are available in the appendices of [136].

(b) Budget sets (originally proposed in [82])

$$\mathcal{Q}_B := \left\{ q \in \left[(1-\alpha)q^0, (1+\alpha)q^0 \right] : \sum_{i \in \Omega} q_i \leq (1+\alpha\beta) \sum_{i \in \Omega} q_i^0, \forall \Omega \in \{NE, NW, SW, SE\} \right\}.$$

This set stipulates that each customer's demand can deviate from the nominal value by at most $\alpha \cdot 100\%$, but the cumulative demand of each quadrant may not exceed its nominal value by $\alpha\beta \cdot 100\%$. As [82] did, we choose $\alpha = 0.1, \beta = 0.5$ and increase the vehicle capacity by 20%.

(c) Factor models (originally proposed in [82])

$$\mathcal{Q}_F := \left\{ q \in \mathbb{R}^n : q = q^0 + \Psi\xi, \xi \in [-1,1]^4, \left| \sum_{f=1}^{4} \xi_f \right| \leq 4\beta \right\}.$$

This set models the customer demand q_i as a linear combination of 4 factors that can be interpreted as quadrant demands with the weights reflecting the relative proximity of customer i to the quadrant. Specifically, we set $\Psi_{if} = \alpha q_i^0 \psi_{if} / \sum_{f'=1}^{4} \psi_{if'}$, where ψ_{if} denotes the inverse distance between customer i and the centroid of quadrant $f \in \{1,2,3,4\}$. As [82] did, we choose $\alpha = 0.1, \beta = 0.5$ and increase the vehicle capacity by 20%.

Using the reduction procedure described in the proof of Proposition 5.2, one can represent $\overline{\text{Ext}}(\mathcal{Q}_F)$ as a discrete set with only 4 demand scenarios. In particular, each scenario corresponds to the case where $(\xi_1, \xi_2, \xi_3, \xi_4)$ is one permutation of $(1,1,1,-1)$. Since the size of $\overline{\text{Ext}}(\mathcal{Q}_F)$ is small, we choose $\widetilde{\mathcal{Q}} = \overline{\text{Ext}}(\mathcal{Q}_F)$. As a result, routes generated from pricing subproblems will be all robust feasible, i.e., $\widetilde{\mathcal{R}} = \mathcal{R}$.

(d) Ellipsoidal sets (originally proposed in [159])

$$\mathcal{Q}_E := \left\{ q \in \mathbb{R}^n : q = q^0 + \Sigma^{1/2}\xi, \xi^T\xi \leq 1 \right\}.$$

We define $\Sigma = (1-\beta)\Psi\Psi^\top + \beta \text{diag}\left(\alpha q_1^0, \alpha q_2^0, ..., \alpha q_n^0\right)$, where Ψ is the factor loading matrix defined above while $\text{diag}(\cdot)$ denotes a diagonal matrix. When $\beta \in (0,1)$, $\mathcal{Q}_E$ represents a general n-dimensional ellipsoidal set centered at q^0. We choose $\alpha = 0.1, \beta = 0.5$ and increase the vehicle capacity by 10%.

(e) Discrete sets (originally proposed in [159])

$$\mathcal{Q}_D := \text{conv}\left(\{q^0\} \cup \left\{ q^d : d = 1,2,..., \text{nint}(\beta n) \right\} \right).$$

Here, $\text{nint}(\beta n)$ denotes the nearest integer to βn. The points q^d are generated by uniformly sampling $\text{nint}(\beta n)$ points from the n-dimensional hyper-rectangle $\left[(1-\alpha)q^0, (1+\alpha)q^0\right]$. Thus, $\mathcal{Q}_D$ approximates the customer demand vector as independent, uniformly distributed random variables. We choose $\alpha = 0.1, \beta = 0.2$ and increase the vehicle capacity by 10%.

To be consistent with the literature[82, 83, 136], the number of used vehicles is fixed to be the fleet size K when $\mathcal{Q} \in \{\mathcal{Q}_G, \mathcal{Q}_B, \mathcal{Q}_F\}$. That is, we enforce equality for constraint (5.10) in

such cases. In what follows, we present synopsized computational results of our proposed BPC algorithm for addressing RCVRP instances. The detailed results are presented in Section 5.9 of the appendices.

5.7.1.1 $\mathcal{Q}_G$

For each RCVRP instance, the customer demand vector is supported on the cardinality-constrained set $\mathcal{Q}_G$. For a fair comparison, we obtain from [136] the heuristic solution cost to each benchmark and use it as an initial upper bound in our BPC algorithm. In the experiments, $\texttt{NS}^q$ was chosen to be the fleet size K. We impose a time limit of 2 hours for each instance and report in Table 5.3 the consolidated results of our robust cutting-plane algorithm and its comparison against the transformed pricing method from PPSV18. The column "Class" denotes the instance class while the "# inst." denotes the number of instances from this class. We report in columns "# opt.", "Avg. t (sec)" and "Avg. gap (%)" the number of instances that were solved to optimality, the geometric mean solution time (rounded to the nearest integer) for those solved instances, and the average residual gap for those instances for which the algorithm was terminated due to time limit but with valid lower and upper bounds identified, respectively. Out of 90 RCVRP instances, our robust cutting-plane algorithm solved 37 of them to optimality, returning an average residual gap of $2.44\% - 4.81\%$ to the remaining ones; the transformed pricing approach performed significantly better than ours, optimally solving all except three instances. We remark that the superior performance of PPSV18 is expected, since their approach could transform the RSPPRC into a small number (around 20 on average) of deterministic SPPRCs and thus efficiently generate robust feasible routes in pricing subproblems.

Table 5.3: Computational results for BPC algorithms on 90 RCVRP ($\mathcal{Q}_G$) instances

Class	# inst.	PPSV18			This work		
		# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
A	27	27	19	–	11	262	4.79
B	23	20	70	1.65	5	278	4.81
E	13	13	28	–	7	11	4.03
F	2	2	223	–	2	101	–
M	2	2	108	–	0	–	2.44
P	23	23	11	–	12	53	3.05
Total	90	87			37		

5.7.1.2 $\mathcal{Q}_B$

For each RCVRP instance, the customer demand vector is supported on the budget set $\mathcal{Q}_B$. For a fair comparison, we obtain from [136] the heuristic solution cost to each benchmark and use it as an initial upper bound in our BPC algorithm. In our experiments, $\texttt{NS}^q$ is chosen to be 2. We impose a time limit of 2 hours for each instance and report in Table 5.4 the consolidated results of our robust cutting-plane algorithm and its comparison against the transformed pricing method from PPSV18. Out of 90 benchmarks, our robust cutting-plane approach could solve 80 of them to optimality within an average solution time less than 80 seconds, leaving the remaining 10 instances with an average residual gap of $1.48\% - 4.81\%$. The transformed pricing approach performed slightly better, solving 89 instance to optimality. Compared with the branch-and-cut algorithm from [82, 83], our BPC algorithm solved 37 more instances to optimality, which results from the fact that the route-based formulation (e.g., the set partitioning model) has a tighter LP relaxation than the arc-based formulation used in [82, 83].

Table 5.4: Computational results for BPC algorithms on 90 RCVRP ($\mathcal{Q}_B$) instances

Class	# inst.	PPSV18			This work		
		# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
A	26	26	3	–	23	50	1.48
B	23	23	6	–	20	75	2.06
E	11	11	11	–	10	38	0.76
F	3	2	833	0.89	2	36	4.81
M	3	3	154	–	1	32	1.72
P	24	24	1	–	24	14	–
Total	90	89			80		

5.7.1.3 $\mathcal{Q}_F, \mathcal{Q}_E$ *and* $\mathcal{Q}_D$

Now we consider demand uncertainty sets $\mathcal{Q}_F, \mathcal{Q}_E$ and $\mathcal{Q}_D$. When $\mathcal{Q} = \mathcal{Q}_F$, we obtain from [83] the heuristic solution cost to each instance and use it as an initial upper bound in the BPC algorithm; while $\mathcal{Q} \in \{\mathcal{Q}_E, \mathcal{Q}_D\}$, we run the heuristics code from [159] with a time limit of 2 seconds and utilize the returned heuristic solution value as an initial upper bound. For each RCVRP instance, we choose $\texttt{NS}^q = 4, 1, 6$ when $\mathcal{Q} = \mathcal{Q}_F, \mathcal{Q}_E, \mathcal{Q}_D$, respectively. We imposed a time limit of 2 hours for each instance and report the consolidated computational results in Table 5.5.

Among three cases, our robust cutting-plane algorithm performs the best when $\mathcal{Q} = \mathcal{Q}_F$, solving 85 out of 90 instances to optimality. As we mentioned above, every generated route is guaranteed to be robust feasible since we enforce $\widetilde{\mathcal{Q}} = \overline{\text{Ext}}(\mathcal{Q}_F)$ when defining pricing subproblems. As a consequence, the LP relaxation of the set partitioning model is tighter and our algorithm could solve RCVRP instances more efficiently in this case. As expected, the BPC framework outperforms the branch-and-cut implementation from [82, 83], solving 33 more instances to optimality. Our proposed algorithm performs roughly the same on solving RCVRP instances when $\mathcal{Q} = \mathcal{Q}_E$ and $\mathcal{Q} = \mathcal{Q}_D$. The number of solved instances is around 75 and the average residual gap is $1\% - 6\%$.

Table 5.5: Computational results for the BPC algorithm on 90 RCVRP ($\mathcal{Q}_F/\mathcal{Q}_E/\mathcal{Q}_D$) instances

Class	# inst.	$\mathcal{Q}_F$			$\mathcal{Q}_E$			$\mathcal{Q}_D$		
		# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
A	26	26	11	–	23	53	3.23	23	71	3.01
B	23	21	19	1.58	16	46	2.97	15	52	3.70
E	11	11	21	–	9	39	6.22	9	38	3.00
F	3	2	68	1.47	2	18	5.29	2	74	5.16
M	3	1	1,573	1.46	2	1,508	3.44	0	–	3.14
P	24	24	6	–	24	11	–	23	20	1.01
Total	90	85			76			72		

Through the above experiments on RCVRP instances, we make a few remarks: (i) the approach of embedding robust RCI into a deterministic BPC algorithm to address the RCVRP is versatile and efficient for various types of demand uncertainty sets. (ii) the transformed pricing approach demonstrated superior performance than ours for RCVRP instances in which demands are supported on $\mathcal{Q}_G$ and $\mathcal{Q}_B$; (iii) embedding capacity inequalities into a BPC framework yields better performance than the case of branch-and-cut.

5.7.2 *Computational Results on RVRPTW Instances*

We now evaluate our proposed algorithm on solving RVRPTW instances under demand and/or travel time uncertainty. We follow the same procedure from [126] to adapt Solomon instances [156] for generating RVRPTW instances in which the customer demands and vehicle travel times are supported on the cardinality-constrained sets $\mathcal{Q}_G$ and $\mathcal{T}_G$, respec-

tively. The Solomon instances include 100 customers and are classified according to the spatial distribution of customers: classes "C1" and "C2" denote a clustered distribution, "R1" and "R2" denote a random distribution, while "RC1" and "RC2" denote a mixture of both distributions. In addition, instances from C2, R2, and RC2 have wider time windows and larger vehicle capacities than the other classes. The Solomon instances are available at http://neo.lcc.uma.es/vrp/. We apply the convention that travel times and costs are calculated from coordinates and then truncated with one decimal place. The customer demands specified in the benchmark and the calculated travel times are taken to be their nominal values q^0 and t^0, respectively. The maximum allowable deviation in cardinality-constrained sets (5.1) and (5.2) is $\hat{q}_i = \text{trunc}\left(\alpha^q \times q_i^0\right)$ and $\hat{t}_{ij} = 0.1 \times \text{trunc}\left(\alpha^t \times 10 \times t_{ij}^0\right)$, respectively. Thus, each customer's demand can deviate upward from its nominal value by about $\alpha^q \cdot 100\%$ and that each arc traversal time can deviate upward from its nominal value by about $\alpha^t \cdot 100\%$. In our experiments, we choose $\alpha^q = \alpha^t = 0.1, \Gamma^q = \Gamma^t = 5$, as used in [126].

For each RVRPTW instance, we consider three cases: with only demand uncertainty, with only travel time uncertainty, and with both demand and travel time uncertainty. Since we did not have at hand sophisticated heuristics for solving the RVRPTW, we chose to use as the initial upper bound the best known value of each instance modified upwards by an offset of 0.1. Correspondingly, our BPC algorithm always has to locate by itself a feasible solution with a value better than the initial upper bound provided. As [133] pointed out, for VRPTW instances from classes C2, R2, and RC2, capacity constraints are not really binding. Therefore, to ease the solution of the pricing subproblem, we do not include these constraint (i.e., $\widetilde{\mathcal{Q}} = \emptyset$) but enforce them in master problems through capacity inequalities whenever needed. As we have mentioned, route enumeration has to be deactivated whenever travel time uncertainty exists. In what follows, we present synopsized computational results for RVRPTW instances. The detailed results are presented in Section 5.9 of the appendices.

5.7.2.1 $\mathcal{Q}_G$

For each RVRPTW instance, the customer demand vector is supported on cardinality-constrained set $\mathcal{Q}_G$ and the travel times take their nominal values. In our experiments, $\texttt{NS}^q$ was chosen to be 10. We impose a time limit of 1 hour for each RVRPTW instance and report in Table 5.6 the consolidated results of our proposed robust cutting-plane algorithm and its comparison against the direct pricing approach from MMVAGM19[14]. The column names have the same meanings as before, except that the additional column "# no

14 The computational results for MMVAGM19 comes from the appendices of that paper.

LB" denotes the number of instances[15] for which no lower bounds were reported in the appendices of MMVAGM19. Out of 56 instances, our proposed algorithm solved 51 of them to optimality within the time limit of 1 hour while MMVAGM19 solved 34 instances. A noticeable observation is that no valid lower bounds were identified for 7 benchmarks in MMVAGM19. We deem that this was caused by the following: if MMVAGM19 considered to compute the Lagrangian dual value as a valid lower bound, then the reason is that pricing subproblems were never solved exactly in MMVAGM19 for these instances; otherwise, it is owing to the fact that column generation never converged even once before their algorithm terminated. In either case, it implies that solving pricing subproblems exactly via the direct pricing approach is computationally prohibitive for these instances. This confirms our claim in Section 5.6.1.1 that solving the resulting RSPPRC directly has an increased time complexity and might become problematic. Compared with the direct pricing method, our robust cutting-plane algorithm also performs better in terms of both the average solution time and the average residual gap.

Table 5.6: Computational results for BPC algorithms on 90 RVRPTW ($\mathcal{Q}_G$) instances

Class	# inst.	MMVAGM19				This work		
		# opt.	# no LB	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
C1	9	3	0	881	9.94	5	158	2.32
R1	12	9	0	62	2.30	12	18	–
RC1	8	6	0	94	4.67	8	62	–
C2	8	7	1	196	–	8	18	–
R2	11	4	4	598	1.28	10	96	5.36
RC2	8	5	2	283	1.62	8	37	–
Total	56	34	7			51		

5.7.2.2 $\mathcal{T}_G$

For each RVRPTW instance, the travel time vector is supported on cardinality-constrained set $\mathcal{T}_G$ and the customer demands take their nominal values. In our experiments, we choose $\texttt{NS}^t = 1$. We impose a time limit of 1 hour for each instance and report the consolidated results and its comparison against the direct pricing approach from MMVAGM19 in Table 5.7. The direct pricing approach solved 5 more instances than our proposed algorithm within the time limit. When we have a closer look, an interesting observation is that MMVAGM19

15 Hence, these instances are not taken into account when reporting "Avg. gap (%)".

solved more instances from classes C1, R1 and RC1 while our algorithm performed better for instances from classes C2, R2 and RC2. Instances from the former classes have tight time windows and hence there only exists a relatively small number of robust feasible routes in the RSPPRC, such that the direct pricing method turns out to be computationally cheap; instances from the latter classes have wide time windows and hence small perturbations on the travel time vector will not deprive most routes of their feasibility, such that solving the SPPRC defined by $\widetilde{\mathcal{T}}$ rather than the RSPPRC defined by $\mathcal{T}$ to generate routes bears less burden. Another observation is that the direct pricing approach again had difficulty in identifying valid lower bounds for quite a few instances.

Table 5.7: Computational results for BPC algorithms on 56 RVRPTW ($\mathcal{T}_G$) instances

Class	# inst.	MMVAGM19				This work		
		# opt.	# no LB	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
C1	9	8	0	154	2.55	8	67	0.34
R1	12	10	0	156	2.37	3	57	1.29
RC1	8	7	0	148	5.16	0	–	2.04
C2	8	6	1	796	1.29	8	1,093	–
R2	11	4	4	482	2.24	8	165	4.95
RC2	8	5	2	241	1.44	8	134	–
Total	56	40	7			35		

5.7.2.3 $\mathcal{Q}_G \times \mathcal{T}_G$

For each RVRPTW instance, the demand vector and travel time vector are supported on cardinality-constrained set $\mathcal{Q}_G$ and $\mathcal{T}_G$, respectively. We choose $\texttt{NS}^q = 10, \texttt{NS}^t = 1$. We impose a time limit of 1 hour for each instance and report in Table 5.8 the consolidated results of our algorithm and its comparison against the direct pricing approach from MMVAGM19. Both algorithms demonstrate a comparable performance, solving about 30 out of 56 benchmark instances to optimality. Again, our robust cutting-plane algorithm performed better for instances with wide time windows, while the direct pricing method did better for instances with tight time windows. Identifying valid lower bounds for some instances was still challenging for the latter method.

Through the above experiments on RVRPTW instances, we show that our approach of enforcing IPEC as necessary constraints in a BPC framework to address the RVRPTW demonstrates a comparable performance to the direct pricing method from MMVAGM19.

Table 5.8: Computational results for BPC algorithms on 56 RVRPTW ($\mathcal{Q}_G \times \mathcal{T}_G$) instances

Class	# inst.	MMVAGM19				This work		
		# opt.	# no LB	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
C1	9	3	0	1,860	9.73	3	969	2.02
R1	12	10	0	180	2.31	3	74	1.29
RC1	8	5	0	126	4.38	0	–	2.13
C2	8	5	1	850	1.65	8	1,076	–
R2	11	4	4	621	4.39	8	173	6.75
RC2	8	5	2	343	1.44	8	133	–
Total	56	32	7			30		

5.8 CONCLUSIONS

This work is focused on the robust vehicle routing problem under demand/travel time uncertainty. Customer demands and vehicle travel times are assumed to be random variables that can take any values from their respective uncertainty sets. In this chapter, we consider five popular classes of uncertainty sets: cardinality-constrained sets, budget sets, factor models, ellipsoidal sets and discrete sets. One aims to *a priori* identify a set of cost-effective routes for vehicles to traverse such that along these routes capacity constraints and time window conditions are satisfied under any realization of uncertain parameters. We explore BPC algorithms to address this problem. In particular, we synposize the literature approaches as the robust pricing perspective, that is, ensuring *fully robust feasibility* of returned routes when they were generated in pricing subproblems. We propose a novel BPC algorithm that combines the cutting-plane techniques and the deterministic BPC advances. In particular, our proposed approach utilizes a deterministic pricing engine to generate *partially robust feasible* routes and enforces IPEC and robust RCI as necessary constraints to ensure robust feasibility of routing designs. We conduct extensive computational studies on RCVRP and RVRPTW instances with various classes of uncertainty sets. The computational results demonstrate the versatility, flexibility, and efficiency of our robust cutting-plane approach.

5.9 APPENDIX: DETAILED TABLES OF RESULTS

We presents detailed computational results for RCVRP and RVRPTW instances in Table 5.9 - 5.12 and Table 5.13 - 5.14, respectively. As a reference, we also present the computational results for solving deterministic CVRP and VRPTW instances (with the optimal value provided as an upper bound for each instance) via the VRPSolver. In each table, the column "Instance" denotes the instance name; the column "Opt [UB]" then reports the corresponding optimal objective value, while the column "t (sec) [LB]" provides the time to solve the instance to optimality; if an instance could not be solved within the allotted time limit, these columns report (in brackets) the best upper and lower bounds found within this time limit. Instances that were not solved to optimality in the literature but solved in this work for the first time are indicated with an asterisk (*).

Table 5.9: Detailed results for our BPC algorithm on RCVRP instances (A)

Instance	$\{q^0\}$		$\mathcal{Q}_G$		$\mathcal{Q}_B$		$\mathcal{Q}_F$		$\mathcal{Q}_E$		$\mathcal{Q}_D$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
A-n32-k5	784	2	857	119	748	2	748	2	755	4	755	5
A-n33-k5	661	2	675	30	642	13	631	2	652	2	652	14
A-n33-k6	742	2	758	215	717	3	710	2	733	2	730	3
A-n34-k5	778	5	776	4,226	715	3	702	2	747	56	743	252
A-n36-k5	799	3	[823]	[806]	755	3	766	8	783	13	769	4
A-n37-k5	669	2	706	292	650	4	648	3	667	5	665	28
A-n37-k6	949	4	[948]	[905]	892	14	892	6	907	2	907	6
A-n38-k5	730	4	714	22	704	11	693	3	709	3	709	5
A-n39-k5	822	4	[818]	[789]	777	48	772	5	806	32	803	62
A-n39-k6	831	3	850	230	787	2	786	2	813	13	809	14
A-n44-k6	937	2	930	59	909	302	892	2	928	31	919	16
A-n45-k6	944	2	918	278	896	8	891	4	923	11	921	10
A-n45-k7	1,146	5	[1,163]	[1,121]	–	–	–	–	–	–	–	–
A-n46-k7	914	2	988	3,353	888	31	883	7	906	137	902	55
A-n48-k7	1,073	3	[1,129]	[1,067]	1,033	293	1,033*	41	1,060	1,014	1,042	194
A-n53-k7	1,010	5	[1,019]	[983]	974	287	967	65	987	19	984	11
A-n54-k7	1,167	8	[1,169]	[1,110]	1,106	738	1,097*	42	1,145	3,358	1,144	1,228
A-n55-k9	1,073	2	1,107	1,494	1,030	39	1,007	4	1,055	39	1,055	157
A-n60-k9	1,354	10	[1,408]	[1,311]	[1,280]	[1,262]	1,264*	42	1,292	150	1,290	345
A-n61-k9	1,034	7	[1,022]	[984]	983	419	974*	19	1,010	133	1,003	118
A-n62-k8	1,288	16	[1,339]	[1,260]	[1,217]	[1,197]	1,201*	67	1,245	3,128	1,232	6,058
A-n63-k9	1,616	17	[1,620]	[1,501]	1,505	4,754	1,498*	193	1,571	3,343	[1,575]	[1,533]
A-n63-k10	1,314	10	[1,348]	[1,281]	1,233	586	1,222*	19	1,257	482	1,249	1,611
A-n64-k9	1,401	18	[1,417]	[1,338]	1,325	1,017	1,314*	97	[1,385]	[1,343]	[1,380]	[1,335]
A-n65-k9	1,174	6	[1,184]	[1,120]	1,106	188	1,094	5	1,164	1,582	1,144	3,572
A-n69-k9	1,159	10	[1,177]	[1,136]	1,109	953	1,096*	18	[1,149]	[1,114]	1,122	5,295
A-n80-k10	1,763	16	[1,803]	[1,702]	[1,662]	[1,639]	1,644*	1,253	[1,743]	[1,680]	[1,715]	[1,662]
# opt.	27		11		23		26		23		23	

Table 5.10: Detailed results for our BPC algorithm on RCVRP instances (B)

Instance	$\{q^0\}$		$\mathcal{Q}_G$		$\mathcal{Q}_B$		$\mathcal{Q}_F$		$\mathcal{Q}_E$		$\mathcal{Q}_D$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
B-n31-k5	672	3	694	5,842	651	2	651	2	604	2	602	4
B-n34-k5	788	6	789	1,565	768	1,095	748	10	782	264	769	151
B-n35-k5	955	5	[986]	[952]	883	4	883	3	[946]	[921]	921	254
B-n38-k6	805	4	823	33	729	4	729	3	705	69	688	3
B-n39-k5	549	4	561	76	532	23	529	17	534	9	534	42
B-n41-k6	829	3	[838]	[797]	796	456	791	26	800	4	801	10
B-n43-k6	742	8	[779]	[734]	681	167	680	34	683	2	683	22
B-n44-k7	909	5	[943]	[912]	835	10	835	10	856	9	855	95
B-n45-k5	751	7	[739]	[717]	701	421	680	13	708	16	702	37
B-n45-k6	678	6	[668]	[631]	660	11	657	9	670	25	668	2,271
B-n50-k7	741	10	758	74	679	3	699	8	732	1,021	717	88
B-n50-k8	1,312	17	[1,330]	[1,286]	[1,224]	[1,208]	[1,217]	[1,202]	[1,251]	[1,215]	[1,226]	[1,206]
B-n51-k7	1,032	15	[1,027]	[944]	961	238	928	6	[999]	[988]	[996]	[973]
B-n52-k7	747	4	[775]	[750]	675	30	670	14	667	11	662	15
B-n56-k7	707	4	[740]	[730]	623	8	623	27	614	8	612	11
B-n57-k7	1,153	12	[1,132]	[1,054]	1,055	778	1,052*	62	[1,136]	[1,094]	[1,126]	[1,071]
B-n57-k9	1,598	5	[1,656]	[1,594]	1,540	22	1,539	22	[1,520]	[1,488]	[1,490]	[1,474]
B-n63-k10	1,496	8	[1,588]	[1,471]	1,407	6,809	1,405*	392	[1,507]	[1,459]	[1,497]	[1,431]
B-n64-k9	861	6	[865]	[839]	803	66	803	30	805	19	804	41
B-n66-k9	1,316	21	[1,319]	[1,209]	[1,251]	[1,217]	1,210	33	1,269	2,571	[1,266]	[1,215]
B-n67-k10	1,032	11	[1,086]	[1,048]	1,007	7,030	1,001*	248	1,012	1,129	1,001	2,658
B-n68-k9	1,272	40	[1,298]	[1,247]	[1,205]	[1,179]	[1,197]	[1,174]	[1,255]	[1,190]	[1,245]	[1,167]
B-n78-k10	1,221	10	[1,261]	[1,161]	1,131	1,697	1,130*	81	1,162	6,256	[1,200]	[1,140]
# opt.	23		5		20		21		16		15	

Table 5.11: Detailed results for our BPC algorithm on RCVRP instances (E, F and M)

Instance	$\{q^0\}$		$\mathcal{Q}_G$		$\mathcal{Q}_B$		$\mathcal{Q}_F$		$\mathcal{Q}_E$		$\mathcal{Q}_D$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
E-n13-k4	247	1	277	2	–	–	–	–	–	–	–	–
E-n22-k4	375	1	373	1	373	2	373	1	373	1	373	2
E-n23-k3	569	2	570	2	563	2	544	2	564	2	569	2
E-n30-k3	534	10	495	6	475	3	492	7	495	3	495	4
E-n31-k7	379	2	379	11	–	–	–	–	–	–	–	–
E-n33-k4	835	2	836	343	814	125	814	42	828	56	821	9
E-n51-k5	521	2	519	101	516	28	516	24	519	5	518	27
E-n76-k7	682	27	[699]	[681]	661	90	661*	283	665	294	661	76
E-n76-k8	735	19	[736]	[715]	709	314	700*	26	721	511	714	199
E-n76-k10	830	15	[830]	[796]	796	1,580	782*	10	809	202	807	410
E-n76-k14	1,021	7	[1,022]	[963]	952	41	952*	29	987	3,198	982	7,082
E-n101-k8	815	61	[826]	[802]	[789]	[783]	783*	139	[858]	[790]	[808]	[784]
E-n101-k14	1,067	31	[1,121]	[1,054]	1,011	86	1,009*	77	[1,085]	[1,036]	[1,057]	[1,025]
F-n45-k4	724	10	736	65	718	37	714	121	721	9	721	71
F-n72-k4	237	24	236	156	232	35	232	38	235	35	234	77
F-n135-k7	1,162	3,058	–	–	[1,122]	[1,068]	[1,086]	[1,070]	[1,171]	[1,109]	[1,162]	[1,102]
M-n101-k10	820	4	[918]	[907]	809	32	804	1,573	811	585	[827]	[795]
M-n121-k7	1,034	44	[1,030]	[992]	[994]	[980]	[987]	[980]	1,004	3,892	[999]	[984]
M-n151-k12	1,015	119	–	–	[987]	[967]	[991]	[969]	[1,018]	[983]	[1,010]	[969]
# opt.	19		9		13		14		13		11	

Table 5.12: Detailed results for our BPC algorithm on RCVRP instances (P)

Instance	$\{q^0\}$		$\mathcal{Q}_G$		$\mathcal{Q}_B$		$\mathcal{Q}_F$		$\mathcal{Q}_E$		$\mathcal{Q}_D$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
P-n16-k8	450	1	–	–	439	1	439	1	450	2	448	2
P-n19-k2	212	2	195	1	195	2	195	1	195	1	195	1
P-n20-k2	216	2	208	2	208	1	208	1	209	1	209	1
P-n21-k2	211	1	208	2	208	2	208	1	211	2	211	1
P-n22-k2	216	2	213	2	213	2	213	2	216	2	215	2
P-n22-k8	603	1	601	2	537	1	557	1	592	2	587	2
P-n23-k8	529	1	527	4	504	2	503*	1	524	2	524	2
P-n40-k5	458	2	468	14	447	3	447	2	456	3	455	13
P-n45-k5	510	3	512	38	501	41	494	3	503	3	503	9
P-n50-k7	554	3	563	258	539	9	537*	4	546	40	541	114
P-n50-k8	631	4	[614]	[591]	592	6	588*	6	605	5	605	13
P-n50-k10	696	2	695	1,093	656	2	656*	2	676	10	670	7
P-n51-k10	741	2	736	154	707	29	698*	2	722	3	721	10
P-n55-k7	568	4	583	2,083	549	51	544	14	550	89	542	11
P-n55-k8	588	9	[624]	[604]	572	126	568*	4	570	24	570	682
P-n55-k10	694	3	[718]	[691]	670	185	657*	8	669	8	669	51
P-n55-k15	989	2	[945]	[908]	889	4	877*	2	930	11	923	5
P-n60-k10	744	2	[755]	[727]	712	23	705*	6	726	26	726	246
P-n60-k15	968	2	[1,020]	[968]	931	11	916*	3	950	9	949	50
P-n65-k10	792	4	[809]	[780]	765	105	761*	13	781	1,146	766	111
P-n70-k10	827	10	[824]	[794]	785	20	783*	6	809	387	801	1,038
P-n76-k4	593	28	[590]	[585]	590	384	590	321	590	38	[595]	[589]
P-n76-k5	627	33	[621]	[612]	616	970	615*	1,462	621	50	616	219
P-n101-k4	681	144	[681]	[677]	673	852	673	6,347	673	77	673	3,932
# opt.	24		12		24		24		24		23	

Table 5.13: Detailed results for our BPC algorithm on RVRPTW instances (C1, R1 and RC1)

Instance	$\{q^0\} \times \{t^0\}$		$\mathcal{Q}_G \times \{t^0\}$		$\{q^0\} \times \mathcal{T}_G$		$\mathcal{Q}_G \times \mathcal{T}_G$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
C101	827.3	2	981.3	15	848.0	5	994.5	366
C102	827.3	4	975.2*	280	846.2	207	976.6*	3,248
C103	826.3	9	[972.6]	[950.8]	838.8	161	[974.5]	[952.1]
C104	822.9	13	[953.3]	[934.4]	[834.0]	[831.2]	[961.2]	[933.7]
C105	827.3	3	981.3	218	848.0	27	[984.2]	[977.7]
C106	827.3	3	981.3	45	848.0	132	984.1	765
C107	827.3	3	981.3*	2,423	842.8	18	[982.9]	[969.7]
C108	827.3	4	[974.7]	[948.9]	842.8	136	[973.9]	[949.0]
C109	827.3	5	[966.1]	[943.0]	842.8	315	[966.1]	[943.0]
R101	1,637.7	2	1,637.7	3	1,692.1	9	1,692.1	19
R102	1,466.6	2	1,466.6	3	1,505.3	17	1,505.3	16
R103	1,208.7	4	1,208.7	6	1,235.3	1,230	1,235.3	1,363
R104	971.5	27	975.8*	165	[999.5]	[981.6]	[999.5]	[982.9]
R105	1,355.3	3	1,355.3	4	[1,391.7]	[1,368.9]	[1,391.7]	[1,365.9]
R106	1,234.6	5	1,234.6	7	[1,265.5]	[1,244.0]	[1,265.5]	[1,244.0]
R107	1,064.6	13	1,064.6	18	[1,081.3]	[1,074.4]	[1,082.3]	[1,071.4]
R108	932.1	41	938.6*	124	[951.8]	[934.9]	[953.7]	[938.0]
R109	1,146.9	13	1,147.2	19	[1,169.2]	[1,161.9]	[1,169.4]	[1,166.0]
R110	1,068.0	12	1,068.0	15	[1,097.0]	[1,076.8]	[1,097.0]	[1,076.7]
R111	1,048.7	33	1,048.7	44	[1,071.0]	[1,065.8]	[1,070.9]	[1,064.5]
R112	948.6	62	950.9*	164	[961.4]	[950.3]	[961.3]	[951.5]
RC101	1,619.8	4	1,619.8	6	[1,674.8]	[1,651.4]	[1,677.5]	[1,651.1]
RC102	1,457.4	20	1,472.7	32	[1,500.9]	[1,476.2]	[1,512.6]	[1,493.5]
RC103	1,258.0	22	1,264.6	32	[1,331.0]	[1,265.7]	[1,341.3]	[1,273.5]
RC104	1,132.3	48	1,156.7*	2,371	[1,154.4]	[1,138.6]	[1,179.7]	[1,155.6]
RC105	1,513.7	16	1,513.7	16	[1,563.0]	[1,545.2]	[1,565.4]	[1,546.2]
RC106	1,372.7	35	1,388.8	69	[1,400.4]	[1,377.8]	[1,413.0]	[1,391.7]
RC107	1,207.8	14	1,236.7	63	[1,244.0]	[1,214.8]	[1,279.7]	[1,242.4]
RC108	1,114.2	51	1,155.0*	228	[1,141.4]	[1,119.8]	[1,161.2]	[1,143.9]
# opt.	29		25		11		6	

Table 5.14: Detailed results for our BPC algorithm on RVRPTW instances (C2, R2 and RC2)

Instance	$\{q^0\} \times \{t^0\}$		$\mathcal{Q}_G \times \{t^0\}$		$\{q^0\} \times \mathcal{T}_G$		$\mathcal{Q}_G \times \mathcal{T}_G$	
	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]	Opt [UB]	t (sec) [LB]
C201	589.1	9	589.1	10	605.4	488	605.4	557
C202	589.1	15	589.1	20	605.2	1,990	605.2*	1,832
C203	588.7	35	588.7	24	597.7*	1,130	597.7*	1,069
C204	588.1	40	588.1*	43	594.0*	536	594.0*	510
C205	586.4	14	586.4	12	598.9	965	598.9	910
C206	586.0	14	586.0	16	598.9	1,414	598.9	1,688
C207	585.8	20	585.8	17	598.3	1,074	598.3	1,097
C208	585.8	14	585.8	15	598.3	2,369	598.3	1,923
R201	1,143.2	11	1,143.2	12	1,143.2	11	1,143.2	12
R202	1,029.6	80	1,029.6	93	1,032.6	174	1,032.6	204
R203	870.8	87	870.8	70	873.3	113	873.3	120
R204	731.3	195	731.3*	180	731.3*	203	731.3*	219
R205	949.8	87	949.8	86	949.8	406	949.8	365
R206	875.9	137	875.9*	149	875.9*	137	875.9*	149
R207	794.0	96	794.0*	114	794.1*	149	794.1*	151
R208	701.0	3,296	[737.4]	[697.9]	[741.7]	[698.4]	[772.2]	[698.3]
R209	854.8	129	854.8*	134	858.4*	1,460	858.4*	1,513
R210	900.5	168	900.5*	159	[931.7]	[907.6]	[913.2]	[906.1]
R211	746.7	146	746.7*	149	[799.1]	[747.8]	[830.2]	[747.9]
RC201	1,261.8	8	1,261.8	10	1,263.0	21	1,263.0	21
RC202	1,092.3	21	1,092.3	15	1,095.6	37	1,095.6	29
RC203	923.7	28	923.7	28	933.0	387	933.0	524
RC204	783.5	66	783.5*	67	787.5*	520	787.5*	491
RC205	1,154.0	17	1,154.0	15	1,154.0	36	1,154.0	33
RC206	1,051.1	55	1,051.1	53	1,051.1	92	1,051.1	95
RC207	962.9	83	962.9*	112	963.3*	375	963.3*	341
RC208	776.1	80	776.1*	140	778.4*	558	778.4*	580
# opt.	27		26		24		24	

6

A CUSTOMIZED BRANCH-AND-BOUND APPROACH FOR IRREGULAR SHAPE NESTING

In this chapter, we study the Nesting Problem, which aims to determine a configuration of a set of irregular shapes within a rectangular sheet of material of fixed width, such that no overlap among the shapes exists, and such that the length of the sheet is minimized. When both translation and rotation of the shapes are allowed, the problem can be formulated as a nonconvex quadratically constrained programming model that approximates each shape by a set of inscribed circles and enforces that circle pairs stemming from different shapes do not overlap. However, despite many recent advances in today's global optimization solvers, solving this nonconvex model to guaranteed optimality remains extremely challenging even for the state-of-the-art codes. In this chapter, we propose a customized branch-and-bound approach to address the Nesting Problem to guaranteed optimality. Our approach utilizes a novel branching scheme to deal with the reverse convex quadratic constraints in the quadratic model and incorporates a number of problem-specific algorithmic tweaks. Our computational studies on a suite of 64 benchmark instances demonstrate the customized algorithm's effectiveness and competitiveness over the use of general-purpose global optimization solvers, including for the first time the ability to find global optimal nestings featuring five polygons under free rotation.

6.1 INTRODUCTION

"Nesting" refers to the task of finding a packing of two-dimensional shapes that minimizes the amount of material needed to carve them out of a stock. This minimization of waste can be of utmost economic importance in certain large industrial sectors, wherein a minuscule reduction in the amount of stock material used would result in significant monetary savings across the whole industry. For example, obtaining tight nesting solutions is of great practical importance when cutting metal parts for automobiles, airframes and other

machinery, as well as cutting leather and fabrics for apparel and upholstery applications. In commercial settings, a human "nester" usually refines a packing solution by hand from the starting point of a computationally-identified solution [123], thereby necessitating the development of automated solutions.

This chapter focuses on the *Irregular Strip Packing Problem*, also simply known as the *Nesting Problem*, which is is a rather general two-dimensional cutting and packing problem [173] where the shapes to be packed can be different to each other, irregular and nonconvex, and may possibly contain holes. The shapes, which are usually represented (approximated to arbitrary precision) by polygons, are to be packed in a stock that comes in the form of a fixed-width rectangular sheet, whose length is to be minimized. This problem has been studied extensively, with most of the focus being on the development of heuristics to obtain large-scale packings [10, 41, 44, 94]. However, while heuristic methods are practically valuable for the generation of good solutions, they can not rigorously quantify the quality of packings produced and therefore lack any guarantees of optimality. Additionally, heuristic methods often rely on the exact solution of smaller-scale problems as part of their search strategy, motivating the need to develop good exact solution strategies.

The key differences among exact, mathematical optimization-based approaches are the means by which the "non-overlapping" of shapes is satisfied. A traditional method for constraining shapes to not overlap involves constructing a "no-fit" polygon per each pair of shapes in the problem [35]. The no-fit polygon approach simplifies the task of algebraically encoding overlap between two shapes into the relatively easy task of identifying if a point lies in the interior of a polygon. In this way, one can enforce constraints that require a predefined point on a shape to lie outside the no-fit polygons of each of the other shapes. These no-fit polygons can be precomputed inputs to the optimization model, but are only applicable in the case of polygons with fixed orientation (i.e., no rotation allowed). Furthermore, many approaches leveraging the no-fit polygon concept also require the shapes to be convex. In [60], the convexity of the polygons was exploited to formulate a linear programming (LP) model for solving the fixed-orientation version of the Nesting Problem. The algorithm identifies valid constraints that require the horizontal and vertical placement of polygons to shift so as to eliminate an identified overlap.

When nonconvex polygons are considered (still with no rotation), the no-fit polygon may also become nonconvex and the non-overlapping constraints may no longer be expressed in a simple linear form. An approach to handle the nonconvexity of the no-fit polygon was presented in [81], where the authors dynamically select which edges of the nonconvex

no-fit polygon to enforce at a local configuration; however, the authors couple this idea with heuristic methods to guide the placement of shapes, resulting in an inexact, metaheuristic algorithm. [9] build upon this approach by providing a rigorous branch-and-bound (BB) procedure to dynamically search over the sets of no-fit polygon edges. While the authors demonstrated provably optimal packing solutions for up to 12 polygons (5 of which nonconvex), the tractability of their algorithm was found to decrease dramatically with the number of nonconvex polygons considered in the placement.

In [74], the authors adapt the prior approaches by utilizing binary variables to explicitly model the disjunctions between nonconvex parts of the no-fit polygon. This idea results in model formulations that are directly representable as mixed-integer linear programs. At the time, the authors were only able to solve instances with up to 7 pieces (5 of which nonconvex). Consequently, they focused on utilizing their mathematical optimization formulation in the context of a heuristic algorithm for solving the related *Multiple Containment Problem*. [49] later improved upon this approach by representing pieces as combinations of convex polygons and formulating similar mixed-integer programming models.

When considering instances with the freedom to rotate the polygons to be nested, an additional level of nonlinearity is introduced and a geometric idea other than the no-fit polygon must be considered. In [123], the authors adapted their previous translation-only approach to work for continuous rotation by using a BB tree to search through the space of feasible rotations and by repeatedly solving relaxed translation-only problems. However, their algorithm was only practical for packings of 2 or 3 polygons unless further restrictions on angles of rotation were applied. Another idea is to approximate each polygon by a collection of circles, which can then be constrained to not overlap with sets of circles from other polygons. The key feature of such "circle-covering" approaches is the fact that each of the constraints to enforce no overlap between two circles can be readily written with rotation of the parent polygons taken into account. The computational downside, however, is that these non-overlapping constraints are reverse convex quadratic [91] and therefore require use of advanced global optimization (GO) techniques. [144] compare several approaches for statically approximating the polygons via different circle-covering schemes. However, since their schemes over-approximate the true region of the polygons, this method constitutes an inexact, heuristic approach to the Nesting Problem that provides good, feasible packings but no rigorous bound on the best possible one. Furthermore, the authors observe that the tractability of the problem decreases dramatically as more accurate approximations are required. The current state-of-the-art framework for exact polygon nesting under free rotation comes from [100], in an algorithm called *QP-Nest*. This

approach uses inscribed circles to under-approximate the true region of the polygons and attempts to dynamically improve the quality of the approximation by introducing new circles where needed, managing also in this way the trade-off between approximation accuracy and numerical tractability. The *QP-Nest* algorithm was able to globally solve a Nesting Problem instance with four polygons by employing general-purpose GO solvers. A more detailed description of this algorithm is provided later in this chapter.

This work also addresses the Nesting Problem with both nonconvex shapes and free rotation but proposes a tailored algorithm that takes advantage of the specific structure of the circle-covering approach. More specifically, the contributions of this work can be summarized as follows:

- We develop an exact approach to solve the Nesting Problem to global optimality that does not rely on the use of general-purpose global optimization solvers.
- We identify a generic approach to dynamically satisfy reverse convex quadratic constraints commonly found in optimization models within the field of cutting and packing.
- We conduct a comprehensive computational study that illustrates the competitiveness of our approach compared to the previous state-of-the-art.
- We present, for the first time in the open literature, provably optimal solutions for Nesting Problem instances featuring five polygons under free rotation.

The final point above, i.e., the fact that no proven optimal solutions existed to-date for nesting more than four polygons with arbitrary rotation, alludes to the immense difficulty of solving problems of this nature to global optimality. It should be however highlighted that, while exact optimization methods may not be practical for solving industrially-relevant instances (potentially containing an excess of one hundred pieces) at present, the global solution of small instances is a capability utilized by many heuristic methods. To that end, a marginal improvement in the tractability (specifically, the number of polygons that can be accommodated) in an exact approach could provide sizable benefits in the ability of heuristic methods to explore better local solutions when constructing large-scale nestings.

The remainder of this chapter is organized as follows. In Section 6.2, we present the standard quadratically constrained programming (QCP) formulation that models the Nesting Problem, and we discuss how a suitable linear relaxation of the latter can be used as the basis of our BB-based algorithm. Implementation details of the algorithm

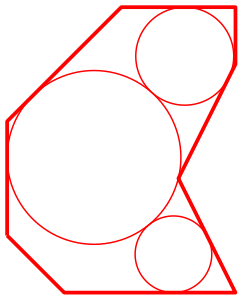


Figure 6.1: Approximation of a polygon's area via a set of inscribed circles.

itself, including several model tightening procedures, are presented in Section 6.3. Finally, Section 6.4 presents results on the algorithm's computational performance as well as a comprehensive comparison with the *QP-Nest* approach utilizing four different state-of-the-art GO solvers. For convenience, a table synopsizing notation is provided at the end of the manuscript.

6.2 MATHEMATICAL MODELING

Let $P = \{1, 2, 3...\}$ denote a set of polygons. For each polygon $p \in P$, let I_p denote the set of vertices, and let (x_{pi}, y_{pi}) denote the nominal coordinates of each vertex $i \in I_p$. Since we allow for arbitrary rotation and translation, let θ_p, h_p, and v_p denote the rotational angle, horizontal translation, and vertical translation for polygon $p \in P$, respectively. Here, the rotational angle is defined with respect to point $(0, 0)$, the origin of the coordinate system. We also use c_p and s_p to represent the cosine and sine of θ_p, respectively. Clearly, after being rotated by θ_p and translated by h_p and v_p, a vertex (x_{pi}, y_{pi}) will lie at new coordinates $(c_p x_{pi} - s_p y_{pi} + h_p, s_p x_{pi} + c_p y_{pi} + v_p)$. The rectangular sheet has a fixed width of W and the objective is to minimize its length L such that all polygons can be placed within it in a way that no two polygons overlap.

6.2.1 *Nonconvex QCP Model*

Every polygon $p \in P$ is approximated by a set of inscribed circles, which we denote as C_p (Fig. 6.1). For each circle $m \in C_p$, let (x_{pm}, y_{pm}) denote the coordinates of its center, and let R_{pm} denote its radius. Given such circle sets for each polygon $p \in P$, the following nonconvex QCP formulation can be defined.

$$\min_{c_p, s_p, h_p, v_p, L} \quad L \tag{6.1}$$

$$\text{s.t.} \quad c_p^2 + s_p^2 = 1 \qquad \forall p \in P \tag{6.2}$$

$$0 \leq c_p x_{pi} - s_p y_{pi} + h_p \leq L \qquad \forall i \in I_p, \forall p \in P \tag{6.3}$$

$$0 \leq s_p x_{pi} + c_p y_{pi} + v_p \leq W \qquad \forall i \in I_p, \forall p \in P \tag{6.4}$$

$$\begin{aligned} &\left[\left(c_p x_{pm} - s_p y_{pm} + h_p\right) - \left(c_q x_{qn} - s_q y_{qn} + h_q\right)\right]^2 + \\ &\left[\left(s_p x_{pm} + c_p y_{pm} + v_p\right) - \left(s_q x_{qn} + c_q y_{qn} + v_q\right)\right]^2 \geq \\ &\left(R_{pm} + R_{qn}\right)^2 \end{aligned} \qquad \begin{aligned} &\forall (m, n) \in C_p \times C_q, \\ &\forall (p, q) \in \{P \times P : q > p\} \end{aligned} \tag{6.5}$$

$$-1 \leq c_p, s_p \leq 1 \qquad \forall p \in P \tag{6.6}$$

In the above formulation, the objective function (6.1) aims to minimize the length of the sheet. The rotational angle is implicitly encoded by its cosine and sine values, which are suitably defined through the trigonometric constraints (6.2). Constraints (6.3) and (6.4) ensure that all vertices of the polygon (and hence, the totality of the polygons) will lie within the sheet.[1] Constraints (6.5) guarantee that there is no overlap between a circle $m \in C_p$ and a circle $n \in C_q$ by imposing that their centers are sufficiently far in terms of Euclidean distance. Finally, constraints (6.6) define the applicable bounds for variables c_p and s_p. Note that the nonconvexity of this model stems from constraints (6.2) and (6.5).

We highlight that, although constraints (6.5) dictate that any circle inscribed in polygon p will not overlap with any circle inscribed in polygon q, the model does not prohibit the penetration of uncovered parts of polygon p into q. Therefore, this model technically constitutes a relaxation of the full Nesting Problem, and its optimal value is merely a lower bound for the latter. The approximation error can however be improved by considering additional circles in the sets C_p that increase the coverage areas of the polygons.

Based on this principle, [100] proposed the *QP-Nest* approach to address the full problem. The circle sets $\{C_p\}_{p \in P}$ are initialized by inscribing N^{init} circles per polygon via a *circle-covering procedure*. After solving the resulting QCP (6.1)–(6.6) to global optimality, a *penetration-computing procedure* is applied to check whether there exists any penetration of one polygon into another. If some degree of overlap is identified between two polygons p

1 In fact, one needs only impose these constraints for the vertices that constitute the convex hull of polygon $p \in P$.

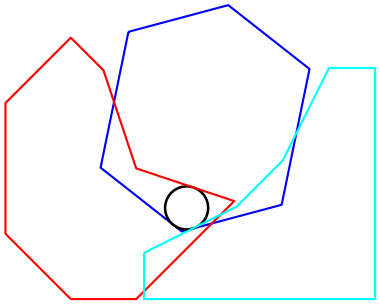


Figure 6.2: Identifying the largest circle within the area where two polygons overlap.

and q, a suitable circle within the overlap area will be generated via a *circle-adding procedure* and appended to the circle sets C_p and C_q (Fig. 6.2). The QCP model is then resolved and the process is iterated until no significant (above tolerance) penetration is found. At that point, the problem is considered to have been solved to global optimality. The overall *QP-Nest* algorithm is illustrated in Fig. 6.3.

6.2.2 *Customized Model Relaxation*

Although the *QP-Nest* approach is straightforward to implement, the resulting iterations of the QCP model quickly grow to the point where they become very hard to solve by even the most advanced of GO solvers. To that end, we develop in this chapter a customized BB-based solver that can develop tighter approximations using smaller circle libraries and that can search through the space of feasible solutions more effectively. Unlike a traditional, general-purpose solver, which must generically handle each nonconvex constraint by relaxing it individually into its convex relaxation and tighten that relaxation via branching on the domains of the problem's native variables, our customized approach recognizes the physical meaning of these constraints and proposes branching on domains of variables that are not explicitly defined in the model. Before we describe the algorithm in detail, we present a suitably parameterized LP relaxation of the QCP model and the main principles for tightening that relaxation.

First, we illustrate our strategy for relaxing the trigonometric constraints (6.2). Let $\left[\theta_p^L, \theta_p^U\right]$ denote the feasible interval of θ_p (originally equal to $[0, 2\pi)$). In the space of (c_p, s_p), the corresponding constraint (6.2) requires that all feasible points project exactly on the circumference of a unit circle (Fig. 6.4a). This constraint can be initially dropped from consideration, resulting in a convex relaxation defined by the box bounds $[-1, +1]$ (Fig. 6.4b). Feasibility of the nonconvex constraint can then be gradually enforced

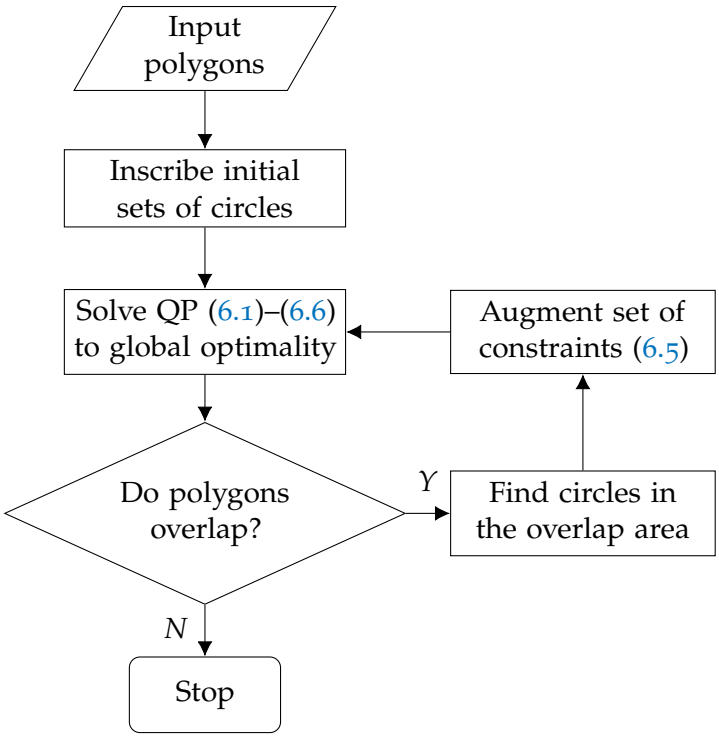


Figure 6.3: The *QP-Nest* approach.

by branching on the implicit θ_p variable domain and tightening the relaxation in the (c_p, s_p) space, as follows.

Let K denote a solution of the relaxed problem with coordinates $\left(c_p^*, s_p^*\right)$ such that it violates the original constraint. Using this solution as a guide, one may dissect the $[0, 2\pi)$ interval into three parts (each representing an angle of $2\pi/3$ radians) and define a set of three linear constraints (corresponding to two tangent lines and one secant) to form the convex relaxation of the original trigonometric constraint in each of the three resulting subdomains. If properly oriented with respect to K (Fig. 6.4c), it can be guaranteed that this parent solution is excluded in all of the three subdomains. In subsequent violations of the trigonometric constraint, one could further dissect the applicable θ_p interval into two parts. If the solution K is used as the guide to define the branching point, one can again guarantee that this parent solution is excluded from both resulting subdomains, defining a proper branching strategy to be employed in the course of a BB search process. More specifically, we propose a domain split that aims to produce a BB tree that is as balanced as possible. Let N be the point with coordinates $\left(\cos\left(\frac{\theta_p^L+\theta_p^U}{2}+\pi\right), \sin\left(\frac{\theta_p^L+\theta_p^U}{2}+\pi\right)\right)$, and let θ_p^* denote the angle in radians between the positive c_p-axis and the point at which the unit circle intersects with the ray that starts at N and passes through K. The resulting

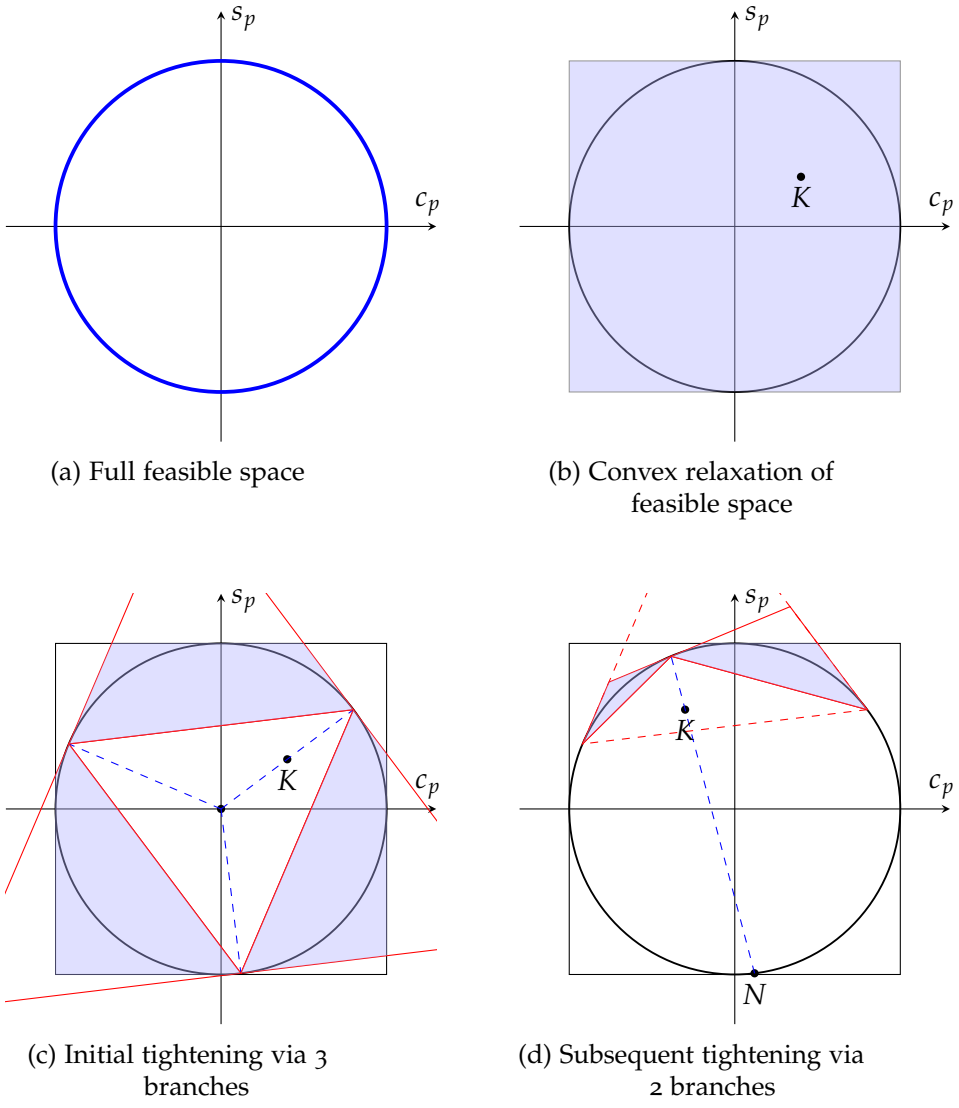
proposed split is then $\left[\theta_p^L, \theta_p^*\right] \cup \left[\theta_p^*, \theta_p^U\right]$ (Fig. 6.4d). Note that our branching scheme works also in the case when K resides outside the trigonometric circle.

With regards to relaxing the non-overlapping constraints (6.5), we follow a similar feasible space dissection strategy. Let us first define for convenience auxiliary variables $\Delta x_{pmqn} := \left(c_p x_{pm} - s_p y_{pm} + h_p\right) - \left(c_q x_{qn} - s_q y_{qn} + h_q\right)$ and $\Delta y_{pmqn} := \left(s_p x_{pm} + c_p y_{pm} + v_p\right) - \left(s_q x_{qn} + c_q y_{qn} + v_q\right)$ for each circle pair for which we have decided to enforce the corresponding non-overlapping constraint. The latter can now be written as:

$$\Delta x_{pmqn}^2 + \Delta y_{pmqn}^2 \geq \left(R_{pm} + R_{qn}\right)^2$$

Let CC_{pmqn} denote the circle that is centered at the origin, $(0,0)$, and has a radius of $\left(R_{pm} + R_{qn}\right)$. Furthermore, let ϕ_{pmqn} be the angle between the positive Δx_{pmqn}-axis and the point given by the coordinates $\left(\Delta x_{pmqn}, \Delta y_{pmqn}\right)$, and let $[\phi_{pmqn}^L, \phi_{pmqn}^U]$ represent its feasible interval (originally equal to $[0, 2\pi)$). In the $\left(\Delta x_{pmqn}, \Delta y_{pmqn}\right)$ space, the non-overlapping constraint (6.5) requires that all feasible points project either outside or exactly on the circumference of circle CC_{pmqn} (Fig. 7.1a). Dropping at first the constraint from consideration results in a convex relaxation that is defined by the full space (Fig. 7.1b). Feasibility of the nonconvex constraint can be then be gradually enforced by branching on the implicit ϕ_{pmqn} variable domain and tightening the relaxation in the $\left(\Delta x_{pmqn}, \Delta y_{pmqn}\right)$ space, as follows.

Let K denote a solution of the relaxed problem with coordinates $\left(\Delta x_{pmqn}^*, \Delta y_{pmqn}^*\right)$ such that it violates the original constraint. Using this solution as a guide, one may dissect the $[0, 2\pi)$ interval into three parts (each representing an arc of angle $2\pi/3$ radians) and define a set of three linear constraints (corresponding to one secant and two boundary lines) to form the convex relaxation of the original non-overlapping constraint in each of the three resulting subdomains. If properly oriented with respect to K (Fig. 7.1c), it can be guaranteed that this parent solution is excluded in all of the three subdomains. In subsequent violations of the same constraint, one could further dissect the applicable ϕ_{pmqn} interval into two parts. If the solution K is used as the guide to define the branching point, one can again guarantee that this parent solution is excluded from both resulting subdomains, defining a proper branching strategy to be employed in the course of a BB search process. More specifically, let N be the point with coordinates $\left(\left(R_{pm} + R_{qn}\right)\cos\left(\frac{\phi_{pmqn}^L + \phi_{pmqn}^U}{2} + \pi\right), \left(R_{pm} + R_{qn}\right)\sin\left(\frac{\phi_{pmqn}^L + \phi_{pmqn}^U}{2} + \pi\right)\right)$, and let ϕ_{pmqn}^* denote the angle in radians between the positive Δx_{pmqn}-axis and the point at which



(a) Full feasible space

(b) Convex relaxation of feasible space

(c) Initial tightening via 3 branches

(d) Subsequent tightening via 2 branches

Figure 6.4: The trigonometric constraint and its dynamically tightened relaxation.

the circle CC_{pmqn} intersects the ray that starts at N and passes through K. The resulting proposed split is then $\left[\phi^{L}_{pmqn}, \theta^{*}_{pmqn}\right] \cup \left[\phi^{*}_{pmqn}, \phi^{U}_{pmqn}\right]$ (Fig. 7.1d).

$$\min_{c_p, s_p, h_p, v_p, L} \quad L \tag{6.7}$$

$$\text{s.t.} \quad \hat{\alpha}_{pk} c_p + \hat{\beta}_{pk} s_p \geq \hat{\gamma}_{pk} \qquad \forall k \in \{1,2,3\}, \forall p \in \hat{P} \tag{6.8}$$

$$0 \leq c_p x_{pi} - s_p y_{pi} + h_p \leq L \qquad \forall i \in I_p, \forall p \in P \tag{6.9}$$

$$0 \leq s_p x_{pi} + c_p y_{pi} + v_p \leq W \qquad \forall i \in I_p, \forall p \in P \tag{6.10}$$

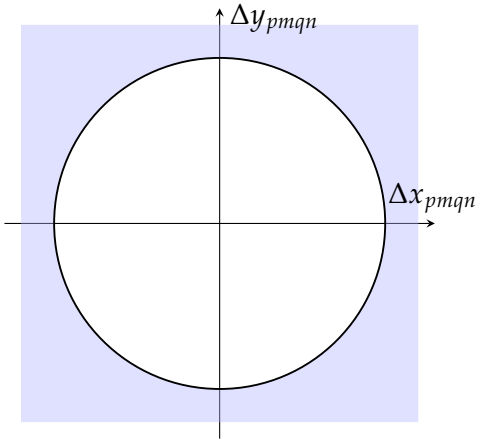
$$\begin{aligned} &\alpha_{pmqnk}\left[\left(c_p x_m - s_p y_m + h_p\right) - \left(c_q x_n - s_q y_n + h_q\right)\right] + \\ &\beta_{pmqnk}\left[\left(s_p x_m + c_p y_m + v_p\right) - \left(s_q x_n + c_q y_n + v_q\right)\right] \geq \\ &\gamma_{pmqnk} \end{aligned} \qquad \begin{aligned} &\forall k \in \{1,2,3\}, \\ &\forall (m,n) \in C^2_{pq}, \\ &\forall (p,q) \in \{P \times P : q > p\} \end{aligned} \tag{6.11}$$

$$-1 \leq c_p, s_p \leq 1 \qquad \forall p \in P \tag{6.12}$$

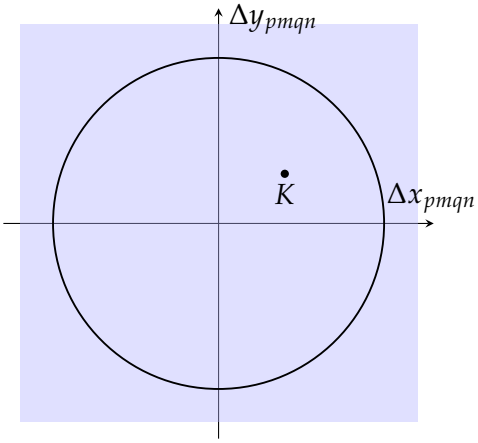
Given the above relaxation strategies, equations (6.7)–(6.12) constitute a relaxed LP formulation to be solved at each node of our BB search process. In this model, the set $\hat{P} \subseteq P$ is the subset of polygons whose rotational angle interval has been branched at least once, while the set $C^2_{pq} \subseteq C_p \times C_q$ denotes the set of circle pairs for which the corresponding ϕ_{pmqn} interval has been branched at least once. All sets $\hat{P}$ and C^2_{pq} begin as empty sets and will be dynamically expanded and updated as the algorithm proceeds. The parameters $\hat{\alpha}_{pk}$, $\hat{\beta}_{pk}$, $\hat{\gamma}_{pk}$, α_{pmqnk}, β_{pmqnk} and γ_{pmqnk} are suitably chosen in each case to define the linear inequalities that relax the original trigonometric and non-overlapping constraints discussed above.

6.3 THE NEW ALGORITHM

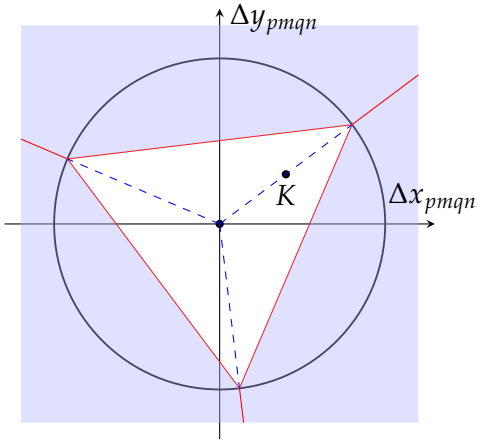
In this section, we discuss the details of our newly proposed algorithm for addressing the Nesting Problem to guaranteed optimality. As already discussed, the algorithm implements a BB search process that solves at each node LP relaxations of the type (6.7)–(6.12). The overall algorithm is illustrated in Fig. 6.6.



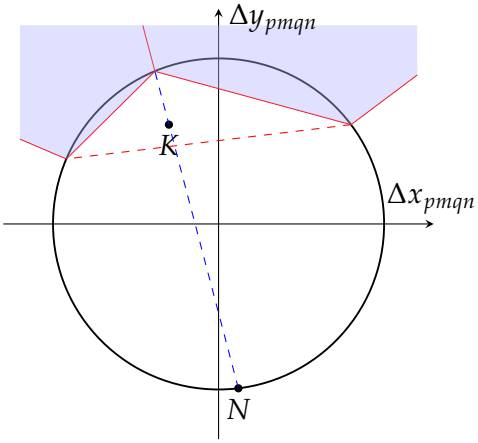
(a) Full feasible space



(b) Convex relaxation of feasible space



(c) Initial tightening via 3 branches



(d) Subsequent tightening via 2 branches

Figure 6.5: The non-overlapping constraint and its dynamically tightened relaxation.

In our algorithm, a circle library $\{C_p\}_{p\in P}$ with N^{lib} initially inscribed circles per polygon is provided in the beginning, but unlike the *QP-Nest* approach, these circles will not be immediately considered in the context of the non-overlapping constraints (6.11). Instead, these circles will be stored in a repository of circles from which to dynamically separate and add such constraints when violated. We should highlight that the violation of non-overlapping constraints is checked repeatedly at every BB node. Hence, querying an off-line computed repository, as opposed to executing the circle-adding procedure each time, has the potential to amount to significant computational benefit. There is also a provision for the circle library to be expanded with new circles in order to improve the approximation in areas of polygon overlap that are not covered by the circles currently in the library. In fact, this ability to gradually incorporate more circles and improve the linear relaxations of the nonconvex constraints within the context of a single BB tree is a key feature of our approach, relieving us from having to resolve problems as in *QP-Nest*.

We remark that, at the root node, the LP relaxation model only includes constraints (6.9), (6.10), and (6.12). Consequently, it does not include any linear relaxation of trigonometric or non-overlapping constraints, since no branching has yet occurred towards enforcing their feasibility.

6.3.1 *Feasibility Checking*

After solving an LP relaxation (6.7)–(6.12), we check whether its optimal solution is a feasible configuration for the Nesting Problem. A feasible configuration has to be insured against two conditions: (i) all rotational angles are correctly defined, that is, trigonometric constraints (6.2) are satisfied; (ii) there is no overlap among polygons. For the former condition, we utilize a parameter ε_θ to denote what resolution of a rotational angle would be satisfactory to achieve. Consequently, we can declare that a trigonometric constraint is satisfied at a given $\left(c_p^*, s_p^*\right)$ when the quantity $\sqrt{c_p^{*2} + s_p^{*2}} - 1$ is bounded from above by $\varepsilon_\theta^{\max} = \frac{1}{\cos\left(\frac{\varepsilon_\theta}{2}\right)} - 1$ and from below by $\varepsilon_\theta^{\min} = 1 - \cos\left(\frac{\varepsilon_\theta}{2}\right)$. For the latter condition, we first check whether all circle pairs covered by the circle library $\{C_p\}_{p\in P}$ satisfy a penetration tolerance ε_ϕ at their applicable $\left(\Delta x_{pmqn}^*, \Delta y_{pmqn}^*\right)$. Assuming this is the case, the penetration-computing procedure proposed in [100] is then applied to check the solution also for possible overlaps not covered by the circle library. We highlight that assessing possible violations against the static library of circles is a step that is significantly faster computationally than the more expensive penetration-computing algorithm. To that end, our strategy ensures that invocation of the latter is only reserved for those cases when

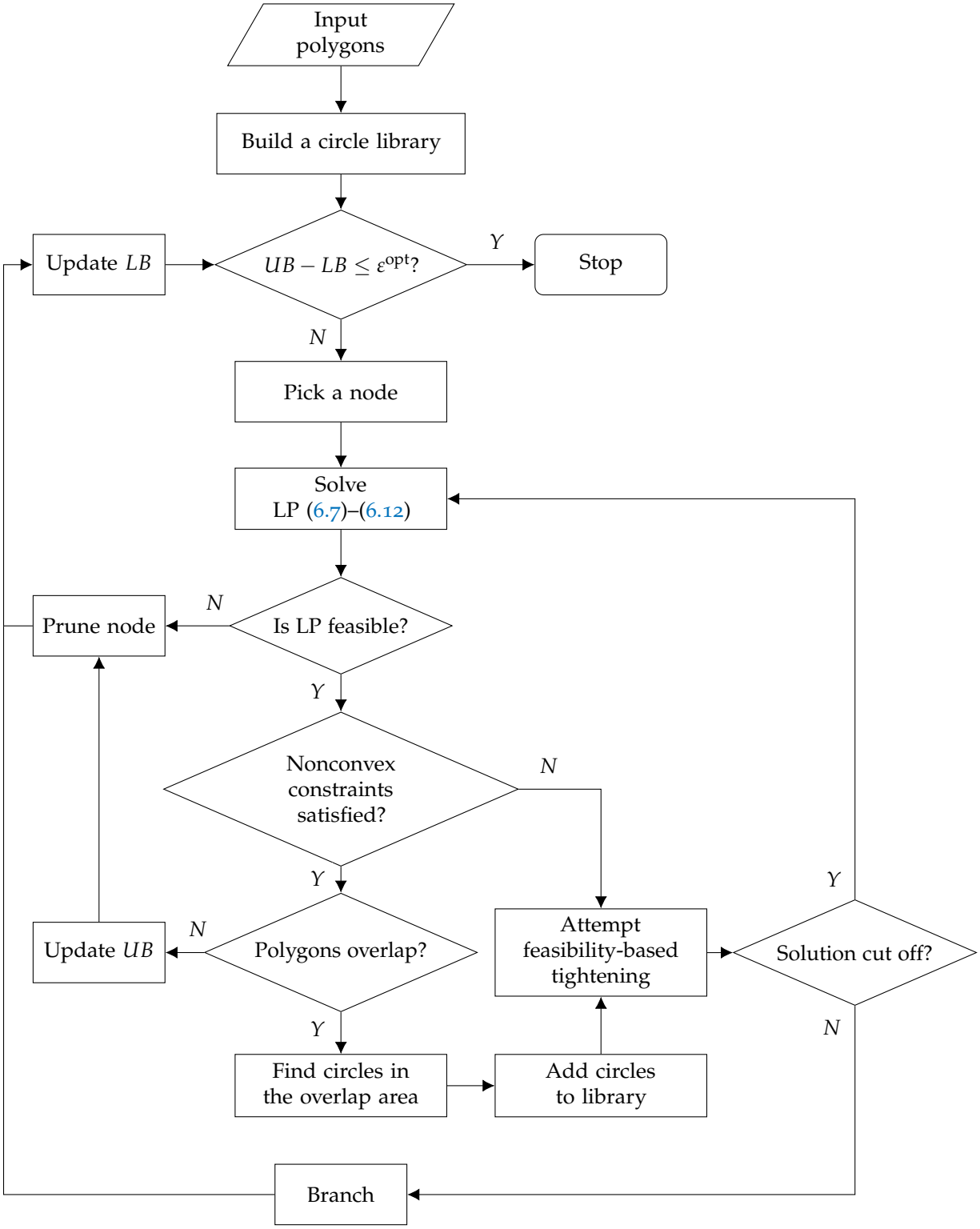


Figure 6.6: A customized BB algorithm for solving the Nesting Problem.

it is absolutely necessary. If both feasibility conditions are met, then a feasible configuration has been obtained; the solution can be considered as a possible new incumbent solution and the node is fathomed. Otherwise, the node must either be tightened or branched upon in order to eliminate the current solution.

6.3.2 *Branching Rule Selection*

As discussed earlier, our BB algorithm utilizes two types of branching rules, one aimed at enforcing the feasibility of the trigonometric constraints and one aimed at doing so for the non-overlapping constraints. In order to properly decide which of these two heterogeneous rules to implement each time branching is to be performed, we first provide a suitable way to normalize the violations of trigonometric constraints. More specifically, let R_p^{ch} denote the radius of a circle with the same area as the convex hull of polygon p. From a geometric point of view, the shortest distance between $\left(c_p^*, s_p^*\right)$ and the corresponding trigonometric circle's circumference, i.e., $\left|\sqrt{c_p^{*2} + s_p^{*2}} - 1\right|$, indicates the extent to which the polygon $p \in P$ is allowed to contract or expand as a result of the relaxation imposed on its rotational angle. Hence, the quantity $V_p^\theta = R_p^{ch}\left|\sqrt{c_p^{*2} + s_p^{*2}} - 1\right|$ can be used as a suitable metric to express a trigonometric constraint's violation in terms of magnitude of polygon overlap. This quantity is directly comparable with the extent to which two circles are overlapping, namely $V_{pmqn}^\phi = R_{pm} + R_{qn} - \sqrt{\Delta x_{pmqn}^{*2} + \Delta y_{pmqn}^{*2}}$.

Let $\hat{p} \in P$ be the polygon with the highest trigonometric constraint violation, and let $(\bar{m}, \bar{n}) \in C_{\bar{p}} \times C_{\bar{q}}$, where $(\bar{p}, \bar{q}) \in \{P \times P : \bar{q} > \bar{p}\}$, be the circle pair with the highest non-overlapping constraint violation. We choose to branch based on the former if $V_{\hat{p}}^\theta > \tau^{\text{no-ov}} V_{\bar{p}\bar{m}\bar{q}\bar{n}}^\phi$, while we branch on the latter otherwise. Here, $\tau^{\text{no-ov}} < 1$ is a weighting factor to prioritize branching on trigonometric constraints before branching on non-overlapping ones, which our computational experience has determined as beneficial to do. Furthermore, since branching into two subdomains will produce fewer child nodes in the BB tree than the initial branching into three subdomains, the violations of currently unbranched non-overlapping constraints are further discounted by some factor $\tau^{\text{unbr}} < 1$.

6.3.3 *Feasibility-based Node Tightening*

Feasibility-based tightening of each BB node can be performed by utilizing the geometric interpretation of the non-overlapping constraints. Consider constraints (6.11) for some $(m,n) \in C^2_{pq}$, where $(p,q) \in \{P \times P : q > p\}$, conveniently rewritten as

$$\alpha_{pmqnk}\Delta x_{pmqn} + \beta_{pmqnk}\Delta y_{pmqn} \geq \gamma_{pmqnk} \quad \forall\, k \in \{1,2,3\}\,. \tag{6.13}$$

For any given $(\bar{m},\bar{n}) \in C_p \times C_q$, Δx_{pmqn} and Δy_{pmqn} can be expressed in terms of $\Delta x_{p\bar{m}q\bar{n}}$ and $\Delta y_{p\bar{m}q\bar{n}}$ via relationships

$$\begin{aligned}\Delta x_{pmqn} &= \Delta x_{p\bar{m}q\bar{n}} + \left(c_p\Delta x_{pm\bar{m}} - s_p\Delta y_{pm\bar{m}}\right) - \left(c_q\Delta x_{qn\bar{n}} - s_q\Delta y_{qn\bar{n}}\right)\\ \Delta y_{pmqn} &= \Delta y_{p\bar{m}q\bar{n}} + \left(s_p\Delta x_{pm\bar{m}} + c_p\Delta y_{pm\bar{m}}\right) - \left(s_q\Delta x_{qn\bar{n}} + c_q\Delta y_{qn\bar{n}}\right),\end{aligned}$$

where: $\Delta x_{pm\bar{m}} \coloneqq x_{pm} - x_{p\bar{m}},\ \Delta y_{pm\bar{m}} \coloneqq y_{pm} - y_{p\bar{m}},\ \Delta x_{qn\bar{n}} \coloneqq x_{qn} - x_{q\bar{n}},\ \Delta y_{qn\bar{n}} \coloneqq y_{qn} - y_{q\bar{n}}$.

Combining these with (6.13), we have

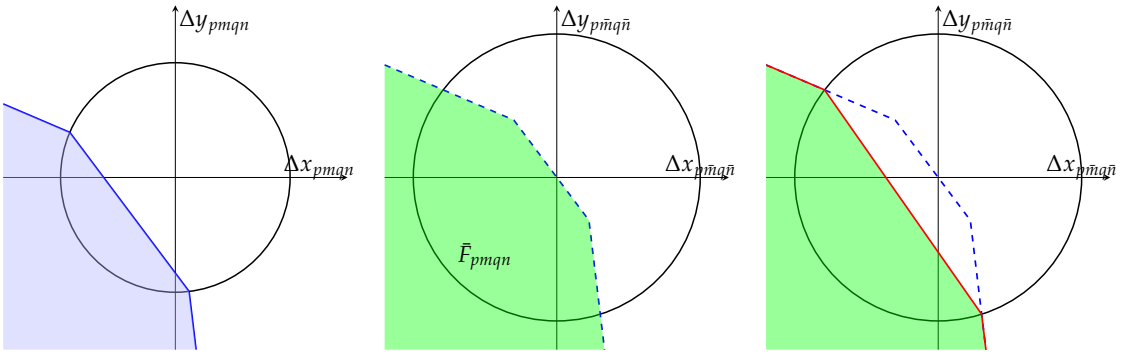
$$\alpha_{pmqnk}\Delta x_{p\bar{m}q\bar{n}} + \beta_{pmqnk}\Delta y_{p\bar{m}q\bar{n}} \geq RHS_k\left(\theta_p,\theta_q\right) \quad \forall\, k \in \{1,2,3\}\,,$$

$$\begin{aligned}\text{where: } RHS_k\left(\theta_p,\theta_q\right) \coloneqq \gamma_{pmqnk} &- \alpha_{pmqnk}\left[\left(c_p\Delta x_{pm\bar{m}} - s_p\Delta y_{pm\bar{m}}\right) - \left(c_q\Delta x_{qn\bar{n}} - s_q\Delta y_{qn\bar{n}}\right)\right]\\ &- \beta_{pmqnk}\left[\left(s_p\Delta x_{pm\bar{m}} + c_p\Delta y_{pm\bar{m}}\right) - \left(s_q\Delta x_{qn\bar{n}} + c_q\Delta y_{qn\bar{n}}\right)\right].\end{aligned}$$

The end result is a set of valid constraints (6.14), which are implied by (6.13), but which can be further tightened as we shall show below.

$$\alpha_{pmqnk}\Delta x_{p\bar{m}q\bar{n}} + \beta_{pmqnk}\Delta y_{p\bar{m}q\bar{n}} \geq \max_{\substack{\theta_p\in[\theta_p^L,\theta_p^U],\\ \theta_q\in[\theta_q^L,\theta_q^U]}} RHS_k\left(\theta_p,\theta_q\right) \quad \forall\, k \in \{1,2,3\} \tag{6.14}$$

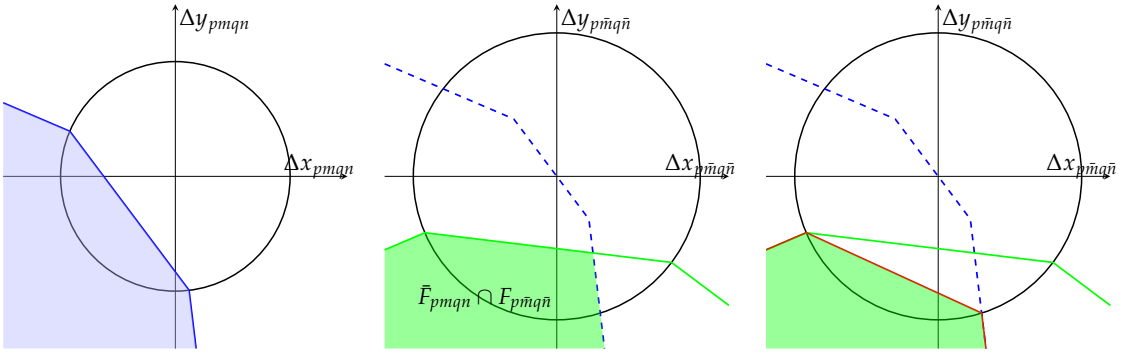
Let F_{pmqn} denote the feasible $\left(\Delta x_{pmqn}, \Delta y_{pmqn}\right)$ space constrained by (6.13) (Fig. 6.7a), and let $\bar{F}_{pmqn}$ denote the feasible $\left(\Delta x_{p\bar{m}q\bar{n}}, \Delta y_{p\bar{m}q\bar{n}}\right)$ space constrained by (6.14) (Fig. 6.7b). Note how the above substitution amounts to projecting F_{pmqn} onto the $\left(\Delta x_{p\bar{m}q\bar{n}}, \Delta y_{p\bar{m}q\bar{n}}\right)$ plane, forming $\bar{F}_{pmqn}$. Since any point inside the circle $CC_{p\bar{m}q\bar{n}}$ is essentially infeasible, $\bar{F}_{pmqn}$ can be further tightened as Fig. 6.7c demonstrates. Clearly, the tightened space is not implied by constraints (6.13). For cases where the feasible space $F_{p\bar{m}q\bar{n}}$ is already partially constrained (e.g., due to a previous branching decision associated with circle $CC_{p\bar{m}q\bar{n}}$), then



(a) Feasible space F_{pmqn} (b) Projection of F_{pmqn} onto $F_{p\bar{m}q\bar{n}}$ (c) Feasible space $F_{p\bar{m}q\bar{n}}$ after tightening

Figure 6.7: Feasibility-based tightening (special case when $F_{p\bar{m}q\bar{n}} = \mathbb{R}^2$)

.



(a) Feasible space F_{pqmn} (b) Projection of F_{pmqn} onto $F_{p\bar{m}q\bar{n}}$ (c) Feasible space $F_{p\bar{m}q\bar{n}}$ after tightening

Figure 6.8: Feasibility-based tightening (general case when $F_{p\bar{m}q\bar{n}} \subset \mathbb{R}^2$).

$\bar{F}_{pmqn}$ shall be intersected with the currently applicable space $F_{p\bar{m}q\bar{n}}$ (Fig. 6.8b), and it is this intersection that may be further tightened (Fig. 6.8c).

We apply the above projection and tightening procedure every time an LP relaxation is solved. More specifically, we focus on the up to N^{fbt} polygon pairs $(p, q) \in \{P \times P : q > p\}$ that overlap the most, identify the pair of circles $(\bar{m}, \bar{n}) \in C_p \times C_q$ that are responsible for these violations, and attempt to tighten the feasible spaces $F_{p\bar{m}q\bar{n}}$ using information from any relevant space F_{pmqn}. If the resulting linear cuts are strong enough to cut off the relaxed solution, these cuts are immediately added to the model (augmenting the set C^2_{pq}, if necessary) and the latter is resolved. It is important to note, however, that even if these cuts do not strengthen the LP relaxation, they may prove useful in tightening the model at some later point during the course of the BB search. To take advantage of this fact, we save the linear cuts to node-specific libraries of previously identified cuts. Future node

solutions are first checked for possible violations against these libraries before applying the full feasibility-based tightening process.

Finally, we remark that feasibility-based tightening can also be applied as soon as a branching step based on an implicit angle ϕ_{pmqn} is taken, in order to strengthen the newly-formed children nodes. More specifically, the restricted feasible space F_{pmqn} in the child node's domain that resulted from application of the branching disjunction can be projected so as to tighten any applicable spaces $F_{p\bar{m}q\bar{n}}$.

6.3.4 *Tighter Variable Bounds*

Consider the largest circle that can be inscribed in the convex hull of a polygon $p \in P$, and let $R_p^{ch,in}$ denote its radius. Furthermore, assume (without loss of generality) that the coordinates of the polygon's vertices have been adjusted such that this circle is centered at $(0,0)$. Since the polygon's convex hull must also lie within the boundaries of the sheet, the following bounds apply for the translation variables.

$$R_p^{ch,in} \leq h_p \leq L^{\text{BKS}} - R_p^{ch,in} \qquad \forall\ p \in P \tag{6.15}$$

$$R_p^{ch,in} \leq v_p \leq W - R_p^{ch,in} \qquad \forall\ p \in P \tag{6.16}$$

Here, L^{BKS} denotes the "best-known solution" (best upper bound) that we may possess (e.g., by employing a heuristic algorithm) for the instance in question. Note that, each time a better incumbent is found during the BB process, the upper bound of h_p is updated according to the new (smaller) L^{BKS} value. From this perspective, constraints (6.15) can be viewed as globally-valid, optimality-based cuts.

6.3.5 *Symmetry Breaking Constraints*

As [100] and [145] have pointed out, several types of symmetry apply in the context of the Nesting Problem. To that end, we utilize appropriate symmetry-breaking constraints in our implementation.

(i) Polygon rotational symmetry: For a polygon $p \in P$ with a rotationally symmetric angle, Θ_p, relative to its center (e.g., for a perfect square, $\Theta_p = \pi/4$), one may break symmetry by imposing $\theta_p \leq \Theta_p$.[2]

2 For compatibility with constraints (6.15) and (6.16), the symmetry center must also be moved to point $(0,0)$.

(ii) Identical polygons: When two polygons p and q, such that $q > p$, are identical, then $h_p \leq h_q$ can be added into the model to break this symmetry.[3]

(iii) Sheet orientation: This symmetry arises due to the fact that an equivalent packing solution can always be obtained when rotating the entire configuration by 180 degrees. Hence, in all instances one can enforce $\theta_{p^\star} \leq \pi$, where $p^\star \in P$ is a specific polygon that one chooses for purposes of defining this constraint.

6.4 COMPUTATIONAL STUDIES

We now test our customized BB approach and compare its performance with the *QP-Nest* approach. Our algorithm was implemented in C++ and the LP relaxation models were solved via CPLEX Optimization Studio 12.7.1 through the C application programming interface. All applicable algorithm tuning parameters are listed in Table 6.1. The *QP-Nest* approach was initialized with $N^{\text{init}} = 3$ circles per polygon,[4] and solved within the GAMS 25.0.2 environment utilizing several state-of-the-art general purpose GO solvers, namely BARON 17.10.16 [166], GloMIQO 2.3 [124], LINDOGlobal 11.0, and SCIP 5.0 [3]. For a fair comparison, applicable strengthening techniques such as symmetry-breaking and tighter variable bounds were also incorporated into the *QP-Nest* implementation. All computational experiments were conducted on an single thread of an Intel Xeon CPU E5-2687Wv3 @ 3.10GHz with 32GB of RAM.

6.4.1 *Instances*

In order to generate our suite of benchmark testing instances, we use the data from the well-studied instance SHAPES2 [128], which features a total of seven different polygons shown in Fig. 6.9. Using all possible combinations of these polygons, we generated a total of 64 nesting instances, namely 35 four-polygon, 21 five-polygon, 7 six-polygon, and 1 seven-polygon instances. According to the dataset, the fixed width of the rectangular sheet is $W = 15$.[5]

3 We prefer to use $h_p + v_p \leq h_q + v_q$, because it is likely to break symmetry in more instances.

4 In the original implementation of [100], the values 5 and 13 were proposed for parameter N^{init}. However, in most instances used in this study, we found that using 5 or more circles per each and every polygon leads to initial QCP models that cannot be solved within the time limit, i.e., the GO solvers could not even complete the first (and also easiest) iteration. In contrast, using the setting $N^{\text{init}} = 3$ and allowing the circle sets to grow more judiciously as overlaps are identified led to much better performance of the *QP-Nest* approach.

5 For the sake of better numerical stability, in our implementations the width was normalized down to the value of 1, and all coordinates were scaled correspondingly.

Table 6.1: Parameters used in the new algorithm.

Symbol	Parameter	Value
ε_θ	Resolution of rotational angle	$\frac{2\pi}{360}$
ε_ϕ	Tolerance for circle penetration checking	10^{-3}
ε^{opt}	Optimality tolerance	10^{-3}
N^{lib}	Number of inscribed circles per polygon initially in library	30
N^{fbt}	Maximum number of circle pairs for which feasibility-based tightening is attempted	6
$\tau^{\text{no-ov}}$	Weighting factor for violation of a non-overlapping constraint	0.1
τ^{unbr}	Weighting factor for violation of an unbranched non-overlapping constraint	0.3

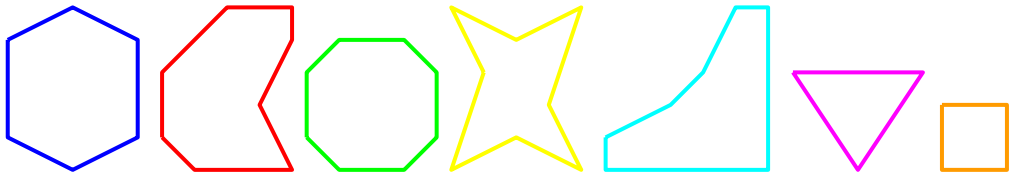


Figure 6.9: Polygon shapes used in our benchmark study.

6.4.2 *Fixed Orientation Case*

Before we attempt to solve for nestings under free rotation, we first analyze our algorithm in the case where the orientation of the polygons is fixed at their nominal values (as shown in Fig. 6.9). In this setting, variables c_p and s_p are fixed for all $p \in P$, and hence, no violation of the trigonometric constraints occurs. Solving the problem exclusively focuses on enforcing the non-overlapping restrictions. For our custom BB algorithm, we adopted the best-bound-first search node selection strategy. Both the new algorithm and the *QP-Nest* approach were provided with an initial upper bound. Given the wide variety of possible heuristics that could be available to obtain such an initial incumbent solution, we utilize the following two options: (i) a naive solution, wherein polygons are lined-up left to right (Fig. 6.10), and (ii) the optimal solution itself. Note how these two cases constitute the two extremes in terms of the performance of a heuristic scheme that one may employ in a real setting, and hence, the results obtained in this manner bound the performance of

the nesting algorithms with respect to the effect of the quality of the initial incumbent provided.

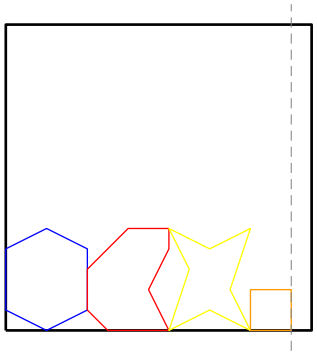


Figure 6.10: A naive solution.

Table 6.2 presents the computational results for when *QP-Nest* and our custom BB approach are initialized with the naive incumbent. The table presents the number of test instances of a given input size (# inst.), and the average number of nonconvex polygons in those instances. For those instances that were solved to provable optimality within the chosen time limit of 2 hours, we present the average solution time, while for those instances that were not, we present the average residual gap, defined as $(UB - LB)/UB$ at the time limit. With respect to the results of *QP-Nest*, each instance is solved independently with each of the four GO solvers we consider in this study, and we adopt the best of those runs as the one to account for in the table (instance-wise "best-of-four"). Consequently, this column is to be interpreted as an upper bound on the performance of *QP-Nest*.

Table 6.2: Comparison between *QP-Nest* and the new algorithm for the case of fixed orientation, using a naive solution as incumbent.

# polygons	# inst.	Avg. # of nonconvex polygons	"Best-of-Four" *QP-Nest* Approach			New BB-based Approach		
			# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
4	35	1.7	35	309	–	35	4	–
5	21	2.1	12	1,517	67.9	21	94	–
6	7	2.6	1	2,155	70.5	7	591	–
7	1	3.0	0	–	76.0	0	–	0.2

Table 6.2 reveals that, when the naive solution is passed as an initial upper bound, our custom BB approach can solve all test instances to optimality except the seven-polygon

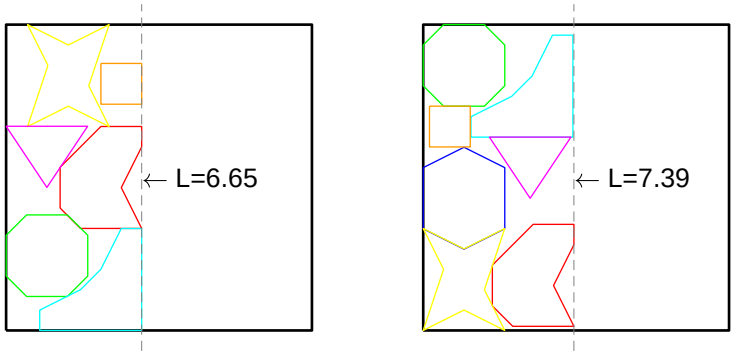


Figure 6.11: Optimal solutions for representative six-polygon (left) and seven-polygon (right) instances under fixed orientation.

problem, which features a residual gap of 0.2%. At the same time, the *QP-Nest* approach could only solve 16 out of 21 five-polygon instances and only 1 out of 7 six-polygon instances, with approximately 70% residual gap for the unsolved instances. The solution time of our approach was also much less than that for the *QP-Nest* approach. Table 6.3 presents the numerical results when the optimal solution is provided as the initial solution. Under this setup, both approaches demonstrate improved performance over the case with a naive initialization. More specifically, the new BB algorithm can now solve all test instances (including the seven-polygon one) to global optimality within less computation time. In contrast, the *QP-Nest* approach could not solve any additional instances compared to the naive incumbent case. It is noteworthy, however, that the average residual gap for the *QP-Nest* approach decreases dramatically, which is to be attributed to the better upper bound involved in the calculation of residual gaps. Figure 6.11 presents the optimal solutions for a representative six-polygon and the seven-polygon nesting instances.

Table 6.3: Comparison between *QP-Nest* and the new algorithm for the case of fixed orientation, using an optimal solution as incumbent.

# polygons	# inst.	Avg. # of nonconvex polygons	"Best-of-Four" *QP-Nest* Approach			New BB-based Approach		
			# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
4	35	1.7	35	116	–	35	2	–
5	21	2.1	16	1,587	1.6	21	55	–
6	7	2.6	1	1,398	3.1	7	406	–
7	1	3.0	0	–	6.6	1	4,176	–

6.4.3 *Free Rotation Case*

Here, we consider the general case where the polygons are allowed to rotate arbitrarily in the interest of achieving better nesting configurations. These experiments were conducted with a increased time limit of 10 hours. All algorithmic parameters were kept the same as in the fixed-orientation case, with only exception being the node selection strategy for our BB-based algorithm, which was changed to depth-first search in order to alleviate memory limitations arising from large lists of unprocessed nodes during the search. With regards to choosing an initial upper bound, we again utilize the naive initial incumbent solutions as an example of a poor heuristic. However, since no optimal solutions were available for these (never before solved) instances, we utilize the optimal solutions we obtained for the case of fixed orientation (see Table 6.3) as a representative set of high-quality initial solutions.

From an overall numerical tractability perspective, we observe that solving for nestings with free rotation is significantly more challenging compared to the case when the polygon orientation is fixed. As Tables 6.4 and 6.5 show, our customized BB approach could solve all four-polygon instances and approximately half of the five-polygon ones when the good incumbent was passed initially. In contrast, the *QP-Nest* approach could only solve about two-thirds of the four-polygon problems and none of the five-polygon ones. Figure 6.12 presents a handful of optimal five-polygon nestings under arbitrary rotation. We highlight that this is the first time in the open literature that these solutions are obtained.

Table 6.4: Comparison between *QP-Nest* and the new algorithm for the case of free rotation, using a naive solution as incumbent.

# polygons	# inst.	Avg. # of nonconvex polygons	"Best-of-Four" *QP-Nest* Approach			New BB-based Approach		
			# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
4	35	1.7	22	6,565	71.2	34	1,646	58.3
5	21	2.1	0	–	80.1	2	18,938	54.3
6	7	2.6	0	–	94.6	0	–	66.9
7	1	3.0	0	–	$\star$	0	–	61.5

$\star$ = GO solvers could not complete the first iteration of the *QP-Nest* approach.

Table 6.5: Comparison between *QP-Nest* and the new algorithm for the case of free rotation, using an optimal fixed-orientation solution as incumbent.

# polygons	# inst.	Avg. # of nonconvex polygons	"Best-of-Four" *QP-Nest* Approach			New BB-based Approach		
			# opt.	Avg. t (sec)	Avg. gap (%)	# opt.	Avg. t (sec)	Avg. gap (%)
4	35	1.7	23	6,764	13.0	35	287	–
5	21	2.1	0	–	26.9	10	17,058	41.6
6	7	2.6	0	–	71.9	0	–	54.8
7	1	3.0	0	–	⋆	0	–	52.6

⋆ = GO solvers could not complete the first iteration of the *QP-Nest* approach.

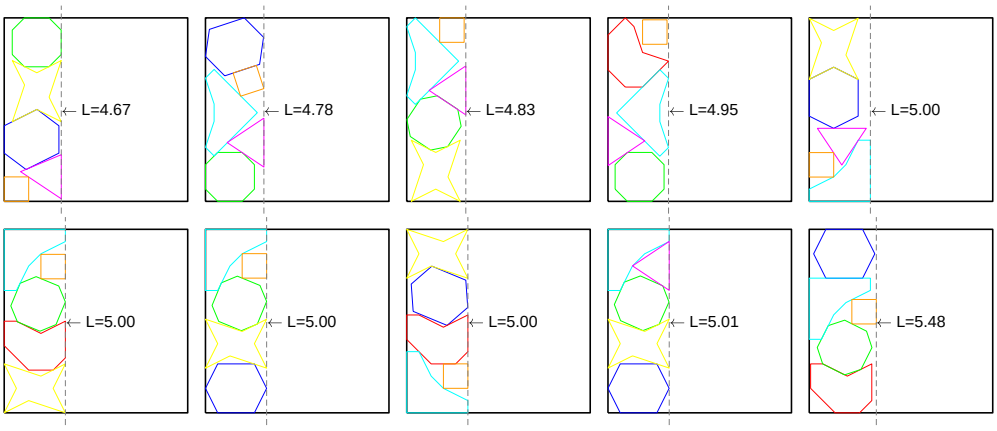


Figure 6.12: Optimal solutions for several five-polygon instances under free rotation.

6.5 CONCLUSIONS

Nesting irregular shapes within rectangular sheets while allowing both translation and free rotation is an industrially important, yet notoriously hard, problem for which little effort has been devoted on how to solve to provable optimality. To the best of our knowledge, the *QP-Nest* approach proposed in [100] was the only exact method that could obtain global optimal solutions for certain instances up to four polygons. This performance was confirmed also via the computational study we conducted in this work, where we used the latest available versions of four different global optimization solvers as the basis for the *QP-Nest* method.

Our new approach adopts the circle-covering idea, but instead of solving quadratically constrained programming nesting model approximations iteratively, it proposes to tackle the full problem in a single branch and bound search, using a customized linear relaxation and a novel branching scheme. We also propose several model tightening techniques, which can be readily incorporated in the dynamic process of the search. Many of the proposed ideas can in fact have wider applicability, beyond the context of shape nesting, when addressing optimization models that feature reverse convex quadratic constraints. Our computational studies on a suite of 64 benchmark instances demonstrated the competitiveness of the new approach. Meanwhile, for the first time in the open literature, five-polygon nestings under free rotation were solved to global optimality.

6.6 APPENDIX: NOMENCLATURE

Indices

p, q	polygon
i, j	vertex
m, n	circle

Sets

P	set of polygons to nest
I_p	set of vertices of polygon p
C_p	set of circles inscribed in polygon p
$\hat{P}$	subset of polygons whose rotational angle interval has been branched
C^2_{pq}	set of circle pairs between polygon p and polygon q that have been branched
F_{pmqn}	feasible space of relaxed non-overlapping constraints for circle m in polygon p and circle n in polygon q

Continuous Variables

h_p	horizontal translation of polygon p
v_p	vertical translation of polygon p
θ_p	rotation of polygon p
c_p	cosine of θ_p
s_p	sine of θ_p
L	length of sheet on which shapes are nested
$(\Delta x_{pmqn}, \Delta y_{pmqn})$	displacement between circle m in polygon p and circle n in polygon q
ϕ_{pmqn}	angle between center of circle m in polygon p and circle n in polygon q

Parameters

(x_{pi}, y_{pi})	nominal coordinates of vertex i on polygon p
(x_{pm}, y_{pm})	nominal coordinates of center of circle m inscribed in polygon p
R_{pm}	radius of circle m inscribed in polygon p
W	width of sheet on which shapes are nested
L_{BKS}	sheet length of best known nesting solution
N^{init}	number of circles initially inscribed per polygon
N^{lib}	number of circles initially present in a library of inscribed circles per polygon
(θ_p^L, θ_p^U)	lower and upper bounds on the rotation of polygon p
$(\phi_{pmqn}^L, \phi_{pmqn}^U)$	lower and upper bounds on the angle between center of circle m in polygon p and circle n in polygon q
$\hat{\alpha}_{pk}, \hat{\beta}_{pk}, \hat{\gamma}_{pk}$	coefficients to define relaxed trigonometric constraint k for polygon p
$\alpha_{pmqnk}, \beta_{pmqnk}, \gamma_{pmqnk}$	coefficients to define relaxed non-overlapping constraint k for circle m in polygon p and circle n in polygon q
ε_θ	resolution of rotational angles
ε_ϕ	tolerance on shape penetration

7

ON TACKLING REVERSE CONVEX CONSTRAINTS FOR NON-OVERLAPPING OF CIRCLES

In this chapter, we study the circle-circle non-overlapping constraints, a form of reverse convex constraints that often arise in optimization models for cutting and packing applications. The feasible region induced by the intersection of circle-circle non-overlapping constraints is highly non-convex, and standard approaches to construct convex relaxations for spatial branch-and-bound global optimization of such models typically yield unsatisfactory loose relaxations. Consequently, solving such non-convex models to guaranteed optimality remains extremely challenging even for the state-of-the-art codes. In this chapter, we apply a purpose-built branching scheme on non-overlapping constraints and utilize strengthened intersection cuts and various feasibility-based tightening techniques to further tighten the model relaxation. We embed these techniques into a branch-and-bound code and test them on two variants of circle packing problems. Our computational studies on a suite of 75 benchmark instances yielded, for the first time in the open literature, a total of 54 provably optimal solutions, and it was demonstrated to be competitive over the use of the state-of-the-art general-purpose global optimization solvers.

7.1 INTRODUCTION

The circle-circle non-overlapping constraint is imposed to guarantee that two circles do not overlap, which can be achieved by requiring that their centers are sufficiently far in terms of Euclidean distance. In particular, the constraint has the mathematical form

$$\left(a_i - a_j\right)^2 + \left(b_i - b_j\right)^2 \geq \left(r_i + r_j\right)^2, \tag{7.1}$$

where (a_i, b_i) and (a_j, b_j) represent the coordinates of the centers of two circles i and j, while r_i and r_j denote their corresponding radii, respectively.

Studying this type of constraint has both theoretical and practical interest. From a theoretical perspective, mathematical models with non-overlapping constraints usually have a highly non-convex solution space whose convexification towards a tight relaxation still poses a tremendous challenge for the global optimization community; solving these models to guaranteed optimality remains extremely hard, even for the state-of-the-art solvers. From a practical point of view, circle-circle non-overlapping constraints often appear in many models for cutting and packing applications, and efforts to solve them more efficiently can add significant value to industry.

The first and foremost, archetypal family of problems of interest are the *circle packing problems*. These come in many variants, such as packing identical circles into a rectangular container with the objective of minimizing the container's area [90], or identifying the minimal radius of a circle within which other circles can simultaneously be placed [149], among others. Related applications include container loading, cyclinder packing and wireless communication network layout, to name but a few [46]. Another related cutting and packing application is the irregular shape nesting problem, in which one seeks a feasible configuration for a given set of irregular shapes within a rectangular sheet of a fixed width such that no overlap among these shapes exists and the sheet length is minimized. This problem often appears when carving out of metal rolls parts for automobiles, airplanes and other machinery, as well as when cutting leather and fabrics for apparel and upholstery applications. The work of [100] proposed to enforce no overlap between any two irregular shapes by imposing that circles inscribed within one shape do not overlap with circles inscribed within another one, resulting in an optimization model with non-convex quadratic constraints. Using this approach, the authors obtained for the first time optimal nesting solutions to a four polygon problem, using off-the-shelf global optimization solvers to tackle the non-convex quadratically constrained models. Focusing on the circle-inscribing approach for nesting problems, Chapter 6 proposed a novel branching scheme on circle-circle non-overlapping constraints and demonstrated the ability to find global optimal nestings to benchmark instances with five polygons, as well as solutions to instances with up to seven polygons under fixed rotation. Both the circle packing problem and the irregular shape nesting problem have been studied extensively, with most of the focus being on the development of heuristics to obtain large-scale packings [10, 117]. While heuristic methods are practically valuable for the generation of feasible solutions, they cannot rigorously quantify the optimality gap, and therefore provide no guarantees regarding how much value is left on the table by the packings produced.

Arguably, the exact solution of optimization models with non-overlapping constraints is of great practical importance.

Mathematical models with circle-circle non-overlapping constraints fall into the *quadratically constrained quadratic programming* (QCQP) class, which is currently a very active research area in global optimization. Generally speaking, there are two approaches for convexifying QCQPs, namely semi-definite programming (SDP) relaxation and multi-term polyhedral relaxation with reformulation-linearization techniques [29]. The polyhedral approach calls for approximating the convex hull of the non-convex feasible solution space via linear inequalities, while the SDP method is to characterize the convex hull via semi-definite cones. Interested readers are referred to [13] for details. Although the SDP relaxation is usually tighter than the polyhedral one, optimizing an SDP problem is generally computationally less efficient than a linear one. Therefore, general-purpose global solvers [125, 166, 172] commonly rely on polyhedral relaxations to address QCQPs. Despite many recent advances in global optimization solvers, however, optimizing models with non-overlapping constraints is still incredibly challenging. Our computational studies on circle packing instances (Section 7.5) also show that general-purpose global solvers could only solve instances of up to 10 circles to guaranteed optimality.

At the same time, there exist only a handful of attempts in the open literature to address such models in a customized approach. The work of [103] considered the problem of packing a fixed number of identical circles into a given square with the objective of maximizing the circle radius. The authors conducted a theoretical comparison of several convexification techniques on non-overlapping constraints, including polyhedral and semi-definite relaxations, and assessed their strength theoretically. They pessimistically concluded that the current state-of-the-art bounding techniques are only effective for small-size circle packing problems. In addition, the work of [141] proposed to approximate the quadratic function in the left-hand side of a non-overlapping constraint via piecewise linear functions. This approach necessitates the introduction of binary variables to represent the latter, resulting in a mixed-integer linear programming model, and thus might become impractical as the approximation accuracy increases.

This chapter presents an extension of the branching strategy on non-overlapping constraints previously developed in Chapter 6. The main idea is rooted in the geometric interpretation of a circle-circle non-overlapping constraint. More specifically, the constraint (7.1) dictates that any feasible point in the transformed variable space $(a_i - a_j)$–$(b_i - b_j)$ should lie outside (or exactly on) the circumference of the circle that has a center with coordinates $(0, 0)$ and has a radius of $r_i + r_j$. Unlike a traditional, general-purpose spatial branch-and-

bound based solver, which must generically handle each non-convex constraint by relaxing it individually into its convex relaxation and tighten that relaxation via branching on the domains of the problem's variables, our perspective is to split the feasible, non-convex domain imposed by non-overlapping constraints and enforce feasibility in a more direct fashion.

More specifically, we will follow an approach that branches on the domain of constraint, rather than the domain of variables. Additionally, we observe that the non-overlapping constraint imposes a *reverse convex* region [91]. It is well-known that reverse convex constraints can induce *intersection cuts* [23, 91], which are also called *concavity cuts* [95] in the global optimization community. These cuts are computationally cheap to generate, and they can be utilized to strengthen the model relaxation. In order to further tighten the model relaxation, we also propose three feasibility-based tightening techniques. The distinct contributions of our work can be summarized as follows.

- We develop a customized branch-and-bound (BB) approach for solving problems with circle-circle non-overlapping constraints.
- We generalize the intersection cut formula from the seminal paper of [23] to more generic cases with arbitrary variable bounds, and we apply a strengthened version of these cuts to tighten the BB node relaxations.
- We propose three types of feasibility-based tightening techniques to further strengthen the BB node relaxations.
- We conduct a comprehensive computational study based on two popular variants of the circle packing problem to demonstrate that our approach achieves superior performance over the use of various state-of-the-art general-purpose global optimization solvers.

The remainder of the chapter is organized as follows. In Section 7.2, we provide a concise description of an optimization model with non-convexities that stem from the presence of non-overlapping constraints. In Section 7.3, we discuss how to construct a suitable linear relaxation of such a model that we can use as the basis of our BB algorithm, followed by a presentation of the complete algorithmic procedure. In Section 7.4, we discuss strengthening techniques, including strengthened intersection cuts and feasibility-based tightening. Section 7.5 presents results on the algorithm's computational performance as well as a comprehensive comparison against the use of five different state-of-the-art global optimization solvers. Finally, we conclude our chapter with some remarks in Section 7.6.

7.2 PROBLEM DEFINITION

In this work, we focus on the following problem (7.2).

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & f_k(x) \leq 0 \qquad && \forall k \in \mathcal{K} \\ & x_i^2 + x_j^2 \geq r_{ij}^2 && \forall\, (i,j) \in \mathcal{M} \\ & x^L \leq x \leq x^U, \end{aligned} \tag{7.2}$$

where $f_k(x) : \mathbb{R}^n \mapsto \mathbb{R}$ denotes a convex function, for each applicable index $k \in \mathcal{K}$, while $x_i^2 + x_j^2 \geq r_{ij}^2$ represents a circle-circle non-overlapping constraint, for each applicable ordered pair $(i,j) \in \mathcal{M}$.[1] Without loss of generality, we also assume $-\infty \leq x_i^L \leq x_i^U \leq +\infty$, for all $i \in \{1,2,...,n\}$. Since non-overlapping constraints induce a non-convex solution space, problem (7.2) is a global optimization problem. In this work, we aim to develop exact, custom-built methods for solving this problem to provable global optimality.

7.3 SOLUTION APPROACH

Our approach is based on the construction of a branch-and-bound tree where linear programming (LP) relaxations are solved at each node. Specifically, the convex domain defined by $f_k(x) \leq 0$ in (7.2) is outer-approximated by linear inequalities. The main challenge arises from the non-convexities introduced by the non-overlapping constraints, $x_i^2 + x_j^2 \geq r_{ij}^2$, which we address in the remainder of this section.

7.3.1 *Customized Model Relaxation*

Here, we shall discuss how a suitable linear relaxation is constructed as the basis of the BB search algorithm. First, we adopt the branching scheme from our previous work in Chapter 6, applied on a non-overlapping constraint from the set $(i,j) \in \mathcal{M}$. Let $\bar{x}$ represent the current optimal solution of the LP relaxation at some BB node, and let D_{ij} denote a disk that is centered at the origin $(0,0)$ and has a radius of r_{ij}. Furthermore, let θ_{ij} be the angle between the positive x_i-axis and the point given by the coordinates $(\bar{x}_i, \bar{x}_j)$, and let $\left[\theta_{ij}^L, \theta_{ij}^U\right]$ represent its feasible interval, which is originally equal to $[0, 2\pi)$.

1 For ease of exposition, we simplify the notation of equation (7.1), as follows: $x_i \leftarrow a_i - a_j$, $x_j \leftarrow b_i - b_j$ and $r_{ij} \leftarrow r_i + r_j$.

In the x_i–x_j space, the non-overlapping constraint requires that all feasible points project either outside or exactly on the circumference of disk D_{ij} (Fig. 7.1a). Initially, dropping this constraint from consideration results in a convex relaxation that is defined by the full space (Fig. 7.1b). Feasibility of the non-convex constraint can be then be gradually enforced by branching on the implicit domain of variable θ_{ij} variable, and by tightening the relaxation in the x_i–x_j space, as follows.

Let X denote a solution of the relaxed problem with coordinates $(\bar{x}_i, \bar{x}_j)$ such that it violates the original constraint. Using this solution as a guide, we can split the circumference of D_{ij} into three parts (each representing an arc of angle $2\pi/3$ radians), and we can define a set of three linear constraints (corresponding to one secant and two boundary lines) to form the convex relaxation of the original non-overlapping constraint in each of the three resulting subdomains. As long as they are properly oriented with respect to X (Fig. 7.1c), it can be guaranteed that all of the three subdomains exclude the parent solution.

In subsequent violations of the same constraint $(i, j) \in \mathcal{M}$, one could further dissect the applicable θ_{ij} interval into two parts. Again, as long as the solution X is used as the guide to define the branching point, one can guarantee that this parent solution is excluded from both resulting subdomains, defining a proper branching strategy to be employed in the course of the BB search process. In particular, one can analytically identify a point N on the circumference of disk D_{ij} such that the Euclidean distances between X and two resulting secant lines are equal, in the hope of producing a more balanced BB tree. Let θ^*_{ij} denote the angle in radians between the positive x_i-axis and the point N, then the resulting proposed split is $\left[\theta^L_{ij}, \theta^*_{ij}\right]$ and $\left[\theta^*_{ij}, \theta^U_{ij}\right]$ (Fig. 7.1d).

Given the above relaxation strategy, Eqs.(7.3)–(7.7) constitute a relaxed LP formulation to be solved at each node of our BB search process.

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad c^\top x \tag{7.3}$$

$$\text{subject to} \quad f_k(\tilde{x}) + \nabla f_k(\tilde{x})^\top (x - \tilde{x}) \leq 0 \qquad \forall \tilde{x} \in \mathcal{H}_k, \forall k \in \mathcal{K} \tag{7.4}$$

$$\alpha_{ijl} x_i + \beta_{ijl} x_j \geq \gamma_{ijl} \qquad \forall l \in \{1,2,3\}, \forall (i,j) \in \tilde{\mathcal{M}} \tag{7.5}$$

$$\phi_t^\top x \geq \xi_t \qquad \forall t \in \mathcal{T} \tag{7.6}$$

$$x^L \leq x \leq x^U \tag{7.7}$$

In the above formulation, constraints (7.4) constitute outer-approximation inequalities for $f_k(x) \leq 0, k \in \mathcal{K}$,[2] where $\tilde{x}$ constitute the points of linearization. The parameters α_{ijl}, β_{ijl} and γ_{ijl} in constraints (7.5) are suitably chosen in each case to define the linear inequalities

2 If $f_k(x)$ is an affine function, outer-approximation is not needed.

that relax the original non-overlapping constraints discussed above, while and the set $\tilde{\mathcal{M}} \subseteq \mathcal{M}$ denotes the set of circle pairs for which the corresponding θ_{ij} interval has been branched at least once. Constraints (7.6) generically denote strengthening cuts that we shall discuss in Section 7.4, while variable bounds are provided in constraints (7.7).

Note how, after a branch is applied, the parent node relaxation is tightened by adding elements to the set $\tilde{\mathcal{M}}$, when a new circle pair is branched upon for the first time, or by updating the coefficients $\alpha/\beta/\gamma$, when the circle pair is branched upon further. The relaxation is also tightened by appending and/or updating the set of strengthening cuts $\mathcal{T}$, while the outer-approximation sets $\mathcal{H}_k$ are also expanded as appropriate (see details later).

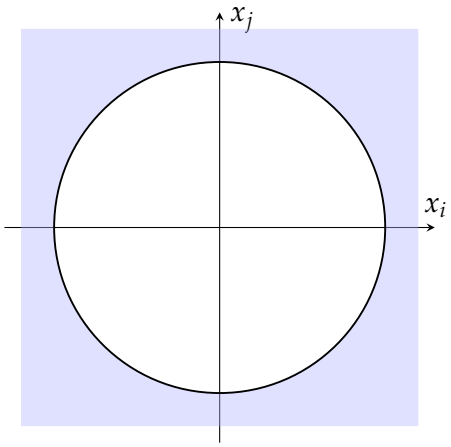
7.3.2 *The Branch-and-Bound Algorithm*

We shall now focus on the implementation of the customized BB algorithm. At the root node, the set $\mathcal{H}_k$ is initialized with points $wx^L + (1-w)\,x^U$, where $w \in \{0.25, 0.5, 0.75\}$, while the sets $\tilde{\mathcal{M}}$ and $\mathcal{T}$ begin as empty sets. All of them will be dynamically expanded and updated as the algorithm proceeds.

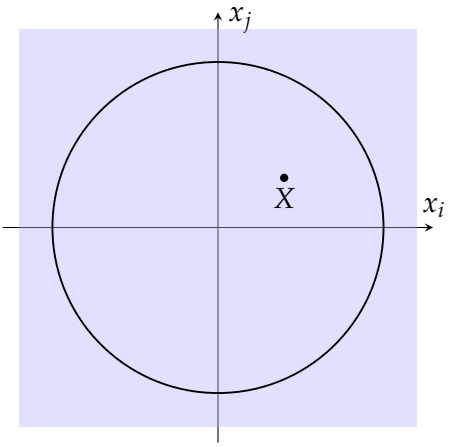
After solving the LP relaxation (7.3)–(7.7) at each node, we check whether each of the convex constraints (set $\mathcal{K}$) as well as each of the non-overlapping constraints (set $\mathcal{M}$) are satisfied within some predefined tolerance ε (we use $\varepsilon = 10^{-5}$) at the current LP solution $\bar{x}$.[3] We first focus on the convex constraints, and if $f_k(\bar{x}) > \varepsilon$ for some $k \in K$, we add an outer-approximation cut by setting $\mathcal{H}_k \leftarrow \mathcal{H}_k \cup \{\bar{x}\}$. Note that, at each iteration, we only add such cuts for the (up to) 5 most violated convex constraints k. If any outer-approximation cuts are added, we resolve the LP relaxation; otherwise, we proceed with checking the feasibility of the non-overlapping constraints, as follows. For each non-overlapping constraint $x_i^2 + x_i^2 \geq r_{ij}^2$, $(i,j) \in \mathcal{M}$, we define $V_{ij} := r_{ij} - \sqrt{\bar{x}_i^2 + \bar{x}_j^2}$, and we consider this constraint to be violated, if $V_{ij} > \varepsilon$. If no violated non-overlapping constraints exist, then a feasible solution $\bar{x}$ has been obtained; this solution is considered as a possible new incumbent, and the node is fathomed. However, if at least one non-overlapping constraint is violated, the node must be either tightened or branched, eliminating the current LP solution in either case.

Deferring the discussion of tightening techniques to Section 7.4, we now focus on our branching strategy, namely how we choose which implicit θ_{ij} variable (element of set $\mathcal{M} : \{V_{ij} > \varepsilon\}$) to branch upon. Recalling that further branching of a previously branched

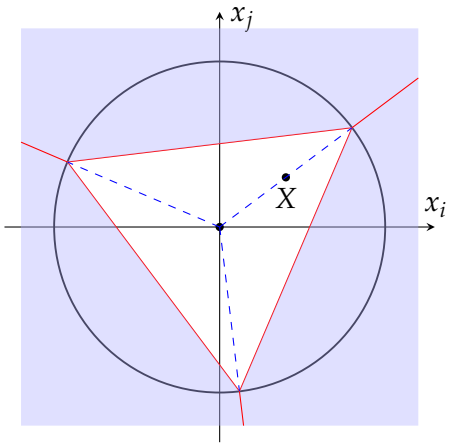
3 For simplicity of presentation, we use here a common tolerance ε for all constraints in problem (7.2), but we remark that in principle one may use a set of constraint-specific feasibility tolerances that are appropriately scaled for each constraint.



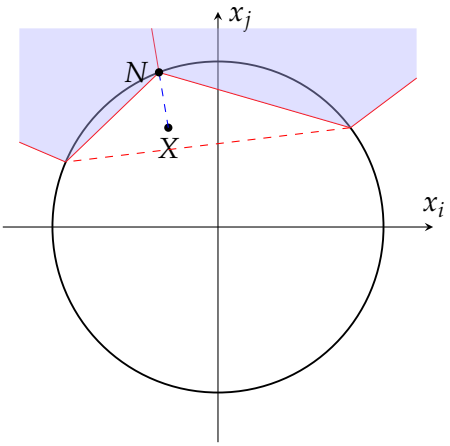
(a) Full feasible space



(b) Convex relaxation of full feasible space



(c) Initial tightening via three branches



(d) Subsequent tightening via two branches

Figure 7.1: The non-overlapping constraint and its dynamically tightened relaxation (adapted from Chapter 6).

θ_{ij} variable produces fewer child nodes in the BB tree than the initial branching (two versus three subdomains), the violation of currently unbranched non-overlapping constraints is first discounted by a factor of τ^{unbr} (we use $\tau^{unbr} = 0.7$); that is, $V_{ij} \leftarrow \left(1 - \tau^{unbr}\right) V_{ij}$, for all $(i,j) \in \mathcal{M} \setminus \tilde{\mathcal{M}}$. We then choose to branch on the implicit θ_{ij} variable that corresponds to the highest violation after such modifications. Note that, according to the earlier discussion, the branching is implemented either as updating existing left-hand side coefficients or by adding new constraints, depending on whether $(i,j) \in \tilde{\mathcal{M}}$ or not.

7.4 STRENGTHENING TECHNIQUES

We consider two types of relaxation strengthening techniques, namely *intersection cuts* and *feasibility-based tightening*. Even though intersection cuts [23] are well-known for their use in integer programming, their applicability goes much beyond that area, being generic tools to deal with non-convexities that may stem from either integrality or continuous nonlinear constraints. However, to the best of our knowledge, only a few works in the literature to date have attempted to apply these techniques in the context of continuous nonlinear optimization. In this work, we propose to utilize *strengthened* intersection cuts to tighten our LP relaxations. Additionally, various feasibility-based tightening techniques are also introduced.

7.4.1 *Intersection Cuts*

The idea of *intersection cuts* was originally proposed in [23] to construct valid cuts for integer programming. It has gradually received more and more attention in other areas such as reverse convex programming [91], bilevel optimization [75], polynomial programming [39], and factorable MINLP [152], among others. To the best of our knowledge, the literature always considers the case where variables are bounded from below by zero, while the more general case where variables are bounded from below and/or above by arbitrary values has been ignored. The challenge comes from the fact that, at the LP optimal solution, some non-basic variable might be at its upper bound, or some non-basic variable might be at its lower bound yet that bound not being 0. In such cases, the intersection cut formula from the seminal paper [23] is not immediately valid. Whereas this can be easily dealt with by explicitly adding to the model variable-bounding constraints (thus forcing all non-basic variables at an optimal LP solution to be slack variables), it is generally much more common and useful for solvers to separate variable bounds from other constraints

and to handle them separately. To this end, we present in Section 7.4.1.1 a complete derivation of intersection cuts for more general cases where variable bounds might be arbitrary numbers. Afterwards, in Section 7.4.1.3, we consider a strengthening method to further tighten these cuts.

7.4.1.1 *Generating Intersection Cuts*

Consider the following problem:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & x \in \mathcal{P} \cap \mathcal{Q}, \end{aligned} \tag{7.8}$$

where $\mathcal{P} := \left\{x \in \mathbb{R}^n : Ax \leq b, x^L \leq x \leq x^U\right\}$,[4] with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, while $\mathcal{Q}$ represents a "complicating" set. In this work, we consider $\mathcal{Q} := \left\{x \in \mathbb{R}^n : x_i^2 + x_j^2 \geq r_{ij}^2, \forall (i,j) \in \mathcal{M}\right\}$. Initially, we relax the feasible region by only considering $x \in \mathcal{P}$, and we introduce slack variables $s \in \mathbb{R}^m_{\geq 0}$ to obtain a linear program in standard form.

Let $t := (x, s)$ denote all variables, for convenience. Assume that the linear program is feasible and that its objective function is bounded; then, it can be solved via the simplex algorithm. Let $\bar{t} = (\bar{x}, \bar{s})$ represent the current LP optimal solution. Without loss of generality, we assume that $\bar{x} \notin \mathcal{Q}$, since otherwise problem (7.8) has been solved. For the time being, assume that we are given a convex set $\mathcal{S}$ whose interior contains $\bar{x}$ but no feasible point; that is, $\bar{x} \in int(\mathcal{S})$ and $int(\mathcal{S}) \cap (\mathcal{P} \cap \mathcal{Q}) = \emptyset$. In the following, we shall discuss how to generate a valid intersection cut to eliminate $\bar{x}$ using $\mathcal{S}$ for the case where structural variables x might be bounded from below and/or above.

Let $\mathcal{I}$ represent the index set of structural variables x. Let $\mathcal{B}$ and $\mathcal{N}$ denote the index sets associated with the basic and non-basic variables, respectively. Furthermore, let $\mathcal{N}^L$ denote the set of non-basic variables that are currently at their lower bounds and $\mathcal{N}^U := \mathcal{N} \setminus \mathcal{N}^L$ represent the set of non-basic variables at their upper bounds. Note that both structural and slack variables might be non-basic. We focus on the structural variables x and add a

4 Structural variables x are not necessarily bounded, i.e., $-\infty \leq x_i^L \leq x_i^U \leq +\infty$, $\forall\, i \in \{1, 2, ..., n\}$.

trivial relation for each non-basic variable x_j, $j \in \mathcal{I} \cap \mathcal{N}$. The modified simplex tableau is demonstrated in (7.9).

$$\begin{aligned} x_i = \bar{x}_i \quad & - \sum_{j \in \mathcal{N}^L} \bar{a}_{ij}(t_j - t_j^L) \quad + \sum_{j \in \mathcal{N}^U} \bar{a}_{ij}(t_j^U - t_j) \quad && \forall i \in \mathcal{I} \cap \mathcal{B} \\ x_j = \bar{x}_j \quad & -(-1)(x_j - x_j^L) && \forall j \in \mathcal{I} \cap \mathcal{N}^L \\ x_j = \bar{x}_j \quad & && +(-1)(x_j^U - x_j) \quad \forall j \in \mathcal{I} \cap \mathcal{N}^U \end{aligned} \tag{7.9}$$

For convenience, tableau (7.9) can also be represented in vector form, where $\bar{a}_j$ are suitably defined vectors:

$$x = \bar{x} - \sum_{j \in \mathcal{N}^L} \bar{a}_j(t_j - t_j^L) + \sum_{j \in \mathcal{N}^U} \bar{a}_j(t_j^U - t_j). \tag{7.10}$$

Associated with the simplex tableau (7.10) is a conic relaxation of $\mathcal{P}$, called the LP-cone, whose apex is $\bar{x}$ and whose facets are defined by n hyperplanes that form the basis of $\bar{x}$. Thus, the LP-cone has n extreme rays, each formed by an extreme direction r^j emanating from the apex $\bar{x}$, where $r^j = -\bar{a}_j$ if $j \in \mathcal{N}^L$, or $r^j = \bar{a}_j$ if $j \in \mathcal{N}^U$. Following the extreme ray $\bar{x} + \lambda_j r^j$ (where $\lambda_j \geq 0$), we seek its intersecting point with the boundary of the set $\mathcal{S}$.[5] More specifically, for each extreme ray $\bar{x} + \lambda_j r^j$, $j \in \mathcal{N}$, we seek to solve

$$\lambda_j^* := \underset{\lambda_j \geq 0}{\text{maximize}} \left\{ \lambda_j : \bar{x} + \lambda_j r^j \in \mathcal{S} \right\}. \tag{7.11}$$

Since $\bar{x} \in int(\mathcal{S})$, problem (7.11) is always feasible (e.g., $\lambda_j = 0$ is a feasible solution). If r^j is a recessive direction of $\mathcal{S}$ (i.e., $r^j \in rec(\mathcal{S})$, where $rec(\mathcal{S})$ denotes the recession cone of $\mathcal{S}$), there is no intersection point between the extreme ray $\bar{x} + \lambda_j r^j$ and the boundary of the set $\mathcal{S}$. Thus, in this case, the objective in (7.11) is unbounded from above and we set $\lambda_j^* = +\infty$. Otherwise, we obtain a unique solution λ_j^* and the intersection point is $\bar{x} + \lambda_j^* r^j$.

Let $\mathcal{N}^1 := \left\{ j \in \mathcal{N} : r^j \in rec(\mathcal{S}) \right\}$ and $\mathcal{N}^2 := \mathcal{N} \setminus \mathcal{N}^1$ for convenience. The intersection cut that cuts off $\bar{x}$ is defined to be the halfspace whose boundary contains each intersection point $\bar{x} + \lambda_j^* r^j$, $j \in \mathcal{N}^2$ and which is parallel to each extreme direction r^j, $j \in \mathcal{N}^1$ (Fig. 7.2). The validity of intersection cuts is intuitive from a geometric perspective. Thus, we skip its proof and instead focus to the derivation of the intersection cut formula.

Proposition 7.1. *The intersection cut can be represented in non-basic space (e.g., using non-basic variables t_j) as follows:*

$$\sum_{j \in \mathcal{N}^L} \frac{t_j - t_j^L}{\lambda_j^*} + \sum_{j \in \mathcal{N}^U} \frac{t_j^U - t_j}{\lambda_j^*} \geq 1. \tag{7.12}$$

5 We assume that $\mathcal{S}$ is a closed set.

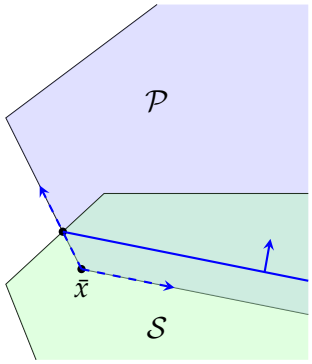


Figure 7.2: Generation of intersection cuts

Proof. We only need to prove that this halfspace passes through all intersection points $\bar{x} + \lambda_j^* r^j$, $j \in \mathcal{N}^2$ and that it is parallel to extreme directions r^j, $j \in \mathcal{N}^1$. We consider the following two cases.

- If $j \in \mathcal{N}^2$, the intersection point $\bar{x} + \lambda_j^* r^j$ is obtained via moving from $\bar{x}$ along the direction r^j by a distance of λ_j^*. If $j \in \mathcal{N}^2 \cap \mathcal{N}^L$, in the non-basic space we have $t_j = t_j^L + \lambda_j^* \cdot 1$, and the other non-basic variables $t_{j'}$ ($j' \in \mathcal{N} \setminus \{j\}$) will remain at their lower or upper bounds at this intersection point. Therefore,

$$\sum_{j' \in \mathcal{N}^L} \frac{t_{j'} - t_{j'}^L}{\lambda_{j'}^*} + \sum_{j' \in \mathcal{N}^U} \frac{t_{j'}^U - t_{j'}}{\lambda_{j'}^*} = \frac{t_j^L + \lambda_j^* - t_j^L}{\lambda_j^*} = 1.$$

  In other words, the intersection cut passes through the intersection point $\bar{x} + \lambda_j^* r^j$. This also applies in the case $j \in \mathcal{N}^2 \cap \mathcal{N}^U$.

- If $j \in \mathcal{N}^1$, we only need to prove that the intersection cut is parallel to r^j. Moving along the direction r^j from the point $\bar{x}$, if $j \in \mathcal{N}^1 \cap \mathcal{N}^L$, we have $t_j = t_j^L + \lambda \cdot 1$ (where $\lambda \geq 0$), and the other non-basic variables $t_{j'}$ ($j' \in \mathcal{N} \setminus \{j\}$) will remain unchanged. Therefore,

$$\sum_{j' \in \mathcal{N}^L} \frac{t_{j'} - t_{j'}^L}{\lambda_{j'}^*} + \sum_{j' \in \mathcal{N}^U} \frac{t_{j'}^U - t_{j'}}{\lambda_{j'}^*} = \frac{t_j^L + \lambda - t_j^L}{\lambda_j^*} = \frac{\lambda}{+\infty} = 0.$$

  The parallel relationship is proved. This also applies in the case $j \in \mathcal{N}^1 \cap \mathcal{N}^U$.

□

As mentioned by [23], λ_j^* should be approximated from below for numerical validity. Readers are referred to [23] for further details, with the observation however that the

intersection cut formula (7.12) is more generic than the one presented in [23], reducing to the latter when structural variables x are only bounded from below by zero.

Note that, given a mathematical model, one has to identify a suitable set $\mathcal{S}$ to derive valid intersection cuts. We highlight that such a convex set $\mathcal{S}$ generally exists, if some constraint in the model imposes a reverse convex region. In fact, from problem (7.11), one can infer that deeper cuts will be produced from larger sets $\mathcal{S}$ whose boundary intersects with extreme rays at intersections points further away from the apex, $\bar{x}$.

7.4.1.2 *Generating Intersection Cuts for Non-overlapping Constraints*

In our context, we can easily identify a valid sets $\mathcal{S}$ for generating intersection cuts. Considering the fact that the non-overlapping constraint $x_i^2 + x_j^2 \geq r_{ij}^2$ represents a reverse convex region [91], we can define $\mathcal{S}_{ij} := \{x \in \mathbb{R}^n : x_i^2 + x_j^2 \leq r_{ij}^2\}$ (Fig. 7.3a) such that $\bar{x} \in int(\mathcal{S}_{ij})$. Furthermore, recalling that enlarging the set whenever possible leads to stronger cuts, we can enlarge S_{ij} using the following formula (Fig. 7.3b) when $\theta_{ij}^U - \theta_{ij}^L < 2\pi$:

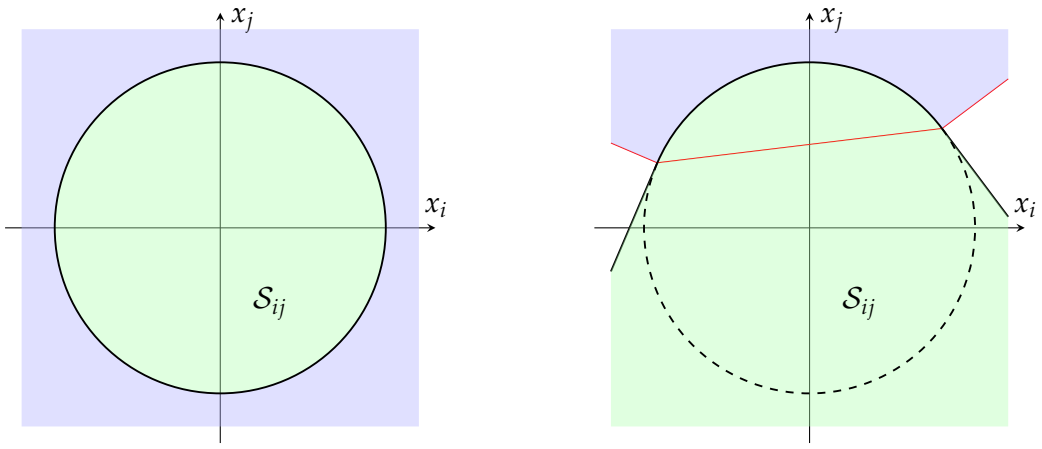
$$\mathcal{S}_{ij} \leftarrow \mathcal{S}_{ij} \cup \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \cos(\theta)x_i + \sin(\theta)x_j \leq r_{ij}, \quad \forall \theta \in \{\theta_{ij}^L, \theta_{ij}^U\} \\ \cos(\frac{\theta_{ij}^L + \theta_{ij}^U}{2})x_i + \sin(\frac{\theta_{ij}^L + \theta_{ij}^U}{2})x_j \leq r_{ij}\cos(\frac{\theta_{ij}^U - \theta_{ij}^L}{2}) \end{array} \right\}. \tag{7.13}$$

The enlarged set $\mathcal{S}_{ij}$ remains convex and does not contain any feasible solution in its interior; therefore, intersection cuts generated from $\mathcal{S}_{ij}$ will be valid. We highlight that enlarging the convex set and generating stronger intersection cuts becomes possible due to the chosen scheme to branch upon non-overlapping constraints.

In our implementation, we define $\mathcal{S}_{ij}$ for every $(i,j) \in \mathcal{M}$. When $\theta_{ij}^U - \theta_{ij}^L < 2\pi$, we use the enlarged version (7.13). We use these sets to generate intersection cuts for which the corresponding non-overlapping constraint is violated, but only the best 5 cuts with Euclidean distances from $\bar{x}$ larger than tolerance $\delta = 10^{-2}$ are added to the model. Furthermore, whenever θ_{ij}^L or θ_{ij}^U is updated due to branching or tightening, we always enlarge the set $\mathcal{S}_{ij}$ and attempt to compute new λ_j^* values from (7.12) towards strengthening the related intersection cuts that have already been generated.

7.4.1.3 *Strengthening Intersection Cuts*

Fig. 7.4 demonstrates a simple case where the intersection cut (blue solid line) generated from Section 7.4.1.1 can be further tightened. The generated intersection cut passes through an intersection point and is parallel to the extreme direction r^j (since $r^j \in rec(\mathcal{S})$). In this case, we can identify another cut $\widehat{IC}$ (red solid line), which also passes through the same



(a) $\theta_{ij}^U - \theta_{ij}^L = 2\pi$ (b) $\theta_{ij}^U - \theta_{ij}^L < 2\pi$

Figure 7.3: Valid convex sets $\mathcal{S}_{ij}$ for generating intersection cuts

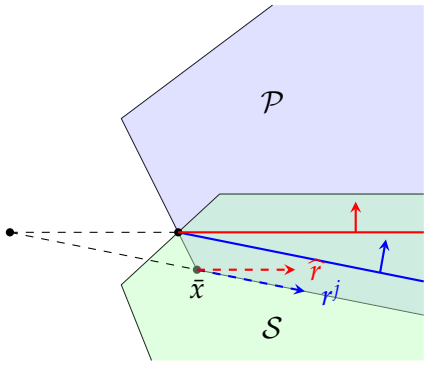


Figure 7.4: Strengthened intersection cuts

intersection point, but which is parallel to some other recessive direction $\hat{r} \in rec(\mathcal{S})$. Apparently, $\widehat{IC}$ is a valid cut that dominates IC.

The validity of $\widehat{IC}$ deduces from the fact that, by convexity of $\mathcal{S}$, any point satisfying IC but not $\widehat{IC}$ lies within the interior of $\mathcal{S}$ and is thus outside $\mathcal{P} \cap \mathcal{Q}$. The stronger cut $\widehat{IC}$ can be regarded as a halfspace that passes through the existing intersection point and a new one residing at the negative part of the extreme ray $\bar{x} + \lambda r^j$ (where $\lambda \geq 0$). In order to maintain the validity of the new intersection cut, as the work of [39] suggested, one has to guarantee that $\hat{r} := (\bar{x} + \lambda_{j'}^* r^{j'}) - (\bar{x} + \lambda_j r^j)$, the direction from the new intersection point, $(\bar{x} + \lambda_j r^j)$, to every intersection point, $(\bar{x} + \lambda_{j'}^* r^{j'})$, is a recessive direction of $\mathcal{S}$ (i.e., $\hat{r} \in rec(\mathcal{S})$) for every $j' \in \mathcal{N}^2$. It is clear from (7.12) that increasing λ_j^* while $\lambda_j^* \leq 0$ leads

to a stronger cut. Thus, for each extreme direction $r^j, j \in \mathcal{N}^1$, we define the following problem:

$$\lambda_j^* := \underset{\lambda_j \leq 0}{\text{maximize}} \left\{ \lambda_j : \lambda_{j'}^* r^{j'} - \lambda_j r^j \in rec(\mathcal{S}), \forall j' \in \mathcal{N}^2 \right\}. \tag{7.14}$$

If problem (7.14) is infeasible, we set $\lambda_j^* = -\infty$;[6] otherwise, a unique solution λ_j^* is obtained, resulting in a new intersection point $(\bar{x} + \lambda_j^* r^j)$. With this, we arrive at Proposition 7.2.

Proposition 7.2. *The intersection cut defined by (7.12) and using λ_j^* from (7.11), for $j \in \mathcal{N}^2$, and λ_j^* from (7.14), for $j \in \mathcal{N}^1$, is valid.*

Given a convex set $\mathcal{S}$, the problem (7.14) is in general hard to solve because no closed-form formula for $rec(\mathcal{S})$ is available. However, in our context, the recession cone $rec(\mathcal{S}_{ij})$ can be easily identified. More specifically, we distinguish three cases:

(i) when $\theta_{ij}^U - \theta_{ij}^L > \pi$, the projection of $\mathcal{S}_{ij}$ onto the x_i–x_j space is a bounded area, and hence its recession cone is

$$rec(\mathcal{S}_{ij}) = \left\{ x \in \mathbb{R}^n : x_i = 0, x_j = 0 \right\},$$

projecting to a singleton on the x_i–x_j space;

(ii) when $\theta_{ij}^U - \theta_{ij}^L = \pi$, the recession cone becomes[7]

$$rec(\mathcal{S}_{ij}) = \left\{ x \in \mathbb{R}^n : x_i = -\lambda \sin(\theta_{ij}^L), x_j = \lambda \cos(\theta_{ij}^L), \forall \lambda \geq 0 \right\},$$

projecting to a ray on the x_i–x_j space;

(iii) when $\theta_{ij}^U - \theta_{ij}^L < \pi$, it is easy to show[8] that

$$\begin{aligned} rec(\mathcal{S}_{ij}) &= rec(\left\{ x \in \mathbb{R}^n : \cos(\theta) x_i + \sin(\theta) x_j \leq r_{ij}, \forall \theta \in \{\theta_{ij}^L, \theta_{ij}^U\} \right\}) \\ &= \left\{ x \in \mathbb{R}^n : \cos(\theta) x_i + \sin(\theta) x_j \leq 0, \forall \theta \in \left\{ \theta_{ij}^L, \theta_{ij}^U \right\} \right\}. \end{aligned} \tag{7.15}$$

We remark that, when case (i) applies, the problem (7.14) is almost always infeasible, due to insufficient degrees of freedom. Furthermore, case (ii) is unlikely to be relevant in the context of double-precision arithmetic, due to inability to detect that $\theta_{ij}^U - \theta_{ij}^L$ equals π exactly, and hence, we did not consider this case in our implementation. It is only under

6 This is equivalent to setting $\lambda_j^* = +\infty$.

7 This deduces from directly substituting $\theta_{ij}^U \leftarrow \theta_{ij}^L + \pi$ in (7.13).

8 The recession cone of a polyhedral set $\mathcal{W} := \{x \in \mathbb{R}^n : Ax \leq b\}$ is $rec(\mathcal{W}) = \{x \in \mathbb{R}^n : Ax \leq 0\}$.

case (iii) that the closed-form formula (7.15) can be plugged into (7.14) and a value for λ_j^* be analytically identified for every $j \in \mathcal{N}^1$. Therefore, in our implementation, we generate intersection cuts using Proposition 7.1 in cases (i) and (ii), and we attempt to generate a strengthened version of such cuts using Proposition 7.2 only in case (iii).

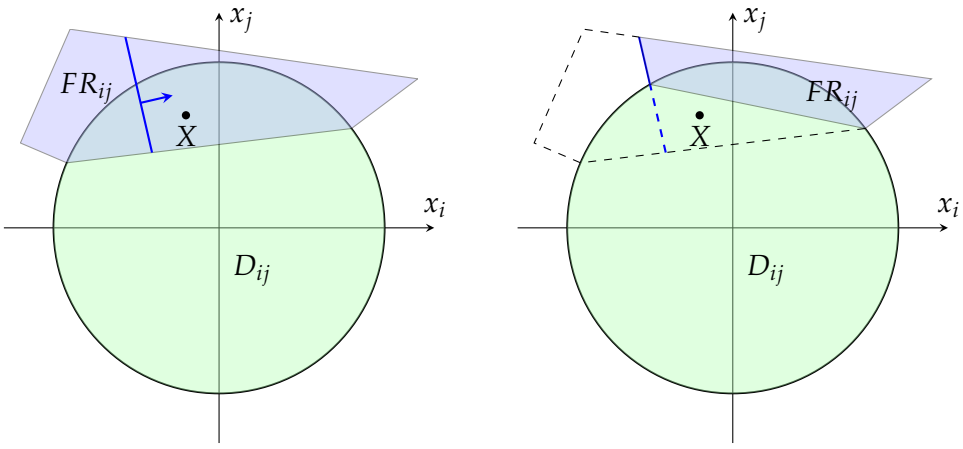
7.4.2 *Feasibility-based Tightening*

Feasibility-based tightening is a common technique in global optimization to reduce variable bounds, and it is usually implemented via interval arithmetic [139]. In our context, since explicit structural variable domains are not branched upon, the interval arithmetic technique would not tighten variable bounds effectively. Considering that our strategy is to branch on the feasible intervals of implicit variables θ_{ij}, our feasibility-based tightening approach will instead focus on reducing the domains for these variables.

From a geometric perspective, the non-overlapping constraint, $x_i^2 + x_j^2 \geq r_{ij}^2$, requires the solution's projection on the x_i–x_j space to lie either outside or exactly on the circumference of disk D_{ij} (Fig. 7.1a). Let FR_{ij} denote the feasible region of the x_i–x_j space. At the root node of the BB tree, we have $FR_{ij} = conv\left(\left\{(x_i, x_j) : x_i^L \leq x_i \leq x_i^U, x_j^L \leq x_j \leq x_j^U\right\} \setminus int(D_{ij})\right)$, and this region gradually reduces due to branching or tightening. We emphasize that the feasible spaces FR_{ij}, $(i,j) \in \mathcal{M}$ are not explicitly enforced in the relaxation model, (7.3)–(7.7), because they are generally implied by other constraints. We also note that FR_{ij} is always a bounded polyhedron,[9] and that this region might correspond to a reduced interval $\left[\theta_{ij}^L, \theta_{ij}^U\right]$ for the implied variable θ_{ij}. To that end, in our implementation, we attach our description of FR_{ij} to the BB node as important information to be used for tightening the θ_{ij} intervals, as follows.

Assume that the projection of the current LP solution $\bar{x}$ lies in the interior of D_{ij}, i.e., $X \in FR_{ij} \cap int(D_{ij})$. Whenever FR_{ij} is reduced due to some tightening technique, we can revisit the geometric meaning of the non-overlapping constraint and obtain $FR_{ij} \leftarrow conv\left(FR_{ij} \setminus int(D_{ij})\right)$ (Fig. 7.5). From that, we can infer possibly tighter $\left[\theta_{ij}^L, \theta_{ij}^U\right]$ bounds, and whenever the latter are indeed tightened, we first update the relaxations for its corresponding non-overlapping constraint and then check whether X is cut off by the halfspace formed by the new secant line. Next, we discuss three opportunities to reducing FR_{ij}.

9 We assume that valid lower and upper bounds for x_i, $i \in \{1, 2, ...n\}$ can be extracted or deduced.



(a) Initial feasible region (b) Tightened feasible region

Figure 7.5: Feasibility-based tightening

7.4.2.1 *Generating Concave Envelopes*

Let us begin by defining $h(x_i, x_j) := x_i^2 + x_j^2$, where $(x_i, x_j) \in FR_{ij}$. We seek the *concave envelope*, i.e., the tightest possible over-estimator, of $h(x_i, x_j)$ over its domain FR_{ij}, which we shall denote as $conc_{FR_{ij}} h(x_i, x_j)$. It is clear that the function $h(x_i, x_j)$ has a *vertex polyhedral* concave envelope over the bounded polyhedral domain FR_{ij} [165]; therefore, its concave envelope coincides with the concave envelope of its restriction to the vertices of FR_{ij} and consists of finitely many hyperplanes. [10]

As a result, we obtain the following constraints that are valid for our relaxation:

$$conc_{FR_{ij}} h(x_i, x_j) \geq r_{ij}^2. \tag{7.16}$$

We remark that constraints (7.16) are not directly added to the relaxation model, (7.3)–(7.7), rather used to reduce FR_{ij}. More specifically, $FR_{ij} \leftarrow FR_{ij} \cap \left\{(x_i, x_j) : conc_{FR_{ij}} h(x_i, x_j) \geq r_{ij}^2\right\}$. If FR_{ij} is indeed reduced in this manner, we then attempt to apply feasibility-based tightening as described in the preamble of this section to further tighten FR_{ij}. Note that this process can be applied recursively, since a smaller polytope FR_{ij} will induce a tighter over-estimator for $h(x_i, x_j)$, which might in turn further reduce FR_{ij}.

10 From our computational experience, the number of vertices of FR_{ij} is usually small (around 10), and thus identifying its concave envelope via enumeration is computationally efficient. Regardless, if the number of vertices of FR_{ij} is large, one can always properly relax the domain FR_{ij} and obtain a new bounded polyhedron that contains FR_{ij} and has a small number of vertices.

7.4.2.2 *Calculating Minkowski Sums*

Another typical feasibility-based tightening technique in global optimization is *bound propagation* [139]. Adapting this idea into our context, we propose to apply *domain propagation*. For example, in the context of packing three circles i, j and k, one usually seeks to enforce non-overlapping constraints in a pairwise sense. In this case, any feasible point $(a_i - a_j, b_i - b_j)$ satisfying the non-overlapping constraint $(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2$ must correspond to a point $(a_i - a_k, b_i - b_k)$ in the domain FR_{ik} as well as to a point $(a_k - a_j, b_k - b_j)$ in the domain FR_{kj}. This results from the simple observation that

$$(a_i - a_j, b_i - b_j) = (a_i - a_k, b_i - b_k) + (a_k - a_j, b_k - b_j). \tag{7.17}$$

Consequently, one can calculate the Minkowski sum of FR_{ik} and FR_{kj} to tighten FR_{ij}; that is, $FR_{ij} \leftarrow FR_{ij} \cap \left(FR_{ik} \oplus FR_{kj}\right)$. If FR_{ij} is indeed reduced, the feasibility-based tightening $FR_{ij} \leftarrow conv\left(FR_{ij} \setminus int(D_{ij})\right)$ will be considered. Note that the Minkowski sum of two polytopes with n_1 and n_2 vertices, respectively, can be computed in $O(n_1 + n_2)$ time [36]; thus, the proposed procedure is computationally efficient.

7.4.2.3 *Solving LPs*

We can also refine feasible region FR_{ij} by using the projection of the feasible region of the relaxation model, (7.3)–(7.7), onto the x_i–x_j space. Considering that computing the exact projection is not practical, we choose to outer-approximate it via linear inequalities defined in the x_i–x_j space. More specifically, one can solve an LP with the objective of minimizing some linear function, e.g., $\mu x_i + \nu x_j$, and with constraints (7.4)–(7.7), where μ and ν are properly chosen coefficients. Let σ be its optimal value, thus $\mu x_i + \nu x_j \geq \sigma$ is a valid inequality to characterize the projection area and can be used to refine FR_{ij}; that is, $FR_{ij} \leftarrow FR_{ij} \cap \left\{(x_i, x_j) : \mu x_i + \nu x_j \geq \sigma\right\}$. If FR_{ij} is successfully reduced in this manner, we then apply feasibility-based tightening as before.

In our context, we apply the above process for two separate objective functions, namely those that are parallel to the two linear boundaries of FR_{ij} that neighbor the secant line. These were specifically chosen due to their potential to immediately tighten the secant line, which might thus help to eliminate the current LP solution, $\bar{x}$.

7.4.3 *Implementation Details*

We note the following details about our implementation.

- With regards to the protocol via which we employ the various tightening techniques, we note the following. We first apply intersection cuts, followed by calculating Mikowski sums, and finally by solving LPs. Tightening from concave envelopes is swiftly applied as soon as FR_{ij} is tightened during the latter two steps, and whenever $\bar{x}$ is cut off, we skip the subsequent routines and turn our attention to resolving the LP relaxation (7.3)–(7.7).

- Since the intersection cuts are formulated in non-basic space, and since we do not have access to non-basic slack variables when enforcing a constraint in the LP solver, we first convert it to the structural space. After this step, we apply some presolve reductions to mitigate numerical difficulties [4]. In particular, we first scale the coefficients in a cut such that their largest absolute value is 1.0, and we then perform proper reductions on small coefficients (e.g., less than 10^{-4}) while maintaining the cut's validity.

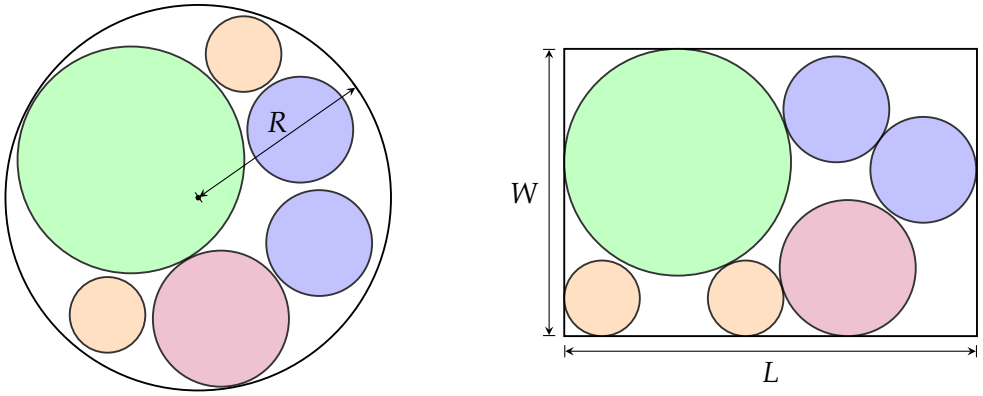
- In order to alleviate tapering off from using intersection cuts, we stop the separation if the gap is decreased by less than 0.01% a total of 3 times.

- Outer-approximation and intersection cuts are removed from the LP model (7.3)–(7.7), if they are not active for the past 50 LP-solving rounds. We note that removed intersection cuts are still stored in the cut pool and will be strengthened when the corresponding convex sets are enlarged, in the hope that they might be added back to the set $\mathcal{T}$ if violated at some future point.

- Whenever FR_{ij} is reduced at some BB node, we recursively call this routine to further reduce the feasible region, and this continues until the reduction in area of FR_{ij} is less than 1%. Moreover, whenever the area of FR_{ij} is reduced by at least 5%, we attempt to apply domain propagation to tighten relevant domains. However, since solving LPs can add up to the overall computational time, we only consider the 10 pairs $(i, j) \in \mathcal{M}$ with the largest violation values, V_{ij}.

7.5 COMPUTATIONAL STUDIES

In this section, we test our customized branch-and-bound approach and compare its performance against the state-of-the-art general purpose global optimization solvers, namely ANTIGONE 1.1 [125], BARON 19.7.13 [166], COUENNE 0.5 [32], LINDOGLOBAL 12.0 and SCIP 6.0 [3]. Our algorithm was implemented in C++ and the LP relaxation models



(a) Packing circles into a circle (b) Packing circles into a rectangle

Figure 7.6: Two variants of circle packing problems

were solved via CPLEX Optimization Studio 12.8.0 through the C application programming interface. Global solvers were called within the GAMS 28.2.0 environment. The absolute optimality tolerance was set to be 10^{-4}. All computational experiments were conducted on a single thread of an Intel Xeon CPU E5-2689 v4 @ 3.10GHz with 32GB of RAM.

7.5.1 *Circle Packing*

7.5.1.1 *Problem Definitions*

Let $\mathcal{C} = \{1, 2, 3...\}$ denote a set of circles. For each circle $i \in \mathcal{C}$, let r_i denote its radius. Without loss of generality, we assume a circle ordering such that $r_i \geq r_j$, when $i < j$, for all $i, j \in \mathcal{C}$. We aim to identify a feasible configuration of these circles within (i) a larger circle, and (ii) a sheet of fixed width W, such that no circles overlap and the size of the circle (radius R) or the sheet (length L) is minimized. Following literature practice, we refer to these settings as *packing circles into a circle* and *packing circles into a rectangle*, respectively.

7.5.1.2 *Mathematical Modeling*

Let (a_i, b_i) denote the final coordinates for the center of circle $i \in C$. Assuming that the center of the containing circle is fixed to the point $(0, 0)$, the problem of packing circles into a circle can be formulated as the non-convex model (7.18)–(7.20).

$$\underset{a_i, b_i, R}{\text{minimize}} \quad R \tag{7.18}$$

$$\text{subject to} \quad \sqrt{a_i^2 + b_i^2} \leq R - r_i \qquad \forall i \in \mathcal{C} \tag{7.19}$$

$$(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2 \qquad \forall (i, j) \in \mathcal{C} \times \mathcal{C} : i < j \tag{7.20}$$

Constraints (7.19), which are convex, enforce that all circles lie within the circular container,[11] while the reverse convex constraints (7.20) guarantee that there is no overlap among circles. Additionally, as also pointed out in [149], two types of symmetry-breaking constraints can be added to the above model:

i. Identical circles: When two circles i and j, $i < j$, are identical (i.e., $r_i = r_j$), then $a_i \leq a_j$ breaks this symmetry. [12]

ii. Rotational symmetry and reflection: Rotational symmetry arises due to the fact that an equivalent packing solution can always be obtained via rotating the entire configuration by any angle; hence, one can enforce that $a_1 \leq 0$ and $b_1 = 0$. Furthermore, the reflection of a feasible configuration through the horizontal axis results in an equivalent solution; hence, one can further impose $b_2 \geq 0$.

The mathematical formulation for packing circles into a rectangle is similar. Assuming that the bottom left corner of the rectangle is fixed to the point $(0, 0)$, the model (7.21) – (7.24) applies.

$$\underset{a_i, b_i, L}{\text{minimize}} \quad L \tag{7.21}$$

$$\text{subject to} \quad r_i \leq a_i \leq L - r_i \qquad \forall i \in \mathcal{C} \tag{7.22}$$

$$r_i \leq b_i \leq W - r_i \qquad \forall i \in \mathcal{C} \tag{7.23}$$

$$(a_i - a_j)^2 + (b_i - b_j)^2 \geq (r_i + r_j)^2 \qquad \forall (i, j) \in \mathcal{C} \times \mathcal{C} : i < j \tag{7.24}$$

11 When using the solver SCIP, we instead pass $a_i^2 + b_i^2 \leq (R - r_i)^2$, since this version can be recognized as a *second order cone* constraint [172].

12 In our customized approach, instead of adding this constraint, we tighten $\left[\theta_{ij}^L, \theta_{ij}^U\right] = [\pi/2, 3\pi/2]$ at the root node.

In this case, applicable constraints to break symmetry from reflection are: (i) $a_1 \leq L/2$, and (ii) $b_1 \leq W/2$.

7.5.1.3 *Benchmark Instances*

In order to evaluate the performance of our approach, we consider instances of packing circles into a circle from [1]. Each instance is defined by two numbers, p and N, and the size of circles to be packed in every instance is defined by $r_i = i^p$, where $i = 1, 2, ...N$. We consider $N \in \{5, 6, ..., 11\}$ and $p \in \{1, 1/2, -1/2, -2/3, -1/5\}$; thus, in total we have 35 instances of difference sizes.

For packing circles into a rectangle, we generate a suite of instances using data from the thirty-circle example in [157]. We consider $N \in \{6, 7, ..., 10\}$, and using the first N circles in each case, we generate all possible packing instances with $N - 1$ circles. Therefore, we generate a total of 40 instances, namely 6 five-circle, 7 six-circle, 8 seven-circle, 9 eight-circle and 10 nine-circle instances. The rectangle width W is chosen to be 9.5, as per the literature reference [157].

7.5.2 *Computational Results*

The goals of our computational study are to (i) assess the effect of strengthened intersection cuts and feasibility-based tightening on the BB tree, and (ii) conduct a comprehensive comparison between our proposed algorithm and state-of-the-art global solvers. These two goals are covered in Sections 7.5.2.1 and 7.5.2.2, respectively. To ensure a fair comparison, all algorithms (including the global solvers) were initialized with heuristic solutions, which were obtained after running BARON heuristics with a time limit of 1 hour.

7.5.2.1 *Effect of Node Relaxation Tightening*

We first analyze the effect of using strengthened intersection cuts on the branch-and-bound algorithm. We then enable our feasibility-based tightening techniques and assess the additional tractability gains from doing so. Overall, we consider three versions of our algorithm: (i) the rudimentary branch-and-bound algorithm (denoted by "BB"); (ii) "BB" enhanced with strengthened intersection cuts (denoted by "BB+SIC"); (iii) "BB+SIC" enhanced with feasibility-based tightening (denoted by "BB+SIC+FBT"). For each version, we adopt the best-bound first search node selection strategy during the branch-and-bound process.

Tables 7.1 and 7.2 present the computational results for packing circles into a circle and a rectangle, respectively. The first two columns in these tables list the input size (number of circles) and the number of instances of a given input size, for a respective total of 35 and 40 instances, as described above. The tables also present the number of instances that were solved to provable optimality within a given time limit of 1 hour, as well as the geometric means of solution time and number of branch-and-bound nodes explored; for the remaining instances for which optimality could not be proven, we present the average residual gap, defined as $(UB - LB)/UB$, at the time limit.

Table 7.1 shows that BB could solve 20 out of 35 instances optimally, while both BB+SIC and BB+SIC+FBT solved 23 and 26 of them to optimality, respectively, including one of the largest instances featuring the packing of eleven circles. Putting aside the fact that it allowed us to solve three more instances within the allotted time limit, the addition of strengthened intersection cuts resulted in a noticeable improvement in average solution times, number of nodes explored and residual gaps (when applicable). The BB+SIC+FBT version performed even better in terms of these metrics, especially for instances of larger input size, such as instances with ten or eleven circles.

Turning our attention to Table 7.2, we observe that all three variants can easily address instances featuring up to seven circles. The BB+SIC improved upon the baseline approach in terms of being able to prove optimality within the allotted time limit for a number of the eight-circle instances, with the BB+SIC+FBT approach solving the majority of them while also demonstrating a noticeable reduction in the average solution time to do so. Instances with nine circles remained elusive to all methods, with a significant average residual gap above 15%.

Considering all instances across both datasets, we conclude that the utilization of both strengthened intersection cuts and feasibility-based tightening has an overall positive impact on performance of the algorithm. Therefore, we adopt approach BB+SIC+FBT in the remainder of our computational studies.

7.5.2.2 *Comparison with Global Optimization Solvers*

In order to evaluate the competitiveness of our proposed approach against general-purpose global optimization solvers, we solve every benchmark instance independently with each of five state-of-the-art solvers, namely ANTIGONE, BARON, COUENNE, LINDOGLOBAL and SCIP, imposing the same time limit of 1 hour. The results are presented in Tables 7.3 and Table 7.4, using similar format as before.

Table 7.1: Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a circle

# circles	# inst.	BB				BB + SIC				BB + SIC + FBT			
		# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)
5	5	5	0.3	903	–	5	0.3	408	–	5	0.2	17	–
6	5	5	3.6	9,388	–	5	2.3	2,657	–	5	0.6	56	–
7	5	5	23	45,175	–	5	13	11,077	–	5	2.0	213	–
8	5	3	149	183,638	5.4	4	68	52,731	5.2	4	17	2,622	3.1
9	5	1	122	202,293	6.7	2	249	160,174	5.3	4	223	22,536	12.1
10	5	1	222	323,408	10.0	1	194	118,791	7.7	2	120	11,901	9.2
11	5	0	–	–	12.0	1	2,489	762,566	10.6	1	967	158,403	9.9

Table 7.2: Effect of strengthened intersection cuts and feasibility-based tightening on solving instances of packing circles into a rectangle

# circles	# inst.	BB				BB + SIC				BB + SIC + FBT			
		# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)	# solved	Time (sec)	# nodes	Gap (%)
5	6	6	0.1	143	–	6	0.1	95	–	6	0.1	28	–
6	7	7	0.2	874	–	7	0.2	252	–	7	0.2	69	–
7	8	8	2.6	21,049	–	8	3.5	3,636	–	8	2.7	693	–
8	9	0	–	–	10.8	3	3,052	770,170	5.1	7	1,264	126,822	9.6
9	10	0	–	–	23.6	0	–	–	15.9	0	–	–	16.5

Table 7.3: Computational results for global solvers on solving instances of packing circles into a circle

# circles	# inst.	ANTIGONE			BARON			COUENNE			LINDOGLOBAL			SCIP		
		# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)
5	5	5	0.3	–	5	0.2	–	5	0.6	–	5	0.3	–	5	6.5	–
6	5	4	0.5	3.8	5	1.0	–	5	33	–	5	2.0	–	4	34	4.2
7	5	3	0.1	5.3	5	14	–	5	896	–	5	18	–	3	7.0	4.9
8	5	1	55	12.5	4	501	25.5	1	127	10.9	4	627	26.5	1	270	7.9
9	5	0	–	14.6	1	1,741	26.4	0	–	16.6	1	2,351	27.9	0	–	9.5
10	5	0	–	17.4	0	–	23.0	0	–	20.6	0	–	30.4	0	–	12.3
11	5	0	–	20.4	0	–	33.8	0	–	24.3	0	–	38.8	0	–	13.5

Table 7.4: Computational results for global solvers on solving instances of packing circles into a rectangle

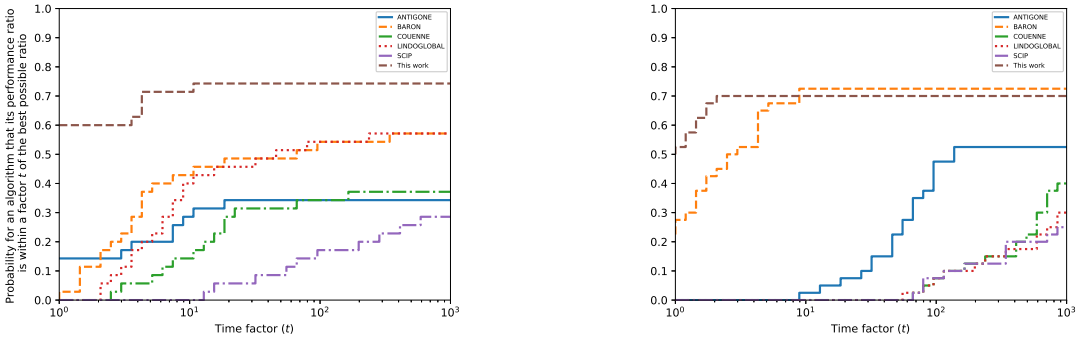
# circles	# inst.	ANTIGONE			BARON			COUENNE			LINDOGLOBAL			SCIP		
		# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)	# solved	Time (sec)	Gap (%)
5	6	6	2.6	–	6	0.2	–	6	11	–	6	9.0	–	6	14	–
6	7	7	13	–	7	0.5	–	6	83	11.7	7	114	–	7	256	–
7	8	8	114	–	8	6.7	–	8	866	–	7	1,415	4.7	6	633	1.2
8	9	0	–	7.2	8	1,033	35.5	0	–	22.0	0	–	33.6	0	–	9.0
9	10	0	–	22.3	0	–	44.2	0	–	32.4	0	–	46.4	0	–	18.8

Considering all five global solvers, BARON performs the best in terms of number of solved instances and solution time. The smallest average residual gaps for unsolved instances were obtained by SCIP, though possibly this does not result from tight relaxations constructed at its branch-and-bound nodes, rather results from aggressive branching. This is corroborated by the observation that the number of branch-and-bound nodes in SCIP runs was extremely larger than other solvers. Processing a large branch-and-bound tree takes more time, which might explain why SCIP could only solve fewer instances to guaranteed optimality within the time limit, performing generally worse than other four global solvers.

More specifically, for the problem of packing circles in a circle, BARON was able to prove optimality in 20 out of 35 instances, with the largest solvable input size being a single nine-circle instance. In contrast, our proposed algorithm (Table 7.1) could in addition solve a handful of nine-circle, ten-circle and eleven-circle instances, pushing the state-of-the-art in terms of what is considered solvable for this class of packing problems. Furthermore, the solution time of our approach is generally much less than that of the global optimization solvers. Turning our attention to the application of packing circles into a rectangle, BARON again performs the best among the five general-purpose global solvers, being able to solve to confirmed optimality 29 out of 40 instances. While our algorithm (Table 7.2) solved 28 instances with comparable solution time, it achieved a much smaller average residual gaps for the remaining instances. Notably, across both problem datasets, our proposed approach was able to close the gap in a total of 6 instances for which none of the global optimization solvers was able to prove optimality. The competitiveness of our approach against these solvers can also be inferred from rigorous performance profiles [66], which we present in Fig. 7.7.

7.6 CONCLUSIONS

In this chapter, we focused on a class of reverse convex constraints called circle-circle non-overlapping constraints, which are popular in many cutting and packing optimization models. Adapting a custom-built branch-and-bound algorithm that we had previously developed to address irregular shape nesting problems in Chapter 6, we proposed strengthened intersection cuts and various feasibility-based tightening techniques to expedite the search based on direct branching upon the set of non-overlapping constraints. To this end, we first generalized the intersection cut formula from the seminal paper of [23] to more generic cases where variables can be bounded by arbitrary values, and used the



(a) Packing circles into a circle

(b) Packing circles into a rectangle

Figure 7.7: Performance profiles across all benchmark instances of each problem variant. In both graphs, "This work" refers to the performance of our proposed algorithm using strengthened intersection cuts and feasibility-based tightening (BB+SIC+FBT). For each curve, the value at $t = 0$ provides the fraction of benchmark instances for which the corresponding solver/algorithm is fastest, while the limiting value at $t \to \infty$ provides the fraction of instances that could be solved within the time limit of 1 hour.

fact that a non-overlapping constraint represents a reverse convex domain to incorporate such intersection cuts into our relaxations. In addition, we proposed feasibility-based tightening techniques that differ from the ones commonly used in global optimization, in the sense that we sought to reduce the feasible domain enforced by non-overlapping constraints directly, rather than rely on domain reduction from variable bounds. Our extensive computational studies on 75 circle packing instances elucidated the effectiveness of tightening the relaxations in terms of speeding up the branch-and-bound process, and showcased that the purposed-build search approach performs favorably as compared to using state-of-the-art, yet general-purpose global optimization solvers.

8

CONCLUSIONS AND FUTURE WORK

In this thesis, we focus on mathematical modeling and algorithmic development for addressing both vehicle routing problems and packing problems. We presented novel mixed-integer linear formulations to model several routing problems of considerable practical relevance; we formulated the problem of packing circles/irregular shapes as a non-convex quadractically constrained quadratic program. To solve the resulting mathematical models, we have proposed solution algorithms that are tailored for each of them. In the following sections, we summarize the key contributions of our work and then present several directions for future research.

8.1 CONTRIBUTIONS

In Chapter 2, we studied the continuous-time inventory routing problem. Inventory management, vehicle routing, and delivery-scheduling decisions are simultaneously considered in the context of inventory routing. The continuous-time feature requires that the distributor has to both monitor inventory levels at customers and make product replenishment decisions in continuous time, so as to ensure that stock levels are maintained within the desired intervals at any moment of the planning horizon.

- We proposed a novel mixed-integer linear programming formulation to model the continuous-time inventory routing problem. This formulation incorporates several ingenious modeling ideas to handle the multi-trip, multi-visit features as well as continuous-time inventory management.
- We proposed various types of tightening techniques to strengthen the linear programming relaxations. In particular, we adapted rounded capacity inequalities into our model and developed protocols to dynamically separate and add them during the branch-and-cut process.

- We conducted extensive computational studies on 90 benchmark instances from the literature and the computational results show that our branch-and-cut algorithm significantly outperforms the state-of-the-art approach. In particular, our algorithm could solve 56 of them to optimality within a reasonable amount of time and return a small residual gap for the remaining ones, while the state-of-the-art algorithm could only solve 26 out of 90 to guaranteed optimality.
- We further evaluated our algorithm on newly generated benchmark instances that are inspired by the real-life data from ROADEF/EURO Challenge 2016. Out of 63 benchmark instances, our branch-and-cut algorithm solved 56 of them to optimality, including a few 20-customer instances. This further demonstrates our proposed algorithm's efficiency.

In Chapter 3, we focused on the full truckload pickup and delivery problem (FTPDP) that is featured by a heterogeneous fleet, optional orders, multiple pickup points and loading dock capacity constraints.

- We proposed a novel mixed-integer linear programming formulation to model the FTPDP, which is featured by full truckload shipments, multiple pickup points, optional orders, and loading dock capacity restrictions.
- We tested our proposed model using real-life operational data and the computational results demonstrated the effectiveness and efficiency of our proposed approach. In particular, our MILP model could solve practical problems of up to 52 consignments to optimality within hundreds of seconds.
- We embedded the mathematical formulation into a simulation engine so as to evaluate the economic effect of allowing for pre-loading trucks. Our computational studies showed that adopting the pre-loading policy could save up to 2% of the operational cost in the application setting of our interest.

In Chapter 4, we considered the problem of estimating the marginal routing cost of serving an extra customer on top of a given distribution network. In this context, the main challenge stems from the intrinsic stochasticity in customer demands.

- We proposed a scenario-sampling framework to estimate the expected marginal cost of serving individual customers. Specifically, we obtained independent scenarios by sampling customer demands from their distribution and considered the sample average as an estimate of the cost. We proved that our proposed framework provides

statistical guarantee of the estimation accuracy, provided that a sufficiently large–informed by Hoeffding's inequality–sample size has been obtained.

- We modeled the multi-depot vehicle routing problem with inter-depot routes (MDVRPI) as a set partitioning formulation and proposed a branch-price-and-cut algorithm that incorporates several state-of-the-art techniques, including *ng*-routes, variable fixing, route enumeration, and limited-memory subset row cuts, among others.

- We conducted computational studies to show that our branch-price-and-cut algorithm significantly outperforms the state-of-the-art exact approach for the MDVRPI. In particular, our algorithm proved optimality for all previously open MDVRPI benchmark instances involving up to 40 customers. We further pushed the envelope by extending the literature benchmarks with 70-customer instances and solving to proven optimality the vast majority of those as well.

- We demonstrated the scenario-sampling framework and the quality of its cost estimates, utilizing thousands of MDVRPI instance samples in each case, and we elucidated the effect on the marginal routing cost of factors such as customer locations and demand levels.

In Chapter 5, we were interested in solving vehicle routing problems under uncertainty from a robust optimization perspective. Given postulated uncertainty sets for customer demands and vehicle travel times, one aims to identify a set of cost-effective routes for vehicles to traverse such that along these routes vehicle capacity constraints and customer time window constraints are respected under any anticipated demand and travel time realization, respectively.

- We considered robust VRPs with uncertainty in customer demands and vehicle travel times. We considered five popular classes of uncertainty sets: cardinality-constrained sets, budget sets, factor models, ellipsoids, and discrete sets. We performed polyhedral studies on the cardinality-constrained set and the factor model, and reduce each set to its equivalent discrete set.

- We proposed a novel BPC algorithm to address robust VRPs under demand and travel time uncertainty. Our algorithm embedded cutting-plane techniques into the deterministic BPC framework. In particular, we utilized a deterministic pricing engine to generate partially robust feasible routes and then dynamically enforced robust rounded capacity inequalities and infeasible path elimination constraints as necessary

constraints to ensure the immunity of a routing design against infeasibility caused by variability in demands and travel times, respectively. To that end, we demonstrated that separating these inequalities can be done efficiently for the aforementioned uncertainty sets.

- We synopsized the existing methods from the literature that have demonstrated success in extending BPC algorithms to the solution of robust VRPs. We then made a detailed comparison between the literature approach and our cutting-plane approach in the following aspects: uncertainty sources, uncertainty sets, the time complexity of pricing subproblems and the tightness of LP relaxations, so as to demonstrate the applicability and limitations of each approach.

- We conducted comprehensive computational studies to evaluate our proposed algorithm on solving robust VRPs under the aforementioned uncertainty sets. Through computational experiments, we showed that the robust cutting-plane algorithm is versatile for the uncertainty sets of our interest. We compared its performance against those from the literature and demonstrate that our proposed algorithm's effectiveness and efficiency over the state-of-the-art approaches.

In Chapter 6, we studied the nesting problem, which aims to determine a configuration of a set of irregular shapes within a rectangular sheet of material of fixed width, such that no overlap among the shapes exists, and such that the length of the sheet is minimized.

- We developed an exact approach to solve the nesting problem to global optimality that does not rely on the use of general-purpose global optimization solvers.

- We identified a generic approach to dynamically satisfy reverse convex quadratic constraints commonly found in optimization models within the field of cutting and packing.

- We conducted a comprehensive computational study that illustrates the competitiveness of our approach compared to the previous state-of-the-art.

- We presented, for the first time in the open literature, provably optimal solutions for nesting problem instances featuring five polygons under free rotation.

In Chapter 7, we studied the circle-circle non-overlapping constraints, a form of reverse convex constraints that often arise in optimization models for cutting and packing applications. The feasible region induced by the intersection of circle-circle non-overlapping constraints is highly non-convex, and standard approaches to construct convex relaxations

for spatial branch-and-bound global optimization of such models typically yield unsatisfactory loose relaxations. Consequently, solving such non-convex models to guaranteed optimality remains extremely challenging even for the state-of-the-art codes.

- We developed a customized branch-and-bound approach for solving problems with circle-circle non-overlapping constraints.
- We generalized the intersection cut formula from the seminal paper of [23] to more generic cases with arbitrary variable bounds, and we applied a strengthened version of these cuts to tighten the BB node relaxations.
- We proposed three types of feasibility-based tightening techniques to further strengthen the BB node relaxations.
- We conducted a comprehensive computational study based on two popular variants of the circle packing problem to demonstrate that our approach achieves superior performance over the use of various state-of-the-art general-purpose global optimization solvers.

8.2 FUTURE DIRECTIONS

In this section, we propose a few research directions that might be worth exploring from either a practical or theoretical perspective. In Chapter 2, we presented a compact mathematical model to exactly solve the continuous-time inventory routing problem. Two main research directions can be pursued in the future.

- The problem definition of inventory routing problems we were concerned in Chapter 2 was adopted from [108]; it lacks several additional realistic features such as a heterogeneous fleet, time windows, driver-trailer scheduling constraints, and the–commonly used in practice–logistic ratio objective. It would be interesting to investigate how to model these practical considerations in a way that is not detrimental to computational tractability.
- From a practical perspective, the exact approach is not efficient to generate feasible solutions of high quality; hence, the development of heuristic approaches (e.g., matheuristics [88, 158]) is a right direction. For example, we can consider decisions to be made in a hierarchical way: a routing design is first fixed and then the optimal replenishment decisions (e.g., customer-visiting times and delivery quan-

tities) are identified via solving a linear program. Perturb the set of routes and iterate this process until a time limit is reached.

In the FTPDP, we allow for pre-loading trucks. Regarding this, two research directions are worth pursuing.

- We introduced a hyper-parameter α into the objective function so as to incentivize the pre-loading action. To increase the robustness of our chosen parameter, it would be more beneficial to develop a policy function that maps the order information into a more informed parameter value.
- Another perspective to allow for pre-loading trucks is to consider the practical application as a multi-period FTPDP, in which pre-loaded orders are specified for the first period and no pre-loading action is considered for the last period while the pre-loading decisions for other periods are to be made by the optimizer. The immediate advantage is that the hyper-parameter disappears while the objective is to minimize the actual monetary cost in a multi-period routing context.

Applying the branch-price-and-cut approach for addressing routing problems under uncertainty can be extended in several directions.

- For travel time uncertainty sets, we have considered two polyhedral sets: cardinality-constrained sets and discrete sets. One may attempt to extend the branch price-and-cut framework to solve routing problems with the travel time vector being supported on other polyhedral sets or ellipsoidal sets.
- Though various types of uncertainty sets were considered to model customer demands and vehicle travel times in Chapter 5, we did not discuss which set should be chosen when historical data is given. This direction is worth exploring.
- In this thesis, we have considered demand and travel time variability. In practice, customer order uncertainty also often arises in routing applications. The work of [161] considered customer order uncertainty in a multi-period vehicle routing problem and presented a branch-and-cut approach. We can extend the branch-price-cut framework to address such more involved, multi-period settings.
- In Chapter 5, the aim was to design *a priori* robust feasible routes; hence it was a single-stage optimization problem and no recourse decisions were to be made. If we model the vehicle routing problem under uncertainty as a two-stage robust optimization problem (i.e., a tentative delivery schedule is a "here-and-now" decision

while recourse actions, e.g., detouring to the depot, are treated as "wait-and-see" decisions after uncertain parameters materialize), an interesting question is how to adapt a branch-price-and-cut framework to solve such an *adjustable robust vehicle routing problem*.

- Robust optimization is one of the most popular frameworks to address optimization under uncertainty. Recently, distributionally robust optimization has received lots of attention. The work of [79] considered the *distributionally robust chance-constrained vehicle routing problem*. An interesting research direction is how to apply the branch-price-and-cut approach for addressing this problem.

The global optimization research is mainly focused on deriving tight relaxations. Two research avenues can be pursued for solving mathematical models with reverse convex constraints.

- Given a reverse convex constraint, we generated intersection cuts from the simplicial cone. As the work of [24] has pointed out, one may choose other polyhedra to derive valid intersection cuts. It would be interesting to investigate alternative polyhedra for deriving valid cuts that are stronger than the standard intersection cuts.

- Another research opportunity is to embed intersection cuts for reverse convex constraints into a convex-relaxation based, spatial branch-and-bound algorithm, and to quantify the computational benefit in the global optimization of generic non-convex problems, as the prevalence of reverse convex terms in the latter increases.

BIBLIOGRAPHY

[1] http://www.packomania.com/. accessed on January 8, 2019.

[2] N. Absi, D. Cattaruzza, D. Feillet, M. Ogier, and F. Semet. "A heuristic branch-cut-and-price algorithm for the ROADEF/EURO challenge on Inventory Routing." *Transportation Science* (2020).

[3] T. Achterberg. "SCIP: solving constraint integer programs." *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.

[4] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. "Presolve reductions in mixed integer programming." *INFORMS Journal on Computing* (2019).

[5] A. Agra et al. "Layered formulation for the robust vehicle routing problem with time windows." In: *International Symposium on Combinatorial Optimization*. Springer. 2012, pp. 249–260.

[6] A. Agra et al. "The robust vehicle routing problem with time windows." *Computers & operations research* 40.3 (2013), pp. 856–866.

[7] A. Agra, M. Christiansen, L. M. Hvattum, and F. Rodrigues. "Robust optimization for a maritime inventory routing problem." *Transportation Science* 52.3 (2018), pp. 509–525.

[8] E. Air Liquide. *ROADEF. Inventory Routing Problem description for ROADEF/EURO 2016 Challenge*.

[9] R Alvarez-Valdes, A Martinez, and J. Tamarit. "A branch & bound algorithm for cutting and packing irregularly shaped pieces." *International Journal of Production Economics* 145.2 (2013), pp. 463–477.

[10] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit. "Reactive GRASP for the strip-packing problem." *Computers & Operations Research* 35.4 (2008), pp. 1065–1083.

[11] A. Alves Pessoa, L. Di Puglia Pugliese, F. Guerriero, and M. Poss. "Robust constrained shortest path problems under budgeted uncertainty." *Networks* 66.2 (2015), pp. 98–111.

[12] E. Angelelli and M. G. Speranza. "The periodic vehicle routing problem with intermediate facilities." *European journal of Operational research* 137.2 (2002), pp. 233–247.

[13] K. M. Anstreicher. "On convex relaxations for quadratically constrained quadratic programming." *Mathematical programming* 136.2 (2012), pp. 233–251.

[14] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[15] M. P. de Aragao and E. Uchoa. "Integer program reformulation for robust branch-and-cut-and-price algorithms." In: *Mathematical Program in Rio: a Conference in Honour of Nelson Maculan*. Citeseer. 2003, pp. 56–61.

[16] C. Archetti, G. Desaulniers, and M. G. Speranza. "Minimizing the logistic ratio in the inventory routing problem." *EURO Journal on Transportation and Logistics* 6.4 (2017), pp. 289–306.

[17] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza. "A branch-and-cut algorithm for a vendor-managed inventory-routing problem." *Transportation science* 41.3 (2007), pp. 382–391.

[18] S. Arunapuram, K. Mathur, and D. Solow. "Vehicle routing and scheduling with full truckloads." *Transportation Science* 37.2 (2003), pp. 170–182.

[19] N. Ascheuer, M. Fischetti, and M. Grötschel. "Solving the asymmetric travelling salesman problem with time windows by branch-and-cut." *Mathematical Programming* 90.3 (2001), pp. 475–506.

[20] P. Augerat et al. *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG, 1995.

[21] P. Augerat, J.-M. Belenguer, E. Benavent, A. Corbéran, and D. Naddef. "Separating capacity constraints in the CVRP using tabu search." *European Journal of Operational Research* 106.2-3 (1998), pp. 546–557.

[22] P. Avella, M. Boccia, and L. A. Wolsey. "Single-period cutting planes for inventory routing problems." *Transportation Science* 52.3 (2017), pp. 497–508.

[23] E. Balas. "Intersection cuts — a new type of cutting planes for integer programming." *Operations Research* 19.1 (1971), pp. 19–39.

[24] E. Balas and F. Margot. "Generalized intersection cuts and a new cut generating paradigm." *Mathematical Programming* 137.1-2 (2013), pp. 19–35.

[25] R. Baldacci, N. Christofides, and A. Mingozzi. "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts." *Mathematical Programming* 115.2 (2008), pp. 351–385.

[26] R. Baldacci, A. Mingozzi, and R. Roberti. "New route relaxation and pricing strategies for the vehicle routing problem." *Operations research* 59.5 (2011), pp. 1269–1283.

[27] R. Baldacci, A. Lim, E. Traversi, and R. Wolfler Calvo. "Optimal solution of vehicle routing problems with fractional objective function." *Transportation Science* (2020).

[28] M. L. Balinski and R. E. Quandt. "On an integer program for a delivery problem." *Operations research* 12.2 (1964), pp. 300–304.

[29] X. Bao, N. V. Sahinidis, and M. Tawarmalani. "Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs." *Optimization Methods & Software* 24.4-5 (2009), pp. 485–504.

[30] L. Baumgartner, V. Schmid, and C. Blum. "Solving the two-dimensional bin packing problem with a probabilistic multi-start heuristic." In: *International Conference on Learning and Intelligent Optimization*. Springer. 2011, pp. 76–90.

[31] J.-M. Belenguer, M. Martinez, and E Mota. "A lower bound for the split delivery vehicle routing problem." *Operations research* 48.5 (2000), pp. 801–810.

[32] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. "Branching and bounds tighteningtechniques for non-convex MINLP." *Optimization Methods & Software* 24.4-5 (2009), pp. 597–634.

[33] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Vol. 28. Princeton University Press, 2009.

[34] A. Ben-Tal and A. Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Vol. 2. Siam, 2001.

[35] J. A. Bennell and J. F. Oliveira. "The geometry of nesting problems: A tutorial." *European journal of operational research* 184.2 (2008), pp. 397–415.

[36] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, 2008.

[37] D. Bertsimas and M. Sim. "The price of robustness." *Operations research* 52.1 (2004), pp. 35–53.

[38] N. Bianchessi and S. Irnich. "Branch-and-cut for the split delivery vehicle routing problem with time windows." *Transportation Science* 53.2 (2019), pp. 442–462.

[39] D. Bienstock, C. Chen, and G. Munoz. "Outer-product-free sets for polynomial optimization and oracle-based cuts." *arXiv preprint arXiv:1610.04604* (2016).

[40] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[41] A. Bortfeldt. "A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces." *European Journal of Operational Research* 172.3 (2006), pp. 814–837.

[42] S. Braaten, O. Gjønnes, L. M. Hvattum, and G. Tirado. "Heuristics for the robust vehicle routing problem with time windows." *Expert Systems with Applications* 77 (2017), pp. 136–147.

[43] K. Buhrkal, A. Larsen, and S. Ropke. "The waste collection vehicle routing problem with time windows in a city logistics context." *Procedia-Social and Behavioral Sciences* 39 (2012), pp. 241–254.

[44] E. Burke, R. Hellier, G. Kendall, and G. Whitwell. "A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem." *Operations Research* 54.3 (2006), pp. 587–601.

[45] E. K. Burke, M. R. Hyde, and G. Kendall. "Evolving bin packing heuristics with genetic programming." In: *Parallel Problem Solving from Nature-PPSN IX*. Springer, 2006, pp. 860–869.

[46] I. Castillo, F. J. Kampas, and J. D. Pintér. "Solving circle packing problems by global optimization: numerical results and industrial applications." *European Journal of Operational Research* 191.3 (2008), pp. 786–802.

[47] D. Cattaruzza, N. Absi, and D. Feillet. "Vehicle routing problems with multiple trips." *4OR* 14.3 (2016), pp. 223–259.

[48] S. Chaudhuri, R. Motwani, and V. Narasayya. "Random sampling for histogram construction: How much is enough?" *ACM SIGMOD Record* 27.2 (1998), pp. 436–447.

[49] L. H. Cherri et al. "Robust mixed-integer linear programming models for the irregular strip packing problem." *European Journal of Operational Research* 253.3 (2016), pp. 570–583.

[50] G. Clarke and J. W. Wright. "Scheduling of vehicles from a central depot to a number of delivery points." *Operations research* 12.4 (1964), pp. 568–581.

[51] L. C. Coelho, J.-F. Cordeau, and G. Laporte. "Consistency in multi-vehicle inventory-routing." *Transportation Research Part C: Emerging Technologies* 24 (2012), pp. 270–287.

[52] L. C. Coelho, J.-F. Cordeau, and G. Laporte. "Thirty years of inventory routing." *Transportation Science* 48.1 (2013), pp. 1–19.

[53] L. C. Coelho, J.-F. Cordeau, and G. Laporte. "Thirty years of inventory routing." *Transportation Science* 48.1 (2014), pp. 1–19.

[54] L. C. Coelho and G. Laporte. "The exact solution of several classes of inventory-routing problems." *Computers & Operations Research* 40.2 (2013), pp. 558–565.

[55] L. C. Coelho and G. Laporte. "Improved solutions for inventory-routing problems through valid inequalities and input ordering." *International Journal of Production Economics* 155 (2014), pp. 391–397.

[56] C. Contardo and R. Martinelli. "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints." *Discrete Optimization* 12 (2014), pp. 129–146.

[57] L. Costa, C. Contardo, and G. Desaulniers. "Exact branch-price-and-cut algorithms for vehicle routing." *Transportation Science* 53.4 (2019), pp. 946–985.

[58] B. Crevier, J.-F. Cordeau, and G. Laporte. "The multi-depot vehicle routing problem with inter-depot routes." *European Journal of Operational Research* 176.2 (2007), pp. 756–773.

[59] C. F. Daganzo. "The distance traveled to visit N points with a maximum of C stops per vehicle: An analytic model and an application." *Transportation science* 18.4 (1984), pp. 331–350.

[60] K. L. Daniels and V. Milenkovic. "Multiple Translational Containment: Approximate and Exact Algorithms." In: *SODA*. Vol. 95. Citeseer. 1995, pp. 205–214.

[61] G. B. Dantzig and J. H. Ramser. "The truck dispatching problem." *Management science* 6.1 (1959), pp. 80–91.

[62] J. De La Vega, P. Munari, and R. Morabito. "Robust optimization for the vehicle routing problem with multiple deliverymen." *Central European Journal of Operations Research* (2017), pp. 1–32.

[63] E. D. Demaine, S. P. Fekete, and R. J. Lang. "Circle packing for origami design is hard." *arXiv preprint arXiv:1008.1224* (2010).

[64] G. Desaulniers, J. G. Rakke, and L. C. Coelho. "A branch-price-and-cut algorithm for the inventory-routing problem." *Transportation Science* 50.3 (2015), pp. 1060–1076.

[65] T. Dinh, R. Fukasawa, and J. Luedtke. "Exact algorithms for the chance-constrained vehicle routing problem." *Mathematical Programming* 172.1-2 (2018), pp. 105–138.

[66] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles." *Mathematical programming* 91.2 (2002), pp. 201–213.

[67] Y. Dong, C. T. Maravelias, J. M. Pinto, and A. Sundaramoorthy. "Solution methods for vehicle-based inventory routing problems." *Computers & Chemical Engineering* 101 (2017), pp. 259–278.

[68] M. Dror. "Note on the complexity of the shortest path models for column generation in VRPTW." *Operations Research* 42.5 (1994), pp. 977–978.

[69] S. Erdoğan and E. Miller-Hooks. "A green vehicle routing problem." *Transportation Research Part E: Logistics and Transportation Review* 48.1 (2012), pp. 100–114.

[70] M. F. Fauske, C. Mannino, and P. Ventura. "Generalized Periodic Vehicle Routing and Maritime Surveillance." *Transportation Science* 54.1 (2020), pp. 164–183.

[71] D. Feillet. "A tutorial on column generation and branch-and-price for vehicle routing problems." *4or* 8.4 (2010), pp. 407–424.

[72] M. A. Figliozzi. "Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints." *Transportation Research Record* 2089.1 (2008), pp. 1–8.

[73] M. Fischetti, J. J. S. Gonzalez, and P. Toth. "Solving the orienteering problem through branch-and-cut." *INFORMS Journal on Computing* 10.2 (1998), pp. 133–148.

[74] M. Fischetti and I. Luzzi. "Mixed-integer programming models for nesting problems." *Journal of Heuristics* 15.3 (2009), pp. 201–226.

[75] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. "On the use of intersection cuts for bilevel optimization." *Mathematical Programming* 172.1-2 (2018), pp. 77–103.

[76] J. E. Fokkema, M. J. Land, L. C. Coelho, H. Wortmann, and G. B. Huitema. "A continuous-time supply-driven inventory-constrained routing problem." *Omega* 92 (2020), p. 102151.

[77] R. Fukasawa et al. "Robust branch-and-cut-and-price for the capacitated vehicle routing problem." *Mathematical programming* 106.3 (2006), pp. 491–511.

[78] M. Gendreau, O. Jabali, and W. Rei. "Chapter 8: Stochastic vehicle routing problems." In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 2014, pp. 213–239.

[79] S. Ghosal and W. Wiesemann. "The distributionally robust chance constrained vehicle routing problem." *Operations Research* (2020).

[80] P. B. Gibbons and S. Tirthapura. "Estimating simple functions on the union of data streams." In: *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*. ACM. 2001, pp. 281–291.

[81] A. M. Gomes and J. F. Oliveira. "Solving irregular strip packing problems by hybridising simulated annealing and linear programming." *European Journal of Operational Research* 171.3 (2006), pp. 811–829.

[82] C. E. Gounaris, W. Wiesemann, and C. A. Floudas. "The robust capacitated vehicle routing problem under demand uncertainty." *Operations Research* 61.3 (2013), pp. 677–693.

[83] C. E. Gounaris, P. P. Repoussis, C. D. Tarantilis, W. Wiesemann, and C. A. Floudas. "An adaptive memory programming framework for the robust capacitated vehicle routing problem." *Transportation Science* 50.4 (2014), pp. 1239–1260.

[84] A. Grimault, N. Bostel, and F. Lehuédé. "An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization." *Computers & Operations Research* 88 (2017), pp. 1–14.

[85] C. Groër, B. Golden, and E. Wasil. "The consistent vehicle routing problem." *Manufacturing & service operations management* 11.4 (2009), pp. 630–643.

[86] M. Gronalt, R. F. Hartl, and M. Reimann. "New savings based algorithms for time constrained pickup and delivery of full truckloads." *European Journal of Operational Research* 151.3 (2003), pp. 520–535.

[87] R. Grønhaug, M. Christiansen, G. Desaulniers, and J. Desrosiers. "A branch-and-price method for a liquefied natural gas inventory routing problem." *Transportation Science* 44.3 (2010), pp. 400–415.

[88] Y. He, C. Artigues, C. Briand, N. Jozefowiez, and S. U. Ngueveu. "A Matheuristic with Fixed-Sequence Reoptimization for a Real-Life Inventory Routing Problem." *Transportation Science* (2020).

[89] K. Helsgaun. "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems." *Roskilde: Roskilde University* (2017).

[90] M. Hifi and R. M'hallah. "A literature review on circle and sphere packing problems: Models and methodologies." *Advances in Operations Research* 2009 (2009).

[91] R. J. Hillestad and S. E. Jacobsen. "Reverse convex programming." *Applied Mathematics and Optimization* 6.1 (1980), pp. 63–78.

[92] W. Hoeffding. "Probability inequalities for sums of bounded random variables." In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[93] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen. "Industrial aspects and literature survey: Fleet composition and routing." *Computers & Operations Research* 37.12 (2010), pp. 2041–2061.

[94] E. Hopper and B. C. Turton. "A review of the application of meta-heuristic algorithms to 2D strip packing problems." *Artificial Intelligence Review* 16.4 (2001), pp. 257–300.

[95] R. Horst and H. Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

[96] C Hu, J Lu, X Liu, and G Zhang. "Robust vehicle routing problem with hard time windows under demand and travel time uncertainty." *Computers & Operations Research* 94 (2018), pp. 139–153.

[97] C. A. Irawan, D. Ouelhadj, D. Jones, M. Stålhane, and I. B. Sperstad. "Optimisation of maintenance routing and scheduling for offshore wind farms." *European Journal of Operational Research* 256.1 (2017), pp. 76–89.

[98] S. Irnich, G. Desaulniers, J. Desrosiers, and A. Hadjar. "Path-reduced costs for eliminating arcs in routing and scheduling." *INFORMS Journal on Computing* 22.2 (2010), pp. 297–313.

[99] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. "Subset-row inequalities applied to the vehicle-routing problem with time windows." *Operations Research* 56.2 (2008), pp. 497–511.

[100] D. R. Jones. "A fully general, exact algorithm for nesting irregular shapes." *Journal of Global Optimization* 59.2-3 (2014), pp. 367–404.

[101] B. Kallehauge, N. Boland, and O. B. Madsen. "Path inequalities for the vehicle routing problem with time windows." *Networks: An International Journal* 49.4 (2007), pp. 273–293.

[102] I Karaoğlan. "A branch-and-cut algorithm for the vehicle routing problem with multiple use of vehicles." *Int'l J. of Lean Thinking* 6.1 (2015).

[103] A. Khajavirad. "Packing circles in a square: a theoretical comparison of various convexification techniques" (2017).

[104] A. Kheiri. "Heuristic Sequence Selection for Inventory Routing Problem." *Transportation Science* (2020).

[105] B.-I. Kim, S. Kim, and S. Sahoo. "Waste collection vehicle routing problem with time windows." *Computers & Operations Research* 33.12 (2006), pp. 3624–3642.

[106] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. "The sample average approximation method for stochastic discrete optimization." *SIAM Journal on Optimization* 12.2 (2002), pp. 479–502.

[107] E. R. Kone and M. H. Karwan. "Combining a new data classification technique and regression analysis to predict the Cost-To-Serve new customers." *Computers & Industrial Engineering* 61.1 (2011), pp. 184–197.

[108] F. Lagos, N. Boland, and M. Savelsbergh. "The Continuous-Time Inventory-Routing Problem." *Transportation Science* (2020).

[109] F. A. Lagos Gonzalez. "Exact Algorithms For Routing Problems." PhD thesis. Georgia Institute of Technology, 2019.

[110] G. Laporte and Y. Nobert. "A branch and bound algorithm for the capacitated vehicle routing problem." *Operations-Research-Spektrum* 5.2 (1983), pp. 77–85.

[111] G. Laporte, Y. Nobert, and M. Desrochers. "Optimal routing under capacity and distance restrictions." *Operations research* 33.5 (1985), pp. 1050–1073.

[112] G. Laporte, S. Ropke, and T. Vidal. "Chapter 4: Heuristics for the vehicle routing problem." In: *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. SIAM, 2014, pp. 87–116.

[113] C. Lee, K. Lee, and S. Park. "Robust vehicle routing problem with deadlines and travel time/demand uncertainty." *Journal of the Operational Research Society* 63.9 (2012), pp. 1294–1306.

[114] T. Lee and C. Kwon. "A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty." *4OR* 12.4 (2014), pp. 373–378.

[115] A. Lodi, S. Martello, and D. Vigo. "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems." *INFORMS Journal on Computing* 11.4 (1999), pp. 345–357.

[116] A. Lodi, S. Martello, and D. Vigo. "Heuristic algorithms for the three-dimensional bin packing problem." *European Journal of Operational Research* 141.2 (2002), pp. 410–420.

[117] C. O. López and J. E. Beasley. "A heuristic for the circle packing problem with a variety of containers." *European Journal of Operational Research* 214.3 (2011), pp. 512–525.

[118] D. Lu and F. Gzara. "The robust vehicle routing problem with time windows: Solution by branch and price and cut." *European Journal of Operational Research* 275.3 (2019), pp. 925–938.

[119] M. E. Lübbecke and J. Desrosiers. "Selected topics in column generation." *Operations research* 53.6 (2005), pp. 1007–1023.

[120] J. Lysgaard. "CVRPSEP: A package of separation routines for the capacitated vehicle routing problem" (2003).

[121] J. Lysgaard, A. N. Letchford, and R. W. Eglese. "A new branch-and-cut algorithm for the capacitated vehicle routing problem." *Mathematical Programming* 100.2 (2004), pp. 423–445.

[122] N. Menakerman and R. Rom. "Bin packing with item fragmentation." In: *Workshop on Algorithms and Data Structures*. Springer. 2001, pp. 313–324.

[123] V. J. Milenkovic. "Rotational polygon containment and minimum enclosure using only robust 2D constructions." *Computational Geometry* 13.1 (1999), pp. 3–19.

[124] R. Misener and C. A. Floudas. "GloMIQO: Global mixed-integer quadratic optimizer." *Journal of Global Optimization* 57.1 (2013), pp. 3–50.

[125] R. Misener and C. A. Floudas. "ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations." *Journal of Global Optimization* 59.2-3 (2014), pp. 503–526.

[126] P. Munari et al. "The robust vehicle routing problem with time windows: compact formulation and branch-price-and-cut method." *Transportation Science* (2019).

[127] I. Muter, J.-F. Cordeau, and G. Laporte. "A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes." *Transportation Science* 48.3 (2014), pp. 425–441.

[128] J. F. Oliveira, A. M. Gomes, and J. S. Ferreira. "TOPOS–A new constructive algorithm for nesting problems." *OR-Spektrum* 22.2 (2000), pp. 263–284.

[129] O. Ö. Özener, Ö. Ergun, and M. Savelsbergh. "Allocating cost of service to customers in inventory routing." *Operations Research* 61.1 (2013), pp. 112–125.

[130] D Pecin. "Exact algorithms for the capacitated vehicle routing problem." *Unpublished doctoral dissertation, Pontifícia Universidade Católica do Rio de Janiero, Rio de Janiero, Brazil* (2014).

[131] D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa. "Improved branch-cut-and-price for capacitated vehicle routing." *Mathematical Programming Computation* 9.1 (2017), pp. 61–100.

[132] D. Pecin, A. Pessoa, M. Poggi, E. Uchoa, and H. Santos. "Limited memory rank-1 cuts for vehicle routing problems." *Operations Research Letters* 45.3 (2017), pp. 206–209.

[133] D. Pecin, C. Contardo, G. Desaulniers, and E. Uchoa. "New enhancements for the exact solution of the vehicle routing problem with time windows." *INFORMS Journal on Computing* 29.3 (2017), pp. 489–502.

[134] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. "Automation and combination of linear-programming based stabilization techniques in column generation." *INFORMS Journal on Computing* 30.2 (2018), pp. 339–360.

[135] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. "A Generic Exact Solver for Vehicle Routing and Related Problems." In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2019, pp. 354–369.

[136] A. A. Pessoa, M. Poss, R. Sadykov, and F. Vanderbeck. "Branch-and-cut-and-price for the robust capacitated vehicle routing problem with knapsack uncertainty" (2018).

[137] S. Poikonen, X. Wang, and B. Golden. "The vehicle routing problem with drones: Extended models and connections." *Networks* 70.1 (2017), pp. 34–43.

[138] L. D. P. Pugliese and F. Guerriero. "A survey of resource constrained shortest path problems: Exact solution approaches." *Networks* 62.3 (2013), pp. 183–200.

[139] Y. Puranik and N. V. Sahinidis. "Domain reduction techniques for global NLP and MINLP optimization." *Constraints* 22.3 (2017), pp. 338–376.

[140] T. Radzik. "Fractional combinatorial optimization." *Handbook of combinatorial optimization* (2013), pp. 1311–1355.

[141] S. Rebennack. "Computing tight bounds via piecewise linear functions through the example of circle cutting problems." *Mathematical Methods of Operations Research* 84.1 (2016), pp. 3–57.

[142] G. Righini and M. Salani. "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints." *Discrete Optimization* 3.3 (2006), pp. 255–273.

[143] R. Roberti and A. Mingozzi. "Dynamic ng-path relaxation for the delivery man problem." *Transportation Science* 48.3 (2014), pp. 413–424.

[144] P. Rocha, R. Rodrigues, A. M. Gomes, F. M. Toledo, and M. Andretta. "Circle covering representation for nesting problems with continuous rotations." *IFAC Proceedings Volumes* 47.3 (2014), pp. 5235–5240.

[145] M. O. Rodrigues, L. H. Cherri, and L. R. Mundim. "MIP models for the irregular strip packing problem: new symmetry breaking constraints." In: *ITM Web of Conferences*. Vol. 14. EDP Sciences. 2017, p. 00005.

[146] S. Røpke. "Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems." *Presentation in Column Generation* (2012).

[147] R. Sadykov, E. Uchoa, and A. Pessoa. "A bucket graph based labeling algorithm with application to vehicle routing." *Cadernos do LOGIS* 7 (2017).

[148] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, and E. Uchoa. "Primal heuristics for branch and price: The assets of diving methods." *INFORMS Journal on Computing* 31.2 (2019), pp. 251–267.

[149] G. Scheithauer. *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*. Vol. 263. Springer, 2017.

[150] M. Schiffer, M. Schneider, G. Walther, and G. Laporte. "Vehicle Routing and Location Routing with Intermediate Stops: A Review." *Transportation Science* (2019).

[151] M. Schneider and M. Drexl. "A survey of the standard location-routing problem." *Annals of Operations Research* 259.1-2 (2017), pp. 389–414.

[152] F. Serrano. "Intersection cuts for factorable MINLP." In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2019, pp. 385–398.

[153] T. Singh, J. E. Arbogast, and N. Neagu. "An incremental approach using local-search heuristic for inventory routing problem in industrial gases." *Computers & Chemical Engineering* 80 (2015), pp. 199–210.

[154] R. Soares, A. Marques, P. Amorim, and J. Rasinmäki. "Multiple vehicle synchronisation in a full truck-load pickup and delivery problem: A case-study in the biomass supply chain." *European Journal of Operational Research* 277.1 (2019), pp. 174–194.

[155] E. Solano-Charris, C. Prins, and A. C. Santos. "Local search based metaheuristics for the robust vehicle routing problem with discrete scenarios." *Applied Soft Computing* 32 (2015), pp. 518–531.

[156] M. M. Solomon. "Algorithms for the vehicle routing and scheduling problems with time window constraints." *Operations research* 35.2 (1987), pp. 254–265.

[157] Y. G. Stoyan and G. Yas' kov. "A mathematical model and a solution method for the problem of placing various-sized circles into a strip." *European Journal of Operational Research* 156.3 (2004), pp. 590–600.

[158] Z. Su, Z. Lü, Z. Wang, Y. Qi, and U. Benlic. "A Matheuristic Algorithm for the Inventory Routing Problem." *Transportation Science* (2020).

[159] A. Subramanyam, P. P. Repoussis, and C. E. Gounaris. "Robust optimization of a broad class of heterogeneous vehicle routing problems under demand uncertainty." *INFORMS Journal on Computing* (2020).

[160] A. Subramanyam, A. Wang, and C. E. Gounaris. "A scenario decomposition algorithm for strategic time window assignment vehicle routing problems." *Transportation Research Part B: Methodological* 117 (2018), pp. 296–317.

[161] A. Subramanyam, F. Mufalli, J. M. Pinto, and C. E. Gounaris. "Robust multi-period vehicle routing under customer order uncertainty." *Optimization Online* (2017).

[162] L. Sun, M. H. Karwan, B. Gemici-Ozkan, and J. M. Pinto. "Estimating the long-term cost to serve new customers in joint distribution." *Computers & Industrial Engineering* 80 (2015), pp. 1–11.

[163] I. Sungur, F. Ordónez, and M. Dessouky. "A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty." *IIE Transactions* 40.5 (2008), pp. 509–523.

[164] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. "A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities." *INFORMS Journal on Computing* 20.1 (2008), pp. 154–168.

[165] F. Tardella. "On the existence of polyhedral convex envelopes." In: *Frontiers in global optimization*. Springer, 2004, pp. 563–573.

[166] M. Tawarmalani and N. V. Sahinidis. "A polyhedral branch-and-cut approach to global optimization." *Mathematical programming* 103.2 (2005), pp. 225–249.

[167] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[168] M. Turkensteen and A. Klose. "Demand dispersion and logistics costs in one-to-many distribution systems." *European Journal of Operational Research* 223.2 (2012), pp. 499–507.

[169] E. Uchoa et al. "New benchmark instances for the capacitated vehicle routing problem." *European Journal of Operational Research* 257.3 (2017), pp. 845–858.

[170] A. W. Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.

[171] T. Vidal, G. Laporte, and P. Matl. "A concise guide to existing and emerging vehicle routing problem variants." *European Journal of Operational Research* (2019).

[172] S. Vigerske and A. Gleixner. "SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework." *Optimization Methods and Software* 33.3 (2018), pp. 563–593.

[173] G. Wäscher, H. Haußner, and H. Schumann. "An improved typology of cutting and packing problems." *European journal of operational research* 183.3 (2007), pp. 1109–1130.

[174] Y. Yao, P. T. Evers, and M. E. Dresner. "Supply chain integration in vendor-managed inventory." *Decision support systems* 43.2 (2007), pp. 663–674.

[175] B. Yildiz and M. Savelsbergh. "Provably high-quality solutions for the meal delivery routing problem." *Transportation Science* 53.5 (2019), pp. 1372–1388.

[176] B. Zeng and L. Zhao. "Solving two-stage robust optimization problems using a column-and-constraint generation method." *Operations Research Letters* 41.5 (2013), pp. 457–461.