Contributions to Exact Algorithms for Optimization in Warehousing and Flexible Manufacturing

Vom Fachbereich Wirtschaftswissenschaften
der Rheinland-Pfälzischen Technischen Universität
Kaiserslautern-Landau
zur Verleihung des akademischen Grades
Doctor rerum politicarum (Dr. rer. pol.)
genehmigte

Dissertation

vorgelegt von

Julia Wahlen, M. Sc.

Tag der mündlichen Prüfung:	09. Mai 2025
Dekan:	Prof. Dr. Florian Sahling
Vorsitzender:	Prof. Dr. Michael Hassemer
Berichterstattende:	Prof. Dr. Timo Gschwind
	Prof. Dr. Sanne Wøhlk

D 386
(2025)

Contents

List of Papers **vi**

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**
1.1 Exact Solution Methods 2
1.2 Considered Problems 3
1.3 Contributions and Outline 4

2 Branch-Price-and-Cut-Based Solution of Order Batching Problems
Julia Wahlen and Timo Gschwind **6**
2.1 Introduction 7
2.1.1 Literature Review 9
2.1.2 Contributions 11
2.1.3 Organization of the Paper 13
2.2 Problem Description and Mathematical Formulation 14
2.3 Branch-Price-and-Cut 16
2.3.1 Pricing Problem 16
2.3.1.1 SPPRC Formulation of the Pricing Problem 17
2.3.1.2 Basic Labeling Algorithm 18
2.3.1.3 Bounding Procedure 19
2.3.1.4 Acceleration Strategies 22
2.3.2 Cutting 22
2.3.2.1 Capacity Cuts 23
2.3.2.2 Subset-Row Cuts 27
2.3.3 Branching 28
2.3.4 BPC-based Heuristics 31
2.4 Computational Results 31
2.4.1 Benchmark Instances 32
2.4.2 Evaluation of Algorithmic Components 32
2.4.3 Computational Analysis of BPC Algorithm 33

2.4.4 Computational Analysis of BPC-based Heuristics 37
2.4.5 Comparison of Routing Strategies 40
2.5 Conclusions 41
Appendix 47
2.A Detailed Description of Routing Strategies 47
2.B Proof of Proposition 2.1 49
2.C Non-Monotonicity of Composite Routing Strategy 53
2.D Algorithm Design Choices 53
2.E Benchmark Instances 55
2.F Detailed Computational Results 56

3 Solving the Multi-Block Order Batching Problem with Branch-Price-and-Cut
Julia Wahlen **107**
3.1 Introduction 108
3.1.1 Contributions 109
3.1.2 Organization of the Paper 110
3.2 Literature Review 110
3.3 Problem Description and Solution Approach 113
3.3.1 Problem Definition and Mathematical Formulation 113
3.3.2 Branch-Price-and-Cut Method 114
3.3.2.1 Exact BPC 114
3.3.2.2 BPC-based Heuristics 116
3.4 Multi-Block Routing Strategies 116
3.4.1 Warehouse Layout 116
3.4.2 Detailed Description of Routing Strategies 117
3.4.3 Monotonicity Property 123
3.5 Computational Results 126
3.5.1 Benchmark Instances 126
3.5.2 Comparison with State-of-the-Art 127
3.5.3 Computational Analysis of BPC Algorithm 128
3.5.4 Evaluation of Routing Strategies 131
3.5.5 Detailed Analysis 132
3.5.6 Computational Analysis of BPC-based Heuristics 134
3.6 Conclusions 138
Appendix 144
3.A `Foodmart` State Space 144
3.B Instance-by-Instance Comparison 145
3.C Detailed Computational Results 146

4 Branch-and-Price for the Set-Union Bin Packing Problem
Julia Wahlen and Timo Gschwind **158**
4.1 Introduction . . . 159
4.1.1 Contributions . . . 160
4.1.2 Organization of the Paper . . . 161
4.2 Literature Review . . . 161
4.3 Problem Description and Mathematical Formulations . . . 163
4.3.1 Problem Definition . . . 163
4.3.2 Symmetric Formulation . . . 164
4.3.3 Asymmetric Representatives Formulation . . . 165
4.3.4 Set-Partitioning Formulation . . . 166
4.4 Branch-and-Price Algorithm . . . 166
4.4.1 Pricing Problem . . . 167
4.4.1.1 IP Formulation . . . 167
4.4.1.2 Item-based SPPRC . . . 168
4.4.1.3 Element-based SPPRC . . . 172
4.4.2 Branching . . . 175
4.5 Computational Results . . . 176
4.5.1 Benchmark Instances . . . 177
4.5.2 Analysis of Pricing Problem Solution Methods . . . 177
4.5.3 Comparison with State-of-the-Art . . . 179
4.5.4 Computational Analysis of B&P Algorithm . . . 181
4.5.5 Analysis of Lower Bounds . . . 184
4.6 Conclusions . . . 185
Appendix . . . 190
4.A Acceleration Strategies for Pricing Problem Solution . . . 190
4.B Algorithm Design Choices . . . 192
4.C Modification of Item-based SPPRC Graph for Branching . . . 193
4.D Benchmark Instances . . . 194
4.E Comparison of Completion Bounds . . . 195
4.F Detailed CG Process for Selected Instances . . . 198
4.G Influence of Instance Characteristics on B&P Algorithm . . . 200
4.H Influence of Additional UBs on B&P Algorithm . . . 205
4.I Instances Not Satisfying IRUP . . . 207

5 Conclusion **211**

Bibliography **213**

List of Papers

- Julia Wahlen[1] and Timo Gschwind[1] (2023). Branch-Price-and-Cut-Based Solution of Order Batching Problems. *Transportation Science* **57**(3), 756–777.
- Julia Wahlen (2024). Solving the Multi-Block Order Batching Problem with Branch-Price-and-Cut. Technical Report L-2024-03, Chair of Logistics, School of Business and Economics, University of Kaiserslautern-Landau, Kaiserslautern, Germany. Submitted to *European Journal of Operational Research*.
- Julia Wahlen and Timo Gschwind (Forthcoming). Branch-and-Price for the Set-Union Bin Packing Problem. *INFORMS Journal on Computing*.

[1]Chair of Logistics, School of Business and Economics, University of Kaiserslautern-Landau, Gottlieb-Daimler-Straße, 67663 Kaiserslautern, Germany

List of Figures

2.1 Rectangular parallel-aisles single-block warehouse layout of the standard OBP . . . 15
2.2 SPPRC representation of the pricing problem: linear directed multigraph for the example in Figure 2.1 and dual prices π_o . . . 18
2.3 Exemplary pricing procedure with labels and completion bounds for OBP instance of Figure 2.1 . . . 21
2.4 SPPRC representation of the pricing problem for the example instance of Figure 2.3 and Ryan-and-Foster branching decisions $\{1,2\}^s$, $\{2,3\}^s$, and $\{4,5\}^t$ . . . 30
2.5 Performance profiles of different variants of our BPC algorithm for the `H&W` instances (left) and the `M&Ö` instances (right) . . . 34
2.6 Picker routes for batch $b = \{2,4,5\}$ and different routing strategies 50
2.7 Picker routes for both interpretations of the composite strategy and batches $b_1 = \{6\}$ (in blue) and $b_2 = \{6,7\}$ (in red), $c_{b_1} = 48$, $c_{b_2} = 46$ 54

3.1 Rectangular parallel-aisles three-block warehouse layout . . . 118
3.2 Picking routes for batch $b = \{1,2,3,4,5\}$ and different routing strategies . . . 121
3.3 Picking routes for the traditional definitions of traversal and combined and batches $b_1 = \{2\}$ (blue) and $b_2 = \{2,3\}$ (red) . . . 125
3.4 Picking routes for the largest gap strategy and batches $b_1 = \{1\}$ (blue) and $b_2 = \{1,4\}$ (red) . . . 125
3.5 Warehouse layout of `Foodmart` instances . . . 144

4.1 Example solution of a unit-weight SUBP instance with four items $I = \{i_1, \ldots, i_4\}$ requiring elements $E_{i_1} = \{e_4, e_5\}$, $E_{i_2} = \{e_1\}$, $E_{i_3} = \{e_2, e_4\}$, $E_{i_4} = \{e_3, e_4\}$, and $Q = 3$ . . . 164
4.2 Linear directed multigraph G of the item-based SPPRC representation of the pricing problem . . . 169
4.3 Example representation for the determination of bound $B_1(L)$ for a label L at vertex i_2 . . . 173
4.4 Linear directed multigraph $\hat{G}$ of the element-based SPPRC representation of the pricing problem . . . 174

4.5 Computation time per pricing iteration with (gray) and without (blue) greedy heuristic for an exemplary instance 180
4.6 Modified linear directed multigraph G with Ryan-and-Foster *separate* branching decisions $\{i_1, i_2\}$ and $\{i_2, i_3\}$ 195
4.7 Computation time per pricing iteration with (gray) and without (blue) heuristic for the second instance with $Q = 15, m = 20, n = 70$ 198
4.8 Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 20$, $m = 35$, $n = 50$. 198
4.9 Computation time per pricing iteration with (gray) and without (blue) heuristic for the third instance with $Q = 25$, $m = 50$, $n = 50$ 199
4.10 Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 35$, $m = 40$, $n = 85$. 199
4.11 Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 50$, $m = 55$, $n = 60$. 200

List of Tables

2.1 Overview of exact solution approaches to the OBP 12
2.2 Summary results for different variants of our BPC algorithm 33
2.3 Comparison of our BPC algorithm with the approach of Muter and Öncan (2015) for routing strategies traversal, return and midpoint on the M&Ö instances 35
2.4 Summary results of our BPC algorithm for the M&Ö and H&W instances and all routing strategies 36
2.5 Detailed results of our BPC algorithm for the M&Ö and H&W instances and routing strategies traversal and optimal 37
2.6 Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the largest gap strategy on a subset of the H&W instances 38
2.7 Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the traversal strategy on a subset of the H&W instances 39
2.8 Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the traversal strategy on the large-scale ZKS instances 39
2.9 Summary results of our BPC-based heuristics with different time limits 39
2.10 Percentage increase in total traveled distances compared to the optimal routing strategy 41
2.11 Summary results of our BPC algorithm for the M&Ö instances 58
2.12 Summary results of our BPC algorithm for the H&W instances 59
2.13 Summary results of our BPC algorithm for the ZKS instances 60
2.14 Summary results of our BPC algorithm for the M&Ö-ext instances . 60
2.15 Summary results of our BPC algorithm for the W&G-g instances . . 61
2.16 Summary results of our BPC algorithm for the W&G-u instances . . 62
2.17 Detailed results of our BPC algorithm for the M&Ö instances and the traversal strategy 63
2.18 Detailed results of our BPC algorithm for the M&Ö instances and the return strategy 64
2.19 Detailed results of our BPC algorithm for the M&Ö instances and the midpoint strategy 65

2.20 Detailed results of our BPC algorithm for the M&Ö instances and the largest gap strategy . . . 66
2.21 Detailed results of our BPC algorithm for the M&Ö instances and the combined strategy . . . 67
2.22 Detailed results of our BPC algorithm for the M&Ö instances and the optimal strategy . . . 68
2.23 Detailed results of our BPC algorithm for the H&W instances and the traversal strategy . . . 69
2.24 Detailed results of our BPC algorithm for the H&W instances and the return strategy . . . 70
2.25 Detailed results of our BPC algorithm for the H&W instances and the midpoint strategy . . . 71
2.26 Detailed results of our BPC algorithm for the H&W instances and the largest gap strategy . . . 72
2.27 Detailed results of our BPC algorithm for the H&W instances and the combined strategy . . . 73
2.28 Detailed results of our BPC algorithm for the H&W instances and the optimal strategy . . . 74
2.29 Detailed results of our BPC algorithm for the ZKS instances and the traversal strategy . . . 75
2.30 Detailed results of our BPC algorithm for the ZKS instances and the return strategy . . . 75
2.31 Detailed results of our BPC algorithm for the ZKS instances and the midpoint strategy . . . 75
2.32 Detailed results of our BPC algorithm for the ZKS instances and the largest gap strategy . . . 76
2.33 Detailed results of our BPC algorithm for the ZKS instances and the combined strategy . . . 76
2.34 Detailed results of our BPC algorithm for the ZKS instances and the optimal strategy . . . 76
2.35 Detailed results of our BPC algorithm for the M&Ö-ext instances and the traversal strategy . . . 77
2.36 Detailed results of our BPC algorithm for the M&Ö-ext instances and the return strategy . . . 78
2.37 Detailed results of our BPC algorithm for the M&Ö-ext instances and the midpoint strategy . . . 79
2.38 Detailed results of our BPC algorithm for the M&Ö-ext instances and the largest gap strategy . . . 80
2.39 Detailed results of our BPC algorithm for the M&Ö-ext instances and the combined strategy . . . 81

2.40 Detailed results of our BPC algorithm for the M&Ö-ext instances and the optimal strategy 82
2.41 Detailed results of our BPC algorithm for the W&G-g instances and the traversal strategy 83
2.42 Detailed results of our BPC algorithm for the W&G-g instances and the return strategy 84
2.43 Detailed results of our BPC algorithm for the W&G-g instances and the midpoint strategy 85
2.44 Detailed results of our BPC algorithm for the W&G-g instances and the largest gap strategy 86
2.45 Detailed results of our BPC algorithm for the W&G-g instances and the combined strategy 87
2.46 Detailed results of our BPC algorithm for the W&G-g instances and the optimal strategy 88
2.47 Detailed results of our BPC algorithm for the W&G-u instances and the traversal strategy 89
2.48 Detailed results of our BPC algorithm for the W&G-u instances and the return strategy 90
2.49 Detailed results of our BPC algorithm for the W&G-u instances and the midpoint strategy 91
2.50 Detailed results of our BPC algorithm for the W&G-u instances and the largest gap strategy 92
2.51 Detailed results of our BPC algorithm for the W&G-u instances and the combined strategy 93
2.52 Detailed results of our BPC algorithm for the W&G-u instances and the optimal strategy 94
2.53 Comparison of the BPC-based heuristics on the M&Ö instances . . . 95
2.54 Comparison of the BPC-based heuristics on the H&W instances . . . 96
2.55 Comparison of the BPC-based heuristics on the ZKS instances . . . 97
2.56 Comparison of the BPC-based heuristics on the M&Ö-ext instances . 99
2.57 Comparison of the BPC-based heuristics on the W&G-g instances . . 100
2.58 Comparison of the BPC-based heuristics on the W&G-u instances . . 101
2.59 Percentage increase in total traveled distances compared to the optimal strategy for the M&Ö instances 102
2.60 Percentage increase in total traveled distances compared to the optimal strategy for the H&W UDD instances 103
2.61 Percentage increase in total traveled distances compared to the optimal strategy for the H&W CBD instances 104
2.62 Percentage increase in total traveled distances compared to the optimal strategy for the ZKS instances 105

3.1 Overview of exact and heuristic solution approaches to the multi-block OBP . . . 111
3.2 Comparison of our BPC algorithm with the B&C approach of Valle *et al.* (2017) for the optimal routing strategy on a subset of the `Foodmart` instances . . . 129
3.3 Comparison of our BPC algorithm with the B&C approach of Zhang and Gao (2023) for the optimal routing strategy on a modified subset of the `Foodmart` instances . . . 129
3.4 Summary results of our BPC algorithm for the `Foodmart` instances and the optimal routing strategy . . . 130
3.5 Summary results of our BPC algorithm for the `Scholz&Wäscher` instances and the optimal routing strategy . . . 131
3.6 Summary results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and all routing strategies . . . 133
3.7 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategies optimal and combined . . . 134
3.8 Comparison of our heuristics BPC-DF and SC with the DAA of Valle and Beasley (2020) and the CGH of Briant *et al.* (2020) for a subset of the `Foodmart` instances and the optimal routing strategy . 135
3.9 Comparison of our heuristics BPC-DF and SC for the routing strategies optimal and combined with the DAA of Valle and Beasley (2020) on a subset of large `Foodmart` instances . . . 136
3.10 Summary results of our heuristics BPC-DF and SC for a subset of large `Foodmart` instances and combined routing with different time limits . . . 137
3.11 Comparison of our heuristics BPC-DF and SC for the combined routing strategy with the CGH of Briant *et al.* (2020) for very large `Foodmart` instances . . . 137
3.12 List of feasible states of the `Foodmart` instances . . . 145
3.13 Comparison of our BPC algorithm with the B&C approach of Valle *et al.* (2017) on a subset of the `Foodmart` instances for the optimal routing strategy . . . 147
3.14 Comparison of our BPC algorithm with the approach of Zhang and Gao (2023) on a subset of the modified `Foodmart` instances for the optimal routing strategy . . . 148
3.15 Comparison of our heuristics BPC-DF and SC for the optimal routing strategy to the heuristic approaches of Valle and Beasley (2020) and Briant *et al.* (2020) on a subset of the `Foodmart` instances . . . 149

3.16 Comparison of our heuristics BPC-DF and SC for the routing strategies optimal and combined to the heuristic approaches of Valle and Beasley (2020) on a subset of large `Foodmart` instances 150
3.17 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy optimal 151
3.18 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy no-reversal . . 152
3.19 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy aisle-by-aisle . . 153
3.20 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy combined . . . 154
3.21 Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy traversal 155
3.22 Summary results of our BPC-based heuristics for the `Foodmart` instances and all routing strategies 156

4.1 Summary results for pricing variants of our B&P for `pagination` instances 178
4.2 Comparison of our B&P with the IPs ARF and SF-LEX-I of Jans and Desrosiers (2013) 181
4.3 Detailed results of our B&P algorithm 183
4.4 B&P results for different amounts of memory allowed 184
4.5 Analysis of IRUP and MIRUP 185
4.6 Summary results for different completion bounds of our B&P for `pagination` instances 197
4.7 Summary results of our B&P aggregated by number of elements . . 202
4.8 Summary results of our B&P aggregated by capacity 203
4.9 Summary results of our B&P aggregated by average frequency . . . 203
4.10 Summary results of our B&P aggregated by average cardinality . . 204
4.11 Summary results of our B&P aggregated by average number of items per bin 204
4.12 Summary results of our B&P incorporating different UBs 206

Chapter 1

Introduction

In the context of rapid industrial advancement and evolving market demands, warehousing operations and flexible manufacturing systems have become essential components of modern supply chains (Gu *et al.* 2007, Ghiani *et al.* 2013). These systems play a critical role in the production, storage, and distribution of goods, thereby maintaining operational continuity and enhancing competitive advantage (de Koster *et al.* 2007). Addressing these challenges requires sophisticated decision-making tools capable of optimizing multifaceted problems under practical constraints.

Combinatorial optimization, a specialized branch of mathematical optimization, provides a powerful framework for tackling such problems. It focuses on identifying optimal solutions within finite, yet often exponentially large, sets of feasible candidates. This field is integral to discrete decision-making tasks, such as minimizing costs or maximizing profits, under predefined constraints (Korte and Vygen 2018). Beyond warehousing and manufacturing, combinatorial optimization addresses diverse applications in logistics, including transportation and routing, location and network design, assignment and scheduling, as well as cutting and packing. Formally, these problems are often modeled using *integer linear programming* (IP). While combinatorial algorithms have proven effective for solving many such problems to optimality (e.g., Irnich *et al.* 2014, Delorme and Iori 2020, Kellerer *et al.* 2004), the $\mathcal{NP}$-hard nature of many problems makes exact solutions computationally challenging (Garey and Johnson 1979). Consequently, developing effective exact methods remains an essential research area, not only for their theoretical value but also as a foundation for heuristics that enable faster, reliable solving of large-scale instances.

This thesis focuses on the exact solution of two combinatorial optimization problems that originate from warehouse operations and flexible manufacturing systems. Section 1.1 provides an overview of the exact solution methods employed in this research. The specific problems studied are detailed in Section 1.2. Section 1.3 summarizes the contributions and outlines the structure of the thesis.

1.1 Exact Solution Methods

Fundamental exact approaches to combinatorial optimization problems include *branch-and-bound* (B&B) and *branch-and-cut* (B&C) methods. The idea of B&B is to systematically explore the solution space, using bounds to prune large portions of the space, and branching strategies to ensure integer feasible solutions (Dakin 1965, Nemhauser and Wolsey 2014). Branching decisions in IP can be made not only based on the values of individual variables but also on subsets of variables or specific properties that those variables exhibit (e.g., Ryan and Foster 1981). However, the branching rules employed must guarantee the exploration of the entire solution space in such a way that integer solutions are always eventually found, provided the problem is feasible. The B&C method dynamically integrates valid inequalities (cuts) to the B&B approach (Padberg and Rinaldi 1988). These cuts are designed to eliminate fractional solutions in the relaxed linear program, thereby improving the quality of the relaxation. Notable examples of these include Chvátal-Gomory cuts, which are general and applicable to all IP problems (Chvátal 1973, Gomory 1963). In contrast, specialized cuts, such as capacity cuts, are tailored to specific problem types (Lysgaard *et al.* 2004, Baldacci *et al.* 2008). Violated constraints are identified through a separation procedure, which detects and incorporates the necessary cuts.

A specialized technique applicable to large-scale combinatorial optimization problems involves *column generation* (CG). This method is particularly effective for solving linear programming relaxations of problems formulated with a large number of variables, such as those modeled through set-partitioning (Desrosiers *et al.* 2024). Notably, large-scale models often emerge from the decomposition of a compact formulation, as introduced by Dantzig and Wolfe (1960). The core idea of CG is to decompose the problem into a *master problem* and a *pricing problem* (Gilmore and Gomory 1961). The master problem is solved with a restricted set of variables, whereas the pricing problem generates new, potentially valuable columns (variables) to be added to the master problem. This iterative process continues until no further improving columns exist. The characteristics and complexity of the pricing problem are closely tied to the underlying combinatorial structure of the problem. For example, in packing or cutting problems, the pricing problem can commonly be formulated as a binary *knapsack problem* or a variant thereof (Martello and Toth 1990, Kellerer *et al.* 2004), both of which are $\mathcal{NP}$-hard (Garey and Johnson 1979). In routing problems, it often reduces to the *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers 2005), which is also $\mathcal{NP}$-hard (Dror 1994). Effectively solving the pricing problem is crucial to the overall performance of the CG approach. As highlighted by Irnich and Desaulniers (2005), *dynamic programming* (DP)-based labeling algorithms (e.g., Dijkstra 1959, Ahuja *et al.* 1995) serve as a primary method to address the SPPRC. To enhance

the computational efficiency of these algorithms, specialized label elimination techniques, such as dominance rules and completion bounds, are commonly applied. These techniques help reducing the search space by discarding labels that cannot contribute to an optimal solution of the SPPRC pricing problem. Additionally, incorporating heuristic pricing methods offers a complementary strategy to further accelerate the solution process.

The integration of CG into B&B frameworks has led to the development of *branch-and-price* (B&P) algorithms, where CG is applied at each node of the B&B tree. This combination significantly enhances the ability to solve large instances exactly by maintaining a manageable number of active variables throughout the optimization process (Desrosiers *et al.* 2024). *Branch-price-and-cut* (BPC) algorithms build upon this concept by incorporating valid inequalities to tighten the formulation, thereby further reducing the solution space. However, both cutting and branching decisions can impact the structure of the pricing problem and must be carefully accounted for in the CG method. Essentially, CG-based exact solution approaches, such as B&P and BPC, are not plug-and-play solvers; their effectiveness hinges on a thorough understanding of the specific problem structure.

1.2 Considered Problems

The first problem addressed in this thesis is the *order batching problem* (OBP) in warehousing. It involves designing a set of picking batches given a set of customer orders, each containing one or more individual items to be picked. The objective is to minimize the total distance traveled by the pickers in the warehouse, ensuring that each customer order is assigned to exactly one batch while all batches adhere to the capacity restrictions of the pickers (Wäscher 2004). To collect the items of a batch, a picker traverses the warehouse according to a predefined routing strategy. If the optimal routing strategy is employed – entailing the determination of the minimum distance tour for each batch – the OBP becomes equivalent to the *soft-clustered vehicle routing problem*, a variant of the prominent *vehicle routing problem*, tailored to a warehouse environment (Aerts *et al.* 2021). Alternatively, heuristic picker routing strategies can be used to simplify navigating, particularly for human pickers (de Koster *et al.* 1999b).

Each item is assigned a unique storage location within a rectangular warehouse with parallel aisles. A warehouse with two cross aisles, located at the front and back of the picking aisles, is classified as a standard *single-block* warehouse. In contrast, a *multi-block* warehouse includes three or more cross aisles, dividing the warehouse into distinct blocks. Given the differing characteristics of these configurations and the distinct routing strategies proposed for single-block and multi-block environments, this work addresses the single-block OBP and multi-block OBP separately.

The second optimization problem explored in this thesis is the *set-union bin packing problem* (SUBP), which has numerous practical applications. One notable example is the *tool switching instants problem* in flexible manufacturing systems (e.g., Konak *et al.* 2008). In these systems, each job requires a specific set of tools to be loaded into a machine, with each machine having its own tool magazine. The capacity of these magazines is typically insufficient to store all the tools needed for every job, resulting in machine stops to switch tools. The objective is to minimize the number of these machine stops, thereby improving production efficiency. Similarly, the *job grouping problem* involves the objective of assigning all jobs to machines such that the number of identical machines required is minimized (e.g., Tang and Denardo 1988).

The SUBP extends the well-known *bin packing problem* (BP) to model the underlying optimization problem associated with these applications. In the SUBP, a set of items is given, each requiring a specific subset of weighted elements. The objective is to allocate all items into the minimum number of bins such that the total weight of all elements needed by the items in a bin does not exceed the bin's capacity. In contrast to the classical BP, the SUBP allows for reduced total capacity usage in a bin whenever items share overlapping element requirements.

1.3 Contributions and Outline

The primary goal of this thesis is to contribute to the development and analysis of new exact solution approaches for the OBP, both in single- and multi-block warehouse configurations, and for the SUBP. Specifically, it introduces novel CG-based algorithms designed to address combinatorial optimization problems with complex cost structures, thereby advancing the understanding and applicability of CG techniques in the context of non-linear objective functions. The work presented in Chapters 2–4 consists of three articles, each of which has either been published in or is currently under review at a scientific journal. In the following, we outline the structure of the thesis and detail the contributions of each chapter.

Chapter 2 introduces the first BPC approach developed for the OBP with general weights. It targets the single-block OBP, employing various routing strategies including traversal, return, midpoint, largest gap, combined, and optimal. The CG pricing problem is formulated as an SPPRC on a linear directed multigraph. The proposed DP labeling algorithm incorporates strong completion bounds and is designed to heuristically terminate early, thereby accelerating computation. The BPC algorithm integrates subset-row cuts and capacity cuts, with three different separation techniques developed for the latter. A two-stage branching scheme, including Ryan-and-Foster branching, is implemented. The DP labeling approach accounts for both non-robust cuts and branching decisions. In addition, leveraging

the powerful CG component, two BPC-based heuristic methods are introduced for the OBP. Computational experiments demonstrate that the proposed exact BPC approach substantially outperforms existing state-of-the-art exact solution methods. Moreover, the heuristics show significant improvements in gap quality compared to current leading heuristic approaches for the single-block OBP.

Chapter 3 addresses the multi-block OBP by extending the BPC approach introduced in Chapter 2, incorporating suitable picker routing strategies. This work pioneers the application of a BPC method to the multi-block OBP and is the first to provide an exact solution approach for heuristic routing strategies to this problem. A key contribution of this study is the comprehensive examination of the monotonicity properties of the multi-block routing strategies optimal, no-reversal, aisle-by-aisle, traversal, combined, and largest gap. This analysis is pivotal for the application of the BPC algorithm. Computational experiments further confirm the efficiency of the BPC-based approaches, showcasing their superior performance over both exact and heuristic state-of-the-art methods for the multi-block OBP.

Chapter 4 presents the first B&P approach for the SUBP. The corresponding CG pricing problem, which is a *set-union knapsack problem*, is formulated in three alternative ways: an IP, an item-based SPPRC, and an element-based SPPRC. Both SPPRC formulations are addressed using DP labeling algorithms that forgo dominance rules but incorporate sophisticated completion bounds. Additionally, a greedy pricing heuristic is developed, significantly reducing the overall computational effort. The most effective pricing strategy identified combines the initial greedy heuristic with the item-based labeling algorithm, which is also compatible with the Ryan-and-Foster branching technique. The proposed B&P approach by far outperforms current state-of-the-art IP formulations and improves the best-known solutions for more than half of the considered benchmark instances.

Chapter 5 provides a summary of the findings and concludes the thesis.

Chapter 2

Branch-Price-and-Cut-Based Solution of Order Batching Problems

Julia Wahlen and Timo Gschwind

Abstract

Given a set of customer orders each comprising one or more individual items to be picked, the *order batching problem* (OBP) in warehousing consists of designing a set of picking batches such that each customer order is assigned to exactly one batch, all batches satisfy the capacity restriction of the pickers, and the total distance traveled by the pickers is minimal. In order to collect the items of a batch, the pickers traverse the warehouse using a predefined routing strategy. We propose a *branch-price-and-cut* (BPC) algorithm for the exact solution of the OBP investigating the routing strategies traversal, return, midpoint, largest gap, combined, and optimal. The column generation pricing problem is modeled as a *shortest path problem with resource constraints* (SPPRC) which can be adapted to handle the implications from non-robust valid inequalities and branching decisions. The SPPRC pricing problem is solved by means of an effective labeling algorithm that relies on strong completion bounds. Capacity cuts and subset-row cuts are used to strengthen the lower bounds. Furthermore, we derive two BPC-based heuristics to identify high-quality solutions in short computation times. Extensive computational results demonstrate the effectiveness of the proposed methods. The BPC is faster by two orders of magnitude compared to the state-of-the-art exact approach and can solve to optimality hundreds of instances that were previously unsolved. The BPC-based heuristics are able to significantly improve the gaps reported for the state-of-the-art heuristic and provide hundreds of new best-known solutions.

2.1 Introduction

Warehousing is an essential part of a supply chain and involves receiving, storing, picking, packing and shipping operations. It makes up about 20-25% of overall logistics costs (Establish Inc. 2013). With a growing trend of online shopping and e-commerce, the importance of efficient warehousing processes is likely to further increase. The performance of warehouse operations is impacted to a large extend by decisions regarding warehouse layout and technology, zoning, storage assignment, and order picking. For a more detailed overview, we refer to the extensive surveys (de Koster *et al.* 2007, Gu *et al.* 2010, Boysen *et al.* 2019).

In many warehouses, order picking, i.e., the process of retrieving articles from their storage locations according to customer orders, is still done manually because unlike automated systems humans can adapt to changes in real time (Grosse *et al.* 2014). Michel (2016), e.g., report that automated parts-to-picker systems are used in less than 10% of the warehouses. In Western Europe, so-called low-level picker-to-parts warehouses are predominant and account for the vast majority of all warehouses (de Koster *et al.* 2007, Marchet *et al.* 2015) while the number of fully automated warehouses is estimated to be only about 40 (Azadeh *et al.* 2019). Low-level picker-to-parts warehouses describe systems in which items are stored in shelves less than two meters high, and the pickers walk or ride along the aisles to collect the items specified on a picking list (Caron *et al.* 2000). In particular in these types of warehouses, order picking is highly labor-intensive and constitutes one of the most cost-intensive operations in warehousing, accounting for more than 50% of the total operating costs (Frazelle 2001, Tompkins *et al.* 2010, Richards 2017).

Within the order-picking process, the traveling of the pickers is a crucial activity being the main factor responsible for the overall picking time (accounting for up to 50%). The other activities' times are either considered constant, e.g., searching and picking time, or negligible, e.g., setup time (de Koster *et al.* 1999a, Tompkins *et al.* 2010). Assuming that the speed of the pickers is constant, minimizing the travel time is equivalent to minimizing the length of all order picking routes.

On an operational level, the order picking process is significantly influenced by picker route planning and order batching (de Koster *et al.* 2007). Given a warehouse layout, storage locations of all items, and a list of items to pick, *picker route planning*, or short *picker routing*, describes the problem of how a single picker should move through the warehouse to collect all items on the picking list while minimizing the distance traveled. In their seminal work, Ratliff and Rosenthal (1983) proposed a *dynamic programming* (DP) algorithm that exactly solves the picker routing problem in a single-block warehouse with parallel aisles and whose complexity is linear in the sum of the number of aisles and the number of picking positions (Heßler and Irnich 2022b). Besides an optimal picker routing, several heuristic picker routing strategies have been proposed in the literature, including

traversal (or s-shape) (Goetschalckx and Ratliff 1988), return, midpoint, largest gap (Hall 1993), composite (Petersen 1995), combined (Roodbergen 2001), and mixed (Bahçeci and Öncan 2022). Such simple, rule-based strategies are often encountered in practice, because they are typically more intuitive for the pickers and may exhibit less risk of in-aisle congestion than an optimal routing (de Koster *et al.* 1999b). More recent works on picker routing have generalized these routing strategies (we include optimal routing in this term) to different warehouse layouts (e.g., Roodbergen and de Koster 2001b, Çelik and Süral 2014, Pansart *et al.* 2018) or other features like scattered storage and a decoupling of picker and cart (Goeke and Schneider 2021).

Order batching is relevant whenever a batch picking strategy is pursued to fulfill customer orders. Batch picking is one of the two basic order picking methods in picker-to-parts warehouses, the other being single order picking (Petersen and Aase 2004). As opposed to single order picking where pickers collect one order at a time, batch picking (or multi-order picking) allows pickers to fulfill multiple customer orders that are combined into a picking batch in a single picking route. Typically, customer orders should not be split and collected in different picking routes, as splitting might lead to an unreasonable sorting effort. While single order picking is the most common order picking method, batch picking has been shown to reduce total picking times significantly (de Koster *et al.* 1999b) by decreasing the total travel distance of pickers not only through reducing the number of trips but also by shortening the length of each trip (Hong *et al.* 2012). The task of profitably combining individual customer orders into picking batches is formalized by the *order batching problem* (OBP). Given a warehouse layout, storage locations of all items, and a set of customer orders each comprising one or more individual items, the OBP consists of designing a set of picking batches such that each customer order is assigned to exactly one batch, all batches satisfy the capacity restriction of the pickers, and the total distance traveled by the pickers is minimal. In order to collect the items of a batch, pickers traverse the warehouse using a predefined routing strategy. For the optimal routing strategy, this gives rise to an integrated planning problem coined the *joint order batching and picker routing problem* (JOBPRP, Valle *et al.* 2016).

In this paper, we focus on the exact solution of the OBP in a rectangular single-block parallel-aisles warehouse. We refer to this setup as the *standard* OBP. As routing strategies, we consider traversal, return, midpoint, largest gap, combined, and optimal. Computationally, the OBP is a challenging problem. On the theoretical side, it has been shown to be $\mathcal{NP}$-hard when the number of orders per batch is greater than two (Gademann and van de Velde 2005). On the practical side, a key difficulty for solution approaches to the OBP is the fact that the travel distances of the batches are given by a function that is not separable in the

comprised orders. This is true for all routing strategies that we investigate. Feasibility of the batches, on the other hand, depends solely on a standard knapsack constraint. The main contribution of this paper is the development of powerful exact and heuristic solution procedures to the OBP. They are tested on several sets of benchmark instances for all six considered routing strategies and are able to significantly outperform the state-of-the-art exact and heuristic algorithms from the literature.

2.1.1 Literature Review

For a general review of solution approaches to the OBP including construction heuristics, metaheuristics, and exact algorithms, we refer to the extensive surveys of de Koster *et al.* (2007) and Henn *et al.* (2012). An overview of more recent heuristic approaches as well as related variants and extensions can be found in (Žulj *et al.* 2018). In the following, we focus on the literature on exact solution algorithms which is still limited, despite the high practical relevance of the OBP and a growing interest in recent years.

Exact approaches to the OBP have been proposed by Gademann and van de Velde (2005), Öncan (2015), Muter and Öncan (2015), Valle *et al.* (2016) and Valle *et al.* (2017), and Bahçeci and Öncan (2022). Table 2.1 summarizes their main characteristics, including our method.

Gademann and van de Velde (2005) consider a special case of the standard OBP with optimal routing, in which all customer orders have unit weights. The knapsack constraint of the batches, thus, reduces to a cardinality constraint (let c be the maximum number of orders in a batch). They formulate the OBP as a set-partitioning problem that is solved by means of a *branch-and-price* (B&P) algorithm. The *column generation* (CG) pricing problem of their approach, which consists of finding batches with negative reduced costs, is solved with a combinatorial *branch-and-bound* (B&B) algorithm. In each level of the B&B tree, they decide on the inclusion of one additional order into a batch. Thus, the maximum level of the B&B tree equals c. A simple lower bound estimating the maximum collectable dual prices of the still undecided orders is used to prune unpromising B&B nodes. The well-known Ryan-and-Foster branching rule is applied to guarantee integer solutions. Problem instances with up to 32 orders and $c = 10$ are solved to proven optimality.

Öncan (2015) considers the standard OBP with routing strategies traversal, midpoint and return. Besides an iterated local search metaheuristic, the author derives three *mixed-integer programming* (MIP) formulations each dedicated to one of the three OBP variants specified by the respective routing strategy. The MIPs are tested using a general-purpose MIP solver and three different classes of instances including those from the benchmark by Henn and Wäscher (2012) which

comprises instances with between 20 and 100 orders and a capacity between 30 and 75. Within a time limit of three hours, a small fraction of the smaller instances of the benchmark are solved to optimality.

The standard OBP with the same three routing strategies traversal, midpoint and return is also addressed by Muter and Öncan (2015). They propose a cut-and-column generation approach with batch enumeration based on the same set-partitioning formulation used by Gademann and van de Velde (2005). Their CG pricing problem, however, involves a knapsack constraint instead of the simpler cardinality constraint. For its solution, a DP labeling algorithm on a linear network is proposed. Due to the non-separability of the travel-distance function, no dominance relations between labels can be exploited. To eliminate unpromising labels, they derive simple lower bounds generalizing the idea of Gademann and van de Velde (2005) to the knapsack constraint: For each label, the LP relaxation of a knapsack problem involving the yet undecided orders and their capacities and dual prices is solved. To speed-up the CG process, a column pool comprising promising batches of previous pricing iterations is maintained and checked for negative reduced-cost batches before calling the computationally expensive labeling algorithm in each pricing iteration. Subset-row cuts are employed to strengthen the set-partitioning formulation. To reach optimal integer solutions, Muter and Öncan (2015) rely on a technique proposed by Baldacci *et al.* (2011) for vehicle routing problems. Using upper bounds from the iterated local search of Öncan (2015), they try to enumerate all batches with reduced costs smaller than the optimality gap and solve the resulting reduced set-partitioning problem over all these batches using a general-purpose MIP solver. They test their approach on a newly generated testbed comprising instances with 20 to 100 orders and capacities 24, 36, and 48. They are able to solve most of the smaller and some of the larger instances with capacity 24, and very few of the smallest instances with capacity 36 and 48.

Valle *et al.* (2016) propose three MIP formulations for an OBP with optimal routing in a rectangular parallel-aisles warehouse with multiple blocks divided by cross aisles. They propose three MIP formulations, one with exponentially many generalized subtour breaking constraints, the other two being compact formulations based on network flows. The former is solved by a *branch-and-cut* (B&C) algorithm, the latter two by a general-purpose MIP solver. In (Valle *et al.* 2017), the same authors improve their B&C algorithm for the non-compact formulation by introducing several families of valid inequalities based on a graph representation of the warehouse. With their improved B&C algorithm, instances for a two-block warehouse with up to 20 orders and a picking capacity of 40 can be solved optimally.

The recent work of Bahçeci and Öncan (2022) considers the standard OBP and proposes dedicated MIP formulations for composite, largest gap, optimal, and

the newly introduced mixed routing strategies. To evaluate the routing strategies and different storage assignment policies, they generate a new set of very small instances with eight and 12 orders and a picking capacity between five and 30. A general-purpose MIP solver is used to solve the proposed MIPs and the MIPs of Öncan (2015) for routing strategies traversal, return, and midpoint for the new instances.

Aerts *et al.* (2021) show that the JOBPRP can be modeled as a *soft-clustered vehicle routing problem* (SoftCluVRP), a variant of the well-known *vehicle routing problem* in which customers are grouped into clusters. Any exact approach to the SoftCluVRP (e.g., Hintsch and Irnich 2020, Heßler and Irnich 2021) can in principle be used to solve the JOBPRP. On the downside, these approaches do not account for the warehouse layout.

2.1.2 Contributions

The main contribution of this paper is the development of a powerful and flexible exact solution approach to the OBP. Furthermore, we derive two effective heuristics based on the exact algorithm. Extensive computational experiments demonstrate the competitiveness of the proposed methods. We elaborate on the main contributions in the following.

The proposed exact approach is, to the best of our knowledge, the first full-fledged BPC algorithm for the OBP with general weights. While the focus of this paper is on the standard OBP and routing strategies traversal, return, midpoint, largest gap, combined, and optimal, the proposed BPC is much more generic. Indeed, it can directly be applied to any warehouse layout and routing strategy, or to other features like scattered storage or decoupling of picker and picking cart as long as (i) a method for solving the corresponding picker routing problem for a given batch is available and (ii) the travel distances of the picker routes are given by a function that is monotone in the orders to be picked.

Our BPC algorithm is based on the same set-partitioning formulation that has been used for tackling the OBP (Gademann and van de Velde 2005, Muter and Öncan 2015) and related problems like vehicle routing or bin packing with CG-based methods like BPC. However, BPC is not an out-of-the-box solver. Crucial for its effectiveness are typically (i) the effective solution of the CG pricing problem and (ii) good strategies to obtain integer solutions using cuts to strengthen the formulation and suitable branching rules. All of these building blocks are highly problem specific and constitute major developing issues in the design of a BPC approach. The core components of the developed method are as follows.

- We model the CG pricing problem as a *shortest path problem with resource constraints* (SPPRC), which provides the flexibility to also handle the implications from non-robust valid inequalities and branching decisions.

Article	Warehouse and picking features		Algorithmic features		Benchm.
	Routing strat.	Other features	Main approach	Other features	
Gademann and van de Velde (2005)	optimal	single-block, unit weights	CG/B&P	pricing prob. sol.: combinatorial B&B, Ryan-and-Foster branching	GBH
Öncan (2015)	traversal, return, midpoint	single-block, general weights	MIP solver		H&W, Ö
Muter and Öncan (2015)	traversal, return, midpoint	single-block, general weights	CG/cuts/ MIP solver	pricing prob. sol.: DP with bounding, batch pool, subset-row cuts, batch enumeration and MIP solver	M&Ö
Valle *et al.* (2017)	optimal	two-block, general weights	B&C	generalized subtour breaking constraints, cuts based on warehouse layout	Foodm
Bahçeci and Öncan (2022)	composite, largest gap, mixed, optimal	single-block, general weights	MIP solver		B&Ö
Our approach	traversal, return, midpoint, largest gap, combined, optimal	single-block, general weights	CG/BPC	pricing prob. sol.: DP with bounding, partial pricing, capcity cuts, subset-row cuts, Ryan-and-Foster branching, strong branching, BPC-based heuristics	H&W, M&Ö, W&G, ZKS

B&Ö: Bahçeci and Öncan (2022), num. orders: 8–12, capacity: 5–30;
Foodm: Thia (2008), num. orders: 5–30, capacity: 40;
GBH: Gademann *et al.* (2001), num. orders: 15–32, capacity: 3–10;
H&W: Henn and Wäscher (2012), num. orders: 20–100, capacity: 30–75;
M&Ö: Muter and Öncan (2015), num. orders: 20–100, capacity: 24–48;
Ö: Öncan (2015), num. orders: 10–100, capacity: 24–48;
W&G: this paper, num. orders: 125–250, capacity: 24–72;
ZKS: Žulj *et al.* (2018), num. orders: 200–600, capacity: 6–15

Table 2.1: Overview of exact solution approaches to the OBP

- We derive strong completion bounds that enable the effective solution of the SPPRC with a labeling algorithm. These bounds are adapted to account for the implications of cutting and branching decisions and to remain tight deeper in the search tree of the BPC.
- We introduce a highly effective pricing heuristic based on the premature termination of the labeling algorithm.
- We employ two families of non-robust valid inequalities: *capacity cuts* (CCs) and *subset-row cuts* (SRCs). For the separation of CCs, which are particularly effective in strengthening the lower bounds, two heuristics and a MIP formulation are derived.

The proposed CG method is very effective in solving the LP relaxation of the set-partitioning formulation of the OBP. Closing the gap and finding an optimal integer solution is by far the most time consuming part of the BPC. We exploit this fact and derive two very effective BPC-based heuristics for the OBP.

Finally, extensive computational experiments on three large sets of benchmark instances from the literature and on newly created larger-sized instances are reported:

- We provide an in-depth computational analysis of the BPC algorithm and its components.
- Compared to the state-of-the-art exact solution approach of Muter and Öncan (2015), who only consider routing strategies traversal, return, and midpoint, our approach is faster by about two orders of magnitude and provides more than three times the number of proven optima. Our BPC is able to solve 90% of the two benchmarks by Henn and Wäscher (2012) and Muter and Öncan (2015). Only a small fraction of these instances has been solved to proven optimality before.
- With both our heuristics, we are able to drastically improve on the gaps of the state-of-the-art heuristic solution approach of Žulj *et al.* (2018), who consider only routing strategies traversal and largest gap. We improve almost 2,000 out of the 2,720 *best-known solutions* (BKS) reported by Žulj *et al.* (2018), and confirm all remaining BKS except for two instances.
- We provide managerial insights on the quality of the six considered routing strategies when applying optimal order batching decisions.

2.1.3 Organization of the Paper

The remainder of the paper is structured as follows. Section 2.2 formally defines the OBP, presents a set-partitioning formulation of the problem, and specifies the warehouse layout and routing strategies that we consider. The details of our exact BPC algorithm and the BPC-based heuristics are given in Section 2.3. Section 2.4 presents our computational results. Final conclusions are drawn in Section 2.5.

2.2 Problem Description and Mathematical Formulation

Problem Definition Let $O = \{1, ..., n\}$ be the set of customer orders. Each order $o \in O$ comprises a set of individual items to be picked. A sufficiently large number of pickers with an identical picking capacity Q is available to pick the ordered items. The capacity consumption of an order $o \in O$ is $q_o \geq 0$. The capacity consumption of the individual items is not relevant, because splitting of the orders is not allowed.

The OBP consists of grouping the customer orders into picking batches, i.e., subsets of the customer orders, such that each customer order is assigned to exactly one batch, each batch satisfies the capacity of the pickers, and the total distance traveled is minimal. Thereby, each batch is assigned to a single picker that walks through the warehouse and collects all items of the orders assigned to that batch. The distance traveled by the pickers depends on the warehouse layout, the storage locations of the items as well as the routing strategy used to traverse the warehouse, all of which are assumed to be fixed a priori.

Set-Partitioning Formulation To formulate the OBP as a set-partitioning problem, let Ω be the set of all feasible batches. Binary parameters a_{ob} indicate if order $o \in O$ is contained in batch $b \in \Omega$ ($a_{ob} = 1$) or not ($a_{ob} = 0$). The distance needed to pick all individual items of a batch $b \in \Omega$ is given by c_b. Note that the distance function c_b is not separable in the orders $o \in b$ (see Section 2.3.1). Finally, let λ_b be binary decision variables equal to one if batch $b \in \Omega$ is selected and zero otherwise. Then the OBP can be formulated as follows:

$$\min \sum_{b \in \Omega} c_b \lambda_b \tag{2.1a}$$

$$\text{s.t.} \sum_{b \in \Omega} a_{ob} \lambda_b = 1 \quad \forall o \in O \tag{2.1b}$$

$$\lambda_b \in \{0, 1\} \quad \forall b \in \Omega \tag{2.1c}$$

The Objective (2.1a) minimizes the total traveled distance while Constraints (2.1b) ensure that all orders are picked exactly once.

Because the number of feasible batches $|\Omega|$ is generally too large, Formulation (2.1) cannot be solved directly. Instead, we resort to a BPC algorithm whose details are presented in Section 2.3.

Warehouse Layout, Storage Locations, and Routing Strategies The standard OBP considers a rectangular warehouse with parallel aisles of equal length and width. A top view of the layout is illustrated in Figure 2.1. Cross aisles at the

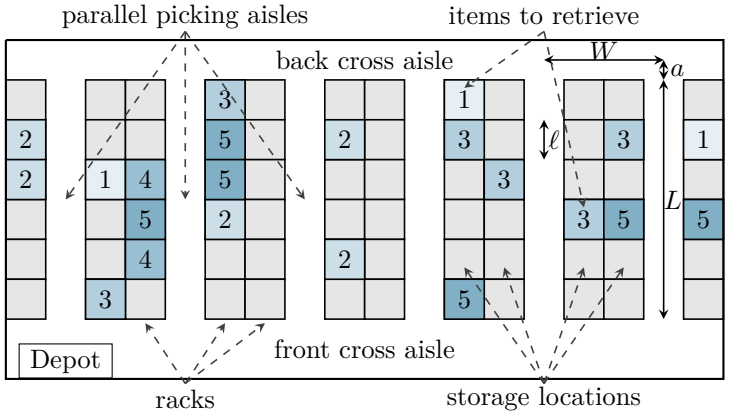


Figure 2.1: Rectangular parallel-aisles single-block warehouse layout of the standard OBP

front and at the back of the warehouse connect the parallel vertical picking aisles. The picking aisles are two-sided, i.e., there are racks on the left and on the right side of each aisle (from a top view). Each rack consists of several storage locations depicted by the gray squares. Each storage location may contain several items, but every item is assigned to a single storage location. The start and end of each picker route is a common depot which is located in front of the leftmost vertical aisle. The pickers can travel in both directions in all picking aisles and both cross aisles. Turnarounds are allowed everywhere.

Figure 2.1 depicts an instance of the standard OBP with $n = 5$ orders each comprising up to six individual items which are labeled with the number of the corresponding order $o \in \{1, \ldots, 5\}$. We assume that the pickers always travel in the horizontal middle of the picking aisles and that the retrieval of the items is performed from the vertical middle of a storage location without the need of a horizontal movement. The length of the storage locations is $\ell = 1$ so that the length of the racks is $L = 6$. The horizontal distance between two picking aisles is $W = 3$. To enter or leave any picking aisle from the front or back cross aisle, a distance of $a = 0.5$ units has to be traveled.

The main routing strategies that have been considered for the parallel-aisles single-block warehouse specified above are traversal, return, midpoint, largest gap, composite, combined, and optimal. In Appendix 2.A, we provide a detailed description of each strategy to clarify our exact interpretation and to allow reproduction of our results. A general trend for all routing strategies is that larger batches comprising more items to pick tend to cause longer picker routes. We formalize this observation as a property of the routing strategy in the following definition.

Definition 2.1. *A routing strategy is* monotone*, if the corresponding distance function c_b is monotone, i.e., if for any two feasible batches $b_1 \subseteq b_2$ it follows that*

$c_{b_1} \leq c_{b_2}$.

While being monotone may also be a desirable property for a routing strategy from a practical point of view, it certainly has important algorithmic implications. If a monotone routing strategy is applied, partitioning constraints (2.1b) can be replaced by the corresponding covering constraints in Formulation (2.1). This is a common technique in CG-based approaches to (extended) set-partitioning problems, because the set-covering counterpart is typically easier to solve. Furthermore, monotonicity of the routing strategy is a requirement for the validity of our bounding procedure to solve the CG pricing problem (see Section 2.3.1.3). The following proposition shows that all mentioned routing strategies except composite are monotone.

Proposition 2.1. *The routing strategies return, midpoint, traversal, largest gap, combined, mixed, and optimal are monotone.*

The proof of Proposition 2.1 is provided in Appendix 2.B. A small counterexample for the composite strategy is given in Appendix 2.C.

2.3 Branch-Price-and-Cut

A BPC algorithm is a B&B algorithm in which the lower bounds are computed by CG and cuts are added dynamically to strengthen the linear relaxations. CG is an iterative procedure that can tackle linear programs containing a huge number of variables. The starting point of our BPC algorithm is the *restricted master problem* (RMP) which is the linear relaxation of Formulation (2.1) defined over a small subset $\Omega' \subset \Omega$ of batches. The CG algorithm then alternates between the reoptimization of the RMP and the solution of the pricing problem to dynamically generate missing batches with negative reduced costs and add them to the RMP, if any exist. If no negative reduced-cost batch exists, an optimal solution to the current RMP is found. The corresponding lower bound can be strengthened by adding valid inequalities. Branching is required to finally ensure integer solutions. For details on CG and BPC, we refer to (Barnhart *et al.* 1998, Lübbecke and Desrosiers 2005).

2.3.1 Pricing Problem

Let π_o be the dual price associated with Constraints (2.1b). The reduced cost of a batch b is given by $\tilde{c}_b = c_b - \sum_{o \in b} \pi_o$. The pricing problem consists of identifying at least one feasible batch $b \in \Omega$ with negative reduced cost or to guarantee that no such batch exists.

Recall that the distance function c_b is a function that is not separable in the orders of b, meaning that c_b always depends on the entire set of orders in b and cannot be calculated, e.g., as the sum or product of some individual values of the comprised orders. Because c_b is a non-separable function in the orders, the reduced costs $\tilde{c}_b$ of the batches are also not separable in the orders, which constitutes a core difficulty in designing effective solution approaches to the pricing problem. This is particularly true for combinatorial algorithms that build batches in an incremental fashion.

2.3.1.1 SPPRC Formulation of the Pricing Problem

The pricing problem is modeled as an SPPRC as follows. Let $G = (V, A)$ be a linear directed multigraph with $n + 1$ vertices $V = \{0, \dots, n\}$ and $2n$ arcs A. Vertex 0 is an artificial source. Vertices $1, \dots, n$ correspond with the n customer orders in any given sorting. For ease of notation, we assume throughout this section that in the SPPRC graph the orders are sorted by their number, meaning that vertex $v \in V \setminus \{0\}$ corresponds with order $o = v$. For each vertex $v \in V \setminus \{0\}$, there are two parallel arcs a_v^1 and a_v^0 connecting vertices $v-1$ and v and indicating the inclusion or not, respectively, of order v. Each arc $a_v^k \in A,\ k \in \{0, 1\}$ is associated with a capacity consumption q_v^k, a dual price π_v^k, and a set of orders O_v^k. Accordingly, for arc a_v^1 we have $q_v^1 = q_v$, $\pi_v^1 = \pi_v$ and $O_v^1 = \{v\}$, while for arc a_v^0 we have $q_v^0 = \pi_v^0 = 0$ and $O_v^0 = \varnothing$. Associating sets of orders O_v^k with the arcs allows the simultaneous consideration of multiple orders which is needed for the incorporation of branching decisions in the pricing (see Section 2.3.3).

Any v_0-v_p-path $(v_0, a_{v_1}^{k_1}, v_1, \dots, a_{v_p}^{k_p}, v_p)$ in G defines a batch $b = \bigcup_{i=1}^{p} O_{v_i}^{k_i}$. It is feasible, if $\sum_{i=1}^{p} q_{v_i}^{k_i} \leq Q$. The reduced cost of batch b is $c_b - \sum_{i=1}^{p} \pi_{v_i}^{k_i}$. The solution of the pricing problem is equivalent to finding a capacity-feasible 0-n-path in G with minimum reduced cost. Figure 2.2 illustrates the graph G for the example OBP instance of Figure 2.1 and dual prices π_o. There are two arcs between each pair of consecutive vertices, indicating the inclusion (blue arc) or not (gray arc) of the order associated with the respective head vertex. Consider vertex $v = 1$. The ingoing blue arc $(3, \pi_1, \{1\}) = a_1^1$ represents the inclusion of the singleton order set $O_1^1 = \{1\}$ into a batch and is associated with the order's capacity consumption of $q_1 = 3$ items and its dual price π_1. The ingoing gray arc $(0, 0, \varnothing) = a_1^0$ corresponds with not including any order ($O_1^0 = \varnothing$) and therefore $q_1^0 = \pi_1^0 = 0$.

This SPPRC representation has also been used for pricing problems with a similar knapsack-type structure (e.g., Heßler *et al.* 2018, Gschwind *et al.* 2019). The main advantage of the SPPRC representation of the pricing problem is its flexibility: Slight modifications of the underlying graph G and/or the labeling algorithm for its solution suffice to account for typical cutting and branching decisions of

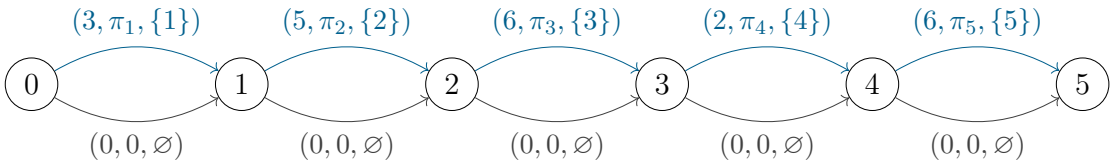


Figure 2.2: SPPRC representation of the pricing problem: linear directed multigraph for the example in Figure 2.1 and dual prices π_o

BPC algorithms as detailed in Sections 2.3.2 and 2.3.3. Notice that the proposed SPPRC is different from the pricing problem representation of Muter and Öncan (2015).

2.3.1.2 Basic Labeling Algorithm

SPPRCs are typically solved with DP labeling algorithms (Irnich and Desaulniers 2005). In a labeling algorithm, partial paths are iteratively extended from a given source to a given sink. The partial paths are implicitly represented by labels storing the accumulated resource consumption along the paths. The propagation of the labels along the network arcs is realized using so-called *resource extension functions* (REFs). To avoid enumerating all feasible paths, dominance relations between labels to eliminate provably non-optimal paths and bounding procedures to discard unpromising paths that cannot reach a given objective value threshold can be applied.

In the OBP, a partial path $P_E = (0, a_1^{k_1}, 1, \ldots, a_v^{k_v}, v)$ from the source 0 to some vertex v is represented by a label $E = (v(E), q(E), \pi(E), O(E), \tilde{c}(E))$ storing its last vertex $v(E)$, its accumulated capacity consumption $q(E)$ and dual price $\pi(E)$, the set of orders $O(E)$ it comprises, and its reduced cost $\tilde{c}(E)$. The initial label at the artificial source 0 is given by $(0, 0, 0, \varnothing, 0)$. Because of the linear nature of G, labels are processed vertex-by-vertex in our labeling algorithm. This means that starting with the initial label at the artificial source, we always propagate all labels at a given vertex $v-1$ along the arcs a_v^0 and a_v^1 to vertex v, before all resulting labels at vertex v are in turn propagated to vertex $v+1$, etc., until finally vertex n is reached and 0-n-paths result.

The extension of a label E at vertex $v-1$ to vertex v along arc a_v^k is feasible, if $q(E) + q_v^k \leq Q$. If the extension is feasible, a new label E' is created according to

the following REFs:

$$v(E') = v \tag{2.2a}$$
$$q(E') = q(E) + q_v^k \tag{2.2b}$$
$$\pi(E') = \pi(E) + \pi_v^k \tag{2.2c}$$
$$O(E') = O(E) \cup O_v^k \tag{2.2d}$$
$$\tilde{c}(E') = c_{O(E')} - \pi(E') \tag{2.2e}$$

The non-separability of the reduced cost $\tilde{c}(E')$ in the orders imposes two major drawbacks on the labeling algorithm. First, in every label propagation, a costly evaluation of the distance function $c_{O(E')}$ is necessary in REF (2.2e). Second, it renders infeasible the standard *less-or-equal* dominance relation of the reduced cost resource applied in many labeling algorithms for SPPRC variants.

2.3.1.3 Bounding Procedure

Because no dominance rule is applied in our labeling algorithm, a strong bounding procedure to eliminate unpromising labels is crucial for its effectiveness. Let $LB(E)$ be a lower bound on the reduced cost of any capacity-feasible 0-n-path in G that contains the 0-$v(E)$-path P_E corresponding to label E. Obviously, any label E with $LB(E) \geq 0$ can be discarded. In this section, we describe a method for computing values $LB(E)$ that is applicable for any monotone routing strategy and that can be adapted to cope with the cutting and branching decisions of our BPC algorithm.

For a label E, let $R(E)$ be the set of $v(E)$-n-paths that can be appended to the 0-$v(E)$-path P_E to form capacity-feasible 0-n-paths. A path $r \in R(E)$ is called a completion of E. Denote by E^r the label corresponding to path (P_E, r) and let $O(r) := O(E^r) \setminus O(E)$. For any monotone routing strategy, it holds that $\tilde{c}(E^r) = c_{O(E^r)} - \pi(E^r) \geq c_{O(E)} - \pi(E^r) = \tilde{c}(E) - \sum_{o \in O(r)} \pi_o$. Thus, a valid lower bound $LB(E)$ on the reduced cost of any capacity-feasible 0-n-path containing path P_E is given by

$$LB(E) := \tilde{c}(E) - \max_{r \in R(E)} \sum_{o \in O(r)} \pi_o. \tag{2.3}$$

Intuitively speaking, the value $\max_{r \in R(E)} \sum_{o \in O(r)} \pi_o$ represents the maximum dual prices that can be collected when extending label E to a 0-n-path. Because set $R(E)$ comprises all completions r such that $q(E^r) = q(E) + \sum_{o \in O(r)} q_o \leq Q$, this value is equivalent to the optimal solution value of a binary *knapsack problem* (KP) over orders $\{o \in O : o > v(E)\}$ with weights q_o, profits π_o, and capacity $Q - q(E)$. Note further that the sets $R(E)$ are identical for all labels E with the same capacity

consumption $q := q(E)$ and last vertex $v := v(E)$. Therefore, we can define associated completion bounds $B_v(q) := \max_{r \in R(E)} \sum_{o \in O(r)} \pi_o$ that are identical for all such labels E and depend only on v and q.

We compute the completion bounds $B_v(q)$ for all $v \in V$ and $q \in \{0, \ldots, Q\}$ by solving a single binary KP over all orders O and capacity Q using a labeling algorithm that runs in pseudo-polynomial time $\mathcal{O}(nQ)$. The labeling algorithm is applied on the SPPRC graph G of the pricing problem described in Section 2.3.1.1 in backward direction. In a backward labeling algorithm, backward paths are gradually extended from the sink to the source.

A backward path $(v, a_v^{k_v}, \ldots, a_n^{k_n}, n)$ of the labeling algorithm for solving the binary KP is represented by a label $E_{kp} = (v(E_{kp}), q(E_{kp}), \pi(E_{kp}))$ storing its first vertex $v(E_{kp})$ and its accumulated weight $q(E_{kp})$ and profit $\pi(E_{kp})$. The initial label at vertex n is given by $(n, 0, 0)$. The extension of a label E_{kp} starting at vertex v against the orientation of arc a_v^k to vertex $v-1$ is feasible if $q(E_{kp}) + q_v^k \leq Q$. If the extension is feasible, a new label $E'_{kp} = (v(E'_{kp}), q(E'_{kp}), \pi(E'_{kp}))$ with $v(E'_{kp}) = v - 1$, $q(E'_{kp}) = q(E_{kp}) + q_v^k$, and $\pi(E'_{kp}) = \pi(E_{kp}) + \pi_v^k$ is created.

A label E_{kp} dominates another label E'_{kp} starting at the same vertex if

$$q(E_{kp}) \leq q(E'_{kp}), \tag{2.4a}$$

$$\pi(E_{kp}) \geq \pi(E'_{kp}). \tag{2.4b}$$

All dominated labels can be discarded as long as, for each of them, at least one dominating label is kept.

By its termination, the labeling algorithm provides at each vertex $v \in V$ a set $\mathcal{E}_v$ of undominated labels corresponding with Pareto-optimal knapsack packings. More precisely, an undominated label $E_{kp} \in \mathcal{E}_v$ corresponds with the optimal solution of a binary KP over orders $\{o \in O : o > v(E_{kp}) = v\}$ with weights q_o, profits π_o, and capacity $q(E_{kp})$. The completion bounds can then be determined as

$$B_v(q) = \max_{E_{kp} \in \mathcal{E}_v : q(E_{kp}) \leq Q - q} \pi(E_{kp}). \tag{2.5}$$

For each vertex v, the corresponding bounding function $B_v(q)$ is a non-increasing step function with $|\mathcal{E}_v|$ constant pieces.

Summing up, in each pricing iteration the solution approach proceeds as follows. Before invoking the main labeling algorithm, we solve the single binary KP on graph G in backward direction to set up the bounding functions $B_v(q)$. Then, in the forward labeling process of the main algorithm, any label E with $\tilde{c}(E) \geq B_{v(E)}(q(E))$ is immediately discarded.

An example of the pricing procedure for the OBP instance of Figure 2.1 is illustrated in Figure 2.3. The picking capacity is assumed to be $Q = 8$, the dual

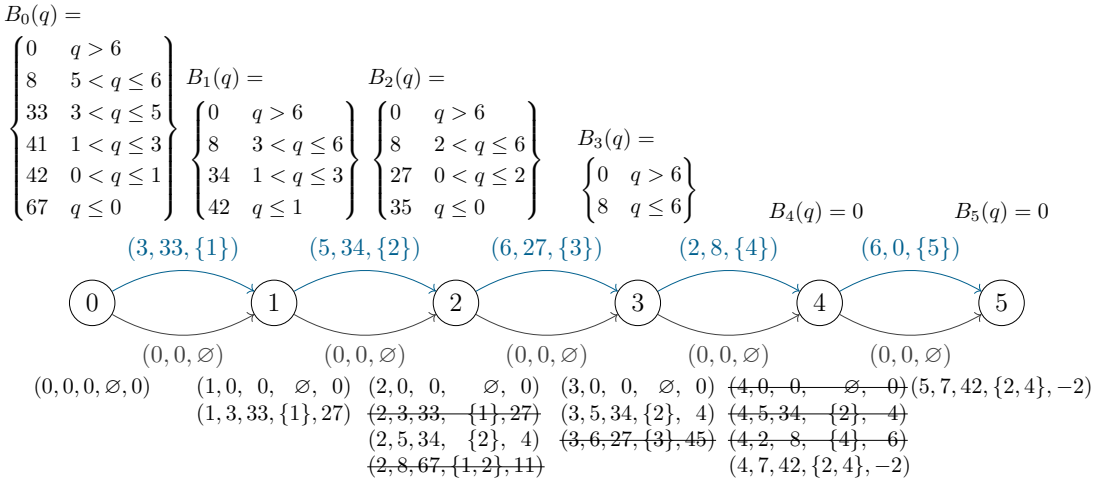


Figure 2.3: Exemplary pricing procedure with labels and completion bounds for OBP instance of Figure 2.1

prices for the orders are $\pi_1 = 33$, $\pi_2 = 34$, $\pi_3 = 27$, $\pi_4 = 8$, and $\pi_5 = 0$. The return routing strategy is applied and the travel distances of the arising batches are $c_{\{1\}} = 60$, $c_{\{2\}} = 38$, $c_{\{1,2\}} = 78$, $c_{\{3\}} = 72$, $c_{\{4\}} = 14$, $c_{\{2,4\}} = 40$, and $c_{\{5\}} = 48$. For each vertex v, the corresponding bounding functions $B_v(q)$ are given above the vertex. For example, at vertex 3 the backward labeling for the binary KP provides two non-dominated knapsack packings: packing only order 4 with a weight (=capacity consumption) of two and a profit (=dual price) of eight and packing neither of the orders for zero weight and profit. Accordingly, the completion bound at vertex 3 is eight for labels with a capacity consumption up to six units (implying a residual capacity of at least $8 - 6 = 2$) while it is zero for labels with a capacity consumption larger than six. The labels of the main labeling algorithm are given below the corresponding vertices. Labels that are discarded due to the bounding procedure are crossed out, capacity-infeasible labels are not created in the algorithm and therefore not shown in the figure. Consider label $E = (1, 3, 33, \{1\}, 27)$ at vertex 1. It represents the batch $b = \{1\}$ comprising only order 1 with capacity consumption three, a collected dual price of 33, and reduced cost of 27. The maximum dual prices that can be collected when extending E to vertex v is $B_1(3) = 34 > 27$, thus label E is kept. Extending E along arc $a_2^0 = (0, 0, \varnothing)$ results in label $E' = (2, 3, 33, \{1\}, 27)$ representing the same batch $b = \{1\}$. Because the corresponding completion bound at vertex 2 is $B_2(3) = 8 \leq 27$, label E' is immediately discarded. In the example, the batch comprising orders 2 and 4 is the optimal solution of the SPPRC and the only batch with negative reduced cost.

2.3.1.4 Acceleration Strategies

Heuristic Pricing In CG, it is not necessary to identify a batch with minimal reduced cost in every iteration. Instead, any negative reduced cost batch is sufficient to continue the CG process and pricing heuristics can be applied to quickly identify such batches. The exact solution algorithm for the pricing problem only has to be invoked if the heuristic pricers fail to identify additional batches.

In our BPC approach, we use a straightforward but highly effective pricing heuristic based on the exact labeling algorithm for the pricing problem. Recall that any feasible 0-v-path P_E in G represented by label E defines a feasible batch $b = O(E)$ with reduced cost $\tilde{c}(E)$. Such a batch could immediately be returned to the RMP without the need to complete the corresponding label to a 0-n-path. Our pricing heuristic proceeds as follows. For each vertex, we count the number of created labels with negative reduced costs (recall that labels are created and extended vertex-by-vertex). As soon as this number reaches the threshold $\mathcal{K} = 0.35n$ for any vertex, the labeling algorithm terminates prematurely and all batches with negative reduced costs are added to the RMP.

Storing Travel Distances The evaluation of the distance function c_b is the computationally most expensive part of the REFs (2.2) and the labeling algorithm. During the course of the BPC, it can be expected that for many batches $b \in \Omega$ function c_b needs to be evaluated multiple times. We, therefore, analyzed the use of a hash table to allow a fast retrieval in amortized constant time of those values c_b that have already been computed. This requires storing in the hash table the key-value pairs (b, c_b) whenever function c_b is evaluated for the first time for batch b.

Overall, using the hash table was not beneficial. We suspect that beside the effort to calculate the hash values and the general overhead for maintaining the hash table, a less effective usage of the CPU cache constitutes a main reason for this behavior. Note that the hash-table implementation performed relatively better for the routing strategies with computationally more costly distance functions (in particular largest gap and optimal) compared to those with less costly distance functions and was almost comparable with the variant without hash table. For more complex warehouse layouts and routing strategies, using a hash table implementation may become beneficial.

2.3.2 Cutting

In our BPC algorithm, two families of valid inequalities are implemented: CCs and SRCs. Both families are non-robust, i.e., they change the structure of the pricing problem.

2.3.2.1 Capacity Cuts

We consider a variant of CCs defined over the variables of Formulation (2.1) introduced by Baldacci *et al.* (2008) for the capacitated vehicle routing problem. Let $S \subseteq O$ be any subset of customer orders and denote by $\kappa(S)$ the minimum number of batches needed to pick all orders in S. The associated CC is

$$\sum_{b \in \Omega_S} \lambda_b \geq \kappa(S), \tag{2.6}$$

where $\Omega_S := \{b \in \Omega : b \cap S \neq \varnothing\}$ is the subset of batches comprising at least one order from subset S.

Impact on Pricing Problem The addition of CCs to the RMP requires the following adjustments to the pricing problem. Let the active CCs be given by the sets $S \in \mathcal{S}$ and denote by $\rho_S > 0$ the corresponding strictly positive dual prices. The reduced cost of a batch b is then given by $\tilde{c}_b = c_b - \sum_{o \in b} \pi_o - \sum_{S \in \mathcal{S}: b \in \Omega_S} \rho_S$, i.e., the dual price ρ_S of S has to be subtracted if b comprises at least one of the orders of S.

To account for the changes to the pricing problem in our solution approach, we adapt graph G of the SPPRC representation and the labeling procedure as follows. Each arc $a_v^k \in A$ is associated with an additional component $cc_v^k(S)$ for each active CC $S \in \mathcal{S}$, with $cc_v^k(S) = 1$ if $k = 1 \ \wedge \ v \in S$, and $cc_v^k(S) = 0$ otherwise. Accordingly, in the labeling algorithm a label E comprises additional resources $cc_S(E)$, one for each $S \in \mathcal{S}$, counting the number of orders $o \in S$ included on path P_E. The REFs for the new resources and the adapted REF for the accumulated dual price when extending label E along arc $a_v^k \in A$ to create the new label E' are

$$cc_S(E') = cc_S(E) + cc_v^k(S) \quad \forall S \in \mathcal{S} \tag{2.7a}$$

$$\pi(E') = \pi(E) + \pi_v^k + \sum_{S \in \mathcal{S}: cc_S(E') \geq 1 \wedge cc_S(E) = 0} \rho_S \tag{2.7b}$$

Due to the linear nature of G, the additional resource corresponding to a CC $S \in \mathcal{S}$ can be disregarded for all labels E with $S \cap \{v(E), ..., n\} = \varnothing$.

Impact on Completion Bounds If there are active CCs in the RMP, $LB(E)$ as defined in (2.3) is no longer valid as it does not account for the strictly positive dual prices ρ_S, $S \in \mathcal{S}$. Consider a label E and a completion $r \in R(E)$. The reduced cost of the corresponding label E^r is

$$\tilde{c}(E^r) = c_{O(E^r)} - \pi(E^r) - \sum_{S \in \mathcal{S}: O(E^r) \cap S \neq \varnothing} \rho_S. \tag{2.8}$$

For monotone routing strategies it holds that

$$\tilde{c}(E^r) \geq c_{O(E)} - \pi(E^r) - \sum_{S \in \mathcal{S}: O(E^r) \cap S \neq \varnothing} \rho_S \tag{2.9a}$$

$$= \tilde{c}(E) - \sum_{o \in O(r)} \pi_o - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing \wedge O(E) \cap S = \varnothing} \rho_S. \tag{2.9b}$$

With respect to the dual prices of the CCs, expression (2.9b) provides the strongest lower bound on the reduced cost $\tilde{c}(E^r)$ exactly incorporating their impact on $\tilde{c}(E^r)$. Because the last term directly depends on $O(E)$, any completion bound based on (2.9b) requires an individual calculation for each label E.

To obtain completion bounds that are more practicable, consider the following weaker lower bound on the reduced cost $\tilde{c}(E^r)$ of E^r

$$\tilde{c}(E^r) \geq \tilde{c}(E) - \sum_{o \in O(r)} \pi_o - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S. \tag{2.10}$$

A corresponding lower bound $LB^1(E)$ on the reduced cost of any capacity-feasible 0-n-path containing path P_E is

$$LB^1(E) := \tilde{c}(E) - \max_{r \in R(E)} \Big(\sum_{o \in O(r)} \pi_o + \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S \Big) \tag{2.11}$$

with associated completion bounds

$$B^1_v(q) := \max_{r \in R(E)} \Big(\sum_{o \in O(r)} \pi_o + \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S \Big) \tag{2.12}$$

that depend only on the vertex $v = v(E)$ and the capacity consumption $q = q(E)$. Similar to $B_v(q)$, the values $B^1_v(q)$ are equivalent to the optimal solution values of extended binary KPs accounting for the additional aspects from the CCs, i.e., the dual prices ρ_S have to be included whenever at least one of the orders of S is packed.

The solution of these KP variants can again be realized by solving a single extended binary KP with a backward labeling algorithm on the modified graph G. To this end, the components of labels E_{kp} of Section 2.3.1.3 and the associated REFs are modified as in the main labeling described above. Additionally, replacing the dominance relation (2.4b) of the profit resource with

$$\pi(E_{kp}) - \sum_{S \in \mathcal{S}: cc_S(E_{kp}) \geq 1 \wedge cc_S(E'_{kp}) = 0} \rho_S \geq \pi(E'_{kp}) \tag{2.13}$$

avoids a point-wise comparison of the new resources $cc_S(E_{kp})$ and $cc_S(E'_{kp})$ by considering all $S \in \mathcal{S}$ for which a common extension of E_{kp} and E'_{kp} may result in

increasing the profit of E'_{kp} without increasing that of E_{kp}. Again, the additional complexity from the CCs can be reduced by disregarding the resources $cc_S(E_{kp})$ for all labels E_{kp} with $S \cap \{0, \ldots, v(E_{kp})\} = \varnothing$ exploiting the linear structure of G.

The completion bounds $B^1_v(q)$ can be retrieved from the undominated labels $\mathcal{E}_v$ as in the case without CCs and the main labeling algorithm for the pricing problem can proceed as described at the end of Section 2.3.1.3 using the modified completion bounds $B^1_v(q)$ instead of $B_v(q)$.

Pretests indicated that the lower bounds $LB^1(E)$ can be rather weak because they include the dual prices cc_S twice if both $O(E) \cap S \neq \varnothing$ and $O(r) \cap S \neq \varnothing$. A stronger lower bound $LB^2(E)$ can be obtained using the following bound on the reduced cost $\tilde{c}(E^r)$ of E^r from equation (2.8). Let $o \in O(E)$. Then,

$$\tilde{c}(E^r) \geq \tilde{c}(E) - \sum_{i \in O(r)} \pi_i - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing \wedge o \notin S} \rho_S \tag{2.14a}$$

$$= \tilde{c}(E) - \Big(\sum_{i \in O(r)} \pi_i + \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing \wedge o \in S} \rho_S \Big). \tag{2.14b}$$

The validity of (2.14) follows directly from (2.9b) and the strict positivity of ρ_S. Because (2.14) holds for all $o \in O(E)$, we have

$$LB^2(E) := \tilde{c}(E) - \max_{o \in O(E)} \max_{r \in R(E)} \Big(\sum_{i \in O(r)} \pi_i + \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing \wedge o \in S} \rho_S \Big). \tag{2.15}$$

$LB^2(E)$ corrects for some of the doubly-included duals ρ_S of $LB^1(E)$.

The corresponding completion bounds

$$B^2_{v,o}(q) := \max_{r \in R(E)} \Big(\sum_{i \in O(r)} \pi_i + \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing} \rho_S - \sum_{S \in \mathcal{S}: O(r) \cap S \neq \varnothing \wedge o \in S} \rho_S \Big) \tag{2.16}$$

to be used in the labeling are now defined for each order o. They can be computed using the information from the solution by backward labeling of the same extended binary KP used for determining $B^1_v(q)$ described above. In fact, we have

$$B^2_{v,o}(q) := \max_{E_{kp} \in \mathcal{E}_v: q(E_{kp}) \leq Q - q} \Big(\pi(E_{kp}) - \sum_{S \in \mathcal{S}: cc_S(E_{kp}) \geq 1 \wedge o \in S} \rho_S \Big). \tag{2.17}$$

Notice that the modified dominance relation (2.13) guarantees that no label $E'_{kp} \notin \mathcal{E}_v$ with $q(E'_{kp}) \leq Q - q$ can provide a larger profit $\pi(E'_{kp}) - \sum_{S \in \mathcal{S}: cc_S(E'_{kp}) \geq 1 \wedge o \in S} \rho_S$ than all labels $E_{kp} \in \mathcal{E}_v$, even if $0 = cc_S(E'_{kp}) < cc_S(E_{kp})$ holds for some $S \in \mathcal{S}$ with $o \in S$ and a smaller amount is deducted from its profit when adjusting for the dual prices of those CCs. Using functions $B^2_{v,o}(q)$ we have

$$LB^2(E) = \tilde{c}(E) - \max_{o \in O(E)} B^2_{v(E),o}(q(E)). \tag{2.18}$$

Separation Procedures In order to identify violated inequalities (2.6), we use three different separation procedures: a greedy construction heuristic, a connected component-based heuristic, and a MIP-based approach.

The exact determination of value $\kappa(S)$ for a subset of orders $S \subseteq O$ requires the solution of a bin packing problem with items $o \in S$, weights q_o and capacity Q. In our separation procedures, we use lower bounds of $\kappa(S)$ that can be computed efficiently. The greedy heuristic and the connected component-based heuristic consider the relaxation $\kappa^3(S) = \max\{\kappa^1(S), \kappa^2(S)\}$ with $\kappa^1(S) := \left\lceil \frac{1}{Q} \sum_{o \in S} q_o \right\rceil$ and $\kappa^2(S) := \left| \left\{ o \in S : q_o > \frac{Q}{2} \right\} \right| + \left\lceil \frac{1}{Q} \sum_{o \in S : q_o = \frac{Q}{2}} q_o \right\rceil$. The MIP-based approach only uses relaxation $\kappa^1(S)$.

Denote by $(\bar{\lambda}_b)_{b \in \Omega'}$ the current fractional solution of the RMP and let $\bar{\lambda}_S := \sum_{b \in \Omega_S} \bar{\lambda}_b$ for any subset $S \subseteq O$ of orders. Moreover, let $\bar{\Omega} := \{b \in \Omega : \bar{\lambda}_b > 0\}$ be the set of batches with positive value in the current solution, let $\bar{\Omega}_S := \{b \in \bar{\Omega} : b \cap S \neq \varnothing\}$ be the set of batches with positive value comprising at least one order from S, and let $\bar{N}_S := \bigcup_{b \in \bar{\Omega}_S} b \setminus S$ be the set of neighboring orders of subset S in the current solution, i.e., orders $o \in O \setminus S$ comprised in any batch of set $\bar{\Omega}_S$. A connected component of $(\bar{\lambda}_b)_{b \in \Omega'}$ is a subset S of orders with $\bar{N}_S = \varnothing$.

The first separation procedure is a randomized greedy construction heuristic. It is initialized with a set S comprising a single order only and each order is tried as a starting point several times. Iteratively, the heuristic adds to the current set S a single order $o \in \bar{N}_S$ maximizing the expression

$$g_o(S) := \frac{\kappa^3(S \cup \{o\}) - \kappa^3(S)}{\bar{\lambda}_{S \cup \{o\}} - \bar{\lambda}_S}. \tag{2.19}$$

To randomize the heuristic, $g_o(S)$ is multiplied by a random number uniformly drawn from the interval $[0.7, 1.3]$. The heuristic stops either if a violated inequality (2.6) is detected or if $\bar{\lambda}_S > \kappa^3(S) + 1$ holds for the current set S. Overall, the greedy construction heuristic tends to find violated CCs with sets S of small cardinality.

The second separation procedure is inspired from a separation heuristic used by Archetti *et al.* (2011) for the split delivery vehicle routing problem. The starting point of the heuristic is the connected components of the RMP solution. For each connected component S, the heuristic proceeds as follows. The batches $b \in \bar{\Omega}_S$ are sorted by non-increasing value $\bar{\lambda}_b - \kappa^3(b)$. In each iteration, the heuristic selects the next batch b in the sorting and removes from S all orders $o \in b$. This choice of orders to remove from S provides subsets with a high potential of violating inequality (2.6), because it decreases the left-hand side by a large value and the right-hand side by a small value. Whenever the current set S violates inequality (2.6) the corresponding cut is generated. The heuristic stops the removal process, when the total number of orders removed from the initial connected component exceeds 25

or when the current set S is empty. The procedure is then restarted a second time for the same connected component, skipping the very first batch b in the sorting. The connected component heuristic seems to predominantly identify violated CCs with large sets S.

A third separation procedure uses a MIP formulation to exactly separate CCs using relaxation $\kappa^1(S)$. The MIP (2.20) adapts to the OBP the formulation of Martinelli *et al.* (2013) for exactly separating CCs for the capacitated arc routing problem. It identifies a subset S and the corresponding right-hand side $\kappa^1(S)$ maximizing the violation $\kappa^1(S) - \sum_{b \in \bar{\Omega}_S} \lambda_b$ of inequality (2.6). The MIP uses the following variables. Binary variables y_o indicate if order $o \in O$ is in the searched subset S or not. Binary variables x_b indicate if batch $b \in \bar{\Omega}$ comprises any order of the searched subset S or not. Integer variable K represents the value $\kappa^1(S)$ while the continuous slack variable $\gamma \in [0, 1)$ describes the fractional difference of applying the ceiling function in the determination of $\kappa^1(S)$. Our separation MIP can then be stated as follows:

$$\max\ K - \sum_{b \in \bar{\Omega}} x_b \bar{\lambda}_b \tag{2.20a}$$

$$\text{s.t. } K = \gamma + \sum_{o \in O} \frac{q_o y_o}{Q} \tag{2.20b}$$

$$x_b \leq \sum_{o \in b} y_o \quad \forall b \in \bar{\Omega} \tag{2.20c}$$

$$|b| x_b \geq \sum_{o \in b} y_o \quad \forall b \in \bar{\Omega} \tag{2.20d}$$

$$x_b \in \{0, 1\} \quad \forall b \in \bar{\Omega} \tag{2.20e}$$

$$y_o \in \{0, 1\} \quad \forall o \in O \tag{2.20f}$$

$$K \in \mathbb{Z}_0^+ \tag{2.20g}$$

$$\gamma \in [0, 1) \tag{2.20h}$$

The Objective (2.20a) maximizes the violation of the CC. Equation (2.20b) ensures the correct value of K. Linking constraints (2.20c) force the indicator variables x_b to zero if none of the orders $o \in b$ is selected, while linking constraints (2.20d) force the indicator variables x_b to one if at least one of the orders $o \in b$ is selected. The variable domains are defined in (2.20e)–(2.20h).

2.3.2.2 Subset-Row Cuts

SRCs were first introduced by Jepsen *et al.* (2008) for the *vehicle routing problem with time windows*. They are Chvátal-Gomory rank-1 cuts based on subsets of the set-partitioning constraints (2.1b). As proposed by Jepsen *et al.* (2008) and followed in the majority of works, we restrict ourselves to subsets of cardinality

three. Let $U \subset O$ be a set of three orders. The associated SRC is given by

$$\sum_{b \in \Omega} \left\lfloor \frac{\sum_{o \in U} a_{ob}}{2} \right\rfloor \lambda_b \leq 1. \tag{2.21}$$

Violated SRCs can be separated by straightforward enumeration.

Impact on Pricing Problem The addition of SRCs to the RMP changes the pricing problem as follows. Let the active SRCs be given by the sets $U \in \mathcal{U}$. The corresponding strictly negative dual prices are $\sigma_U < 0$. The reduced cost of a batch b is now $\tilde{c}_b = c_b - \sum_{o \in b} \pi_o - \sum_{U \in \mathcal{U}: |U \cap b| \geq 2} \sigma_U$, i.e., the dual price σ_U of U has to be subtracted if b comprises at least two of the orders of U.

The altered pricing problem requires adjustments to graph G of the SPPRC representation and the labeling algorithm for its solution similar to the CC case. With each arc $a_v^k \in A$ is associated an additional component $sr_v^k(U)$ for each active SRC $U \in \mathcal{U}$, with $sr_v^k(U) = 1$ if $k = 1 \ \wedge \ v \in U$, and $sr_v^k(U) = 0$ otherwise. Likewise, for each $U \in \mathcal{U}$, an additional resource $sr_U(E)$ is added to the labels E of the labeling algorithm. It counts the number of orders $o \in U$ included on path P_E. The propagation of label E along arc $a_v^k \in A$ to obtain a new label E' uses the following new and modified REFs:

$$sr_U(E') = sr_U(E) + sr_v^k(U) \quad \forall U \in \mathcal{U} \ , \tag{2.22a}$$

$$\pi(E') = \pi(E) + \pi_v^k + \sum_{U \in \mathcal{U}: sr_U(E') \geq 2 \wedge sr_U(E) \leq 1} \sigma_U. \tag{2.22b}$$

As in the CC case, the linear structure of G can be exploited to disregard all SRCs $U \in \mathcal{U}$ for labels E with $U \cap \{v(E), ..., n\} = \varnothing$.

Impact on Completion Bounds The completion bounds $B_v(q)$, $B_v^1(q)$, and $B_v^2(q)$ as defined in Section 2.3.1.3 and above remain valid also in the presence of SRCs, because the dual prices σ_U are strictly negative for all $U \in \mathcal{U}$. Therefore, no modifications to the bounding procedure are needed.

Incorporating the dual prices from the SRCs generally results in stronger completion bounds. Similar to the CC case, these refined bounds require solving extended binary KPs accounting for the additional aspects from the SRCs. Pretests have shown that the additional effort for computing the refined bounds incorporating the SRCs outweighs their benefit of allowing more labels to be discarded in the main labeling. Thus, in our BPC, we ignore the SRCs in the completion bounds.

2.3.3 Branching

We use a two-stage hierarchical branching scheme. On the first level, we branch on the number of pickers $\sum_{b \in \bar{\Omega}} \bar{\lambda}_b$, if fractional, and create the two branches $\sum_{b \in \Omega} \lambda_b \leq$

$\left\lfloor \sum_{b\in\bar{\Omega}} \bar{\lambda}_b \right\rfloor$ and $\sum_{b\in\Omega} \lambda_b \geq \left\lceil \sum_{b\in\bar{\Omega}} \bar{\lambda}_b \right\rceil$. Both decisions are implemented by adding the respective linear constraint to the RMP. No structural changes to the pricing problem are involved.

On the second level, we apply the well-known Ryan-and-Foster branching rule. Let $f_{o_1o_2} := \sum_{b\in\bar{\Omega}} a_{o_1b}a_{o_2b}\bar{\lambda}_b$ be the information if the orders $o_1, o_2 \in O$ are assigned to the same batch or not. We branch on pairs (o_1, o_2) of orders for which $f_{o_1o_2}$ is fractional. Two branches are created, the *separate* branch, which ensures $f_{o_1o_2} = 0$ by forcing variables λ_b with $a_{o_1b} = a_{o_2b} = 1$ to zero, and the *together* branch, which ensures $f_{o_1o_2} = 1$ by forcing variables λ_b with $a_{o_1b} + a_{o_2b} = 1$ to zero. Both types of decisions can be straightforwardly implemented in the RMP by fixing the corresponding variables to zero but impose structural changes to the pricing problem.

Impact on Pricing Problem In the pricing problem, the generation of variables that are incompatible with the Ryan-and-Foster branching decisions has to be prevented. This can be realized by modifying the underlying graph G of the SPPRC representation of the pricing problem. The basic idea is to group together the orders that are affected by mutual branching decisions, representing them by a single vertex in G and to decide on the inclusion of the orders of a group simultaneously. On the modified graph, the same labeling algorithm presented in the previous sections can be applied to solve the pricing problem in the presence of branching decisions.

For ease of notation, we identify a Ryan-and-Foster branching decision by the set $I = \{o_1, o_2\}$ comprising the two involved orders. The type of decision, separate or together, can be ignored for the moment. Let $\mathcal{I} = \{I_1, \ldots, I_p\}$ be the set of active branching decisions at a given B&B node.

Let $\mathcal{I}^1, \ldots, \mathcal{I}^q$ be a partition of $\mathcal{I}$ into subsets, i.e., groups of branching decisions, such that the different subsets are non-overlapping with respect to the involved orders, i.e., $(\bigcup_{I\in\mathcal{I}^i} I) \cap (\bigcup_{I\in\mathcal{I}^j} I) = \varnothing$ holds for all pairs $i, j \in \{1, ..., q\}, i \neq j$, while each individual subset consists of overlapping branching decisions. The branching decisions of a set $\mathcal{I}^i := \{I^i_1, \ldots, I^i_r\}$ are overlapping if they can be ordered such that $(I^i_1 \cup \cdots \cup I^i_{j-1}) \cap I^1_j \neq \varnothing$ for all $j \in \{1, ..., r\}$. For each $\mathcal{I}^i, i \in \{1, \ldots, q\}$, denote by $O(\mathcal{I}^i) := \bigcup_{I\in\mathcal{I}^i} I$ the set of involved orders. Furthermore, let $O_0(\mathcal{I}^i) = \varnothing, O_1(\mathcal{I}^i), \ldots, O_s(\mathcal{I}^i)$ be all feasible combinations (=subsets) of the orders in $O(\mathcal{I}^i)$ such that the branching decisions and the picking capacity are respected.

In the modified graph G, each set $O(\mathcal{I}^i)$ associated with a group of branching decisions $\mathcal{I}^i$ is represented by a single vertex, say vertex v. For each feasible combination of orders $O_k(\mathcal{I}^i), k \in \{0, \ldots, s\}$, an arc a^k_v from vertex $v-1$ to vertex v is created. The components associated with arc a^k_v are determined as $q^k_v = \sum_{o\in O_k(\mathcal{I}^i)} q_o$, $\pi^k_v = \sum_{o\in O_k(\mathcal{I}^i)} \pi_o$, $O^k_v = O_k(\mathcal{I}^i)$, $sr^k_v(U) = |O_k(\mathcal{I}^i) \cap U|$ for

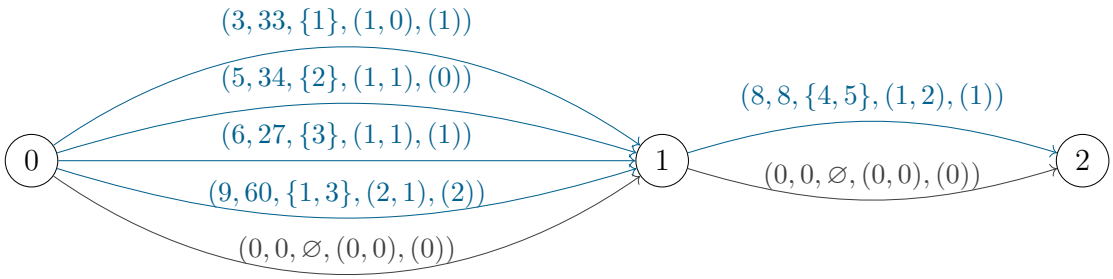


Figure 2.4: SPPRC representation of the pricing problem for the example instance of Figure 2.3 and Ryan-and-Foster branching decisions $\{1,2\}^s$, $\{2,3\}^s$, and $\{4,5\}^t$

all $U \in \mathcal{U}$, and $cc_v^k(S) = |O_k(\mathcal{I}^i) \cap S|$ for all $S \in \mathcal{S}$. Each of the remaining orders $o \in O \setminus \bigcup_{I \in \mathcal{I}} I$ not involved in any branching decision is associated with a single vertex and two ingoing arcs in G as described in Section 2.3.1.1.

An example of graph G after branching is illustrated in Figure 2.4. It continues the example pricing instance of Figure 2.3 except for the capacity which is assumed to be not binding. Furthermore, we assume that $\mathcal{I}$ comprises the three active branching decisions $\{1,2\}^s$, $\{2,3\}^s$, and $\{4,5\}^t$, where the superscript s (t) refers to a separate (together) constraint, resulting in two groups $\mathcal{I}^1$ and $\mathcal{I}^2$ of branching decisions with involved orders $O(\mathcal{I}^1) = \{1,2,3\}$ and $O(\mathcal{I}^2) = \{4,5\}$ and corresponding vertices 1 and 2 in G, respectively. Additionally, there are two active CCs $S_1 = \{1,2,3,5\}$ and $S_2 = \{2,3,4,5\}$ and one active SRC $U = \{1,3,4\}$. Figure 2.4 now depicts the modified graph G showing all arcs a_v^k with their components $(q_v^k, \pi_v^k, O_v^k, (cc_v^k(S))_{S \in \mathcal{S}}, (sr_v^k(U))_{U \in \mathcal{U}})$. Consider vertex 1 associated with orders $O(\mathcal{I}^1) = \{1,2,3\}$. The five ingoing arcs represent all subsets of orders from the set $O(\mathcal{I}^1)$ that respect the mutual branching decisions $\{1,2\}^s$, $\{2,3\}^s$, i.e., that can be feasibly included in a batch. Now consider arc $(9, 60, \{1,3\}, (2,1), (2))$ corresponding with the inclusion of the orders 1 and 3. It is associated with a capacity consumption of $3+6=9$, a dual price of $33+27=60$, the information that orders 1 and 3 are both comprised in CC S_1, one of them (order 3) is comprised in CC S_2, and both are comprised in SRC U.

To contain the size of the B&B tree, we apply strong branching at the second stage of the branching scheme as detailed in Appendix 2.D.

The node selection strategy is best-bound first, because the primary goal of our BPC is to improve the dual bound.

2.3.4 BPC-based Heuristics

In Section 2.4, it becomes apparent that the proposed CG method is very effective, being able to quickly solve the LP relaxation of Formulation (2.1). Closing the gap and finally proving optimality is the hard part where the BPC algorithm spends almost all the computation time for most instances. We, therefore, propose two straightforward BPC-based heuristics to provide high-quality solutions in limited computation time.

The first heuristic (denoted *SC*) consists in solving a restricted version of Formulation (2.1) comprising only the columns generated up to the root node of the BPC with a general-purpose MIP solver. Any cuts generated while solving the root node are removed before invoking the solver. For this heuristic, no branching at all needs to be implemented.

The second heuristic (denoted *BPC-DF*) changes the node selection strategy of the BPC algorithm to a combination of best-bound-first and depth-first search in order to quickly identify high-quality feasible solutions. The search strategy first selects a node according to the best-bound-first rule. It then explores the subtree rooted at this node in a depth-first fashion until the current node either provides an integer solution or can be pruned. The next node to evaluate is again selected according to the best-bound-first rule. Setting a hard time limit guarantees a quick termination of the approach.

2.4 Computational Results

Our BPC algorithm and the BPC-based heuristics were implemented in `C++` and compiled into 64-bit single-thread code with MS Visual Studio 2019. CPLEX 20.10 with default parameters (except for the time limit and allowing only a single thread) is used to reoptimize the RMPs and as MIP-solver for the MIP-based separation of CCs and the SC heuristic. The computations were carried out on the HPC cluster Elwetritsch of the University of Kaiserslautern-Landau consisting of several Intel Xeon Gold 6126 processors running at 2.60 GHz. Notice that the performance of a single thread of the cluster is comparable to that of a standard desktop processor.

An overview of additional design choices and implementation details for our BPC algorithm as well as the specific values used for different parameters of the algorithm is given in Appendix 2.D. These values were obtained in pretests on a small subset of the instances used in our main computational study. Notice that the same computational setup was used for all routing strategies and instances.

More detailed results can be found in Tables 2.11–2.62 of Appendix 2.F. Furthermore, instance-by-instance results of our main BPC and the two BPC-based heuris-

tics together with the BKS are provided at `https://logistik.wiwi.uni-kl.de/obp-bpc-detailedresults`.

2.4.1 Benchmark Instances

Benchmark instances for the standard OBP have been proposed by Henn and Wäscher (2012) (`H&W`), Muter and Öncan (2015) (`M&Ö`), Žulj *et al.* (2018) (`ZKS`), and Bahçeci and Öncan (2022) (`B&Ö`). The main focus of our computational study are the `H&W` and `M&Ö` instances. We also report results for the large-scale `ZKS` instances. To further test the limits of the proposed solution approaches and to allow the comparison of new exact and heuristic algorithms with our approaches in solving larger-sized instances, we additionally consider the `M&Ö` instances with enlarged capacities (`M&Ö-ext`) and have created two new sets of larger instances with uniform (`W&G-u`) and general weights (`W&G-g`) following the work of Hwang and Kim (2005). Results for the latter three sets are reported mainly in Appendix 2.F. The very small-scale `B&Ö` instances, of which the largest ones are smaller than the smallest `M&Ö` instances, are not included in our study. All considered instances are available at `https://logistik.wiwi.uni-kl.de/obp-instances`. They are described in more detail in Appendix 2.E.

2.4.2 Evaluation of Algorithmic Components

We first investigate the impact of different components of our BPC algorithm on its performance. To this end, we consider variants of the BPC with and without strong branching (*StrBr* and *noStrBr*), with and without SRCs (*SRC* and *noSRC*), and with and without CCs (*CC* and *noCC*). When using CCs, we further distinguish variants according to the separation procedures that are used: only the connected component heuristic (*CC|cp*), only the greedy heuristic (*CC|gr*), only the MIP-based approach (*CC|MIP*), or all of them (*CC|all*).

The results for benchmark sets `M&Ö` and `H&W` and all six routing strategies are summarized in Table 2.2 and Figure 2.5. Table 2.2 reports the percentage number of instances solved to proven optimality within the time limit (*%Opt*) and the average solution time in seconds (t[*s*]) where unsolved instances are included with the time limit of 3,600 seconds. Rows *All (eq.)* report average numbers using an equal weighting for the two benchmark sets. Figure 2.5 depicts the performance profiles of the different algorithm variants. The performance profile of an algorithm variant specifies the percentage of instances solved by this variant within τ times the time taken by the fastest variant in an instance-by-instance comparison.

The largest benefit is clearly achieved by the integration of CCs. All variants without CCs perform substantially worse. This is especially true for the `M&Ö` instances where with CCs the computation times are more than halved and the num-

	No capacity cuts				With capacity cuts									
	noSRC		SRC		noSRC,CC|all		SRC,CC|cp		SRC,CC|gr		SRC,CC|MIP		SRC,CC|all	
Class	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]
Panel A: No strong branching														
M&Ö	45.4	2,099	48.1	2,000	69.9	1,266	75.4	1,054	47.8	2,007	72.8	1,138	75.9	1,039
H&W	74.5	1,018	76.8	937	87.0	596	88.8	513	76.7	939	88.0	551	89.3	503
All (eq.)	59.9	1,558	62.5	1,468	78.4	931	82.1	784	62.3	1,473	80.4	844	82.6	771
Panel B: With strong branching														
M&Ö	50.2	1,916	53.0	1,857	80.8	920	85.8	796	52.7	1,860	80.4	994	84.2	852
H&W	78.3	888	79.9	842	91.2	457	92.2	412	79.9	839	91.0	471	92.4	414
All (eq.)	64.2	1,402	66.4	1,349	86.0	689	89.0	604	66.3	1,350	85.7	733	88.3	633

Table 2.2: Summary results for different variants of our BPC algorithm

ber of solved instances is increased by around 60%. The positive impact of strong branching and SRCs is also evident, but much smaller than for the CCs. Again, the effect is more pronounced for the M&Ö instances than for the H&W instances. Regarding the separation procedures for the CCs, using only the connected component heuristic is overall slightly superior to using all separation routines (the picture is reversed for the H&W instances) which is in turn slightly superior to only using the MIP-based separation. The greedy heuristic alone is not beneficial, performing similar to the variants without CCs. Summing up, the best performing variant, which is also used in the following sections, utilizes strong branching, both types of cuts, and only the connected component heuristic. Note that the variant using all separation procedures for the CCs was able to provide substantially more optima for the H&W instances with traversal strategy as well as for the ZKS instances.

Notice that without the pricing heuristic, we run into memory issues already for moderately sized instances, e.g., around $Q = 36/n = 50$-60 and $Q = 48/n = 50$ for the M&Ö benchmark. This is why we did not include variants without the pricing heuristic in our comparison. The same is true when using the weaker completion bounds $B^1_v(q)$ in the solution of the pricing problem with CCs.

2.4.3 Computational Analysis of BPC Algorithm

We first compare our BPC with the state-of-the-art exact approach to the standard OBP of Muter and Öncan (2015) who only consider routing strategies traversal, return, and midpoint and the M&Ö instances. Table 2.3 summarizes the comparison. It provides the number of instances solved to optimality (Opt) and the average solution times in seconds ($t[s]$). Note that the computation times reported in Muter and Öncan (2015) and, therewith, in Table 2.3 comprise only the computation times for the cut-and-column generation phase including the batch enumeration of their approach but not the time for solving the reduced set-partitioning problem

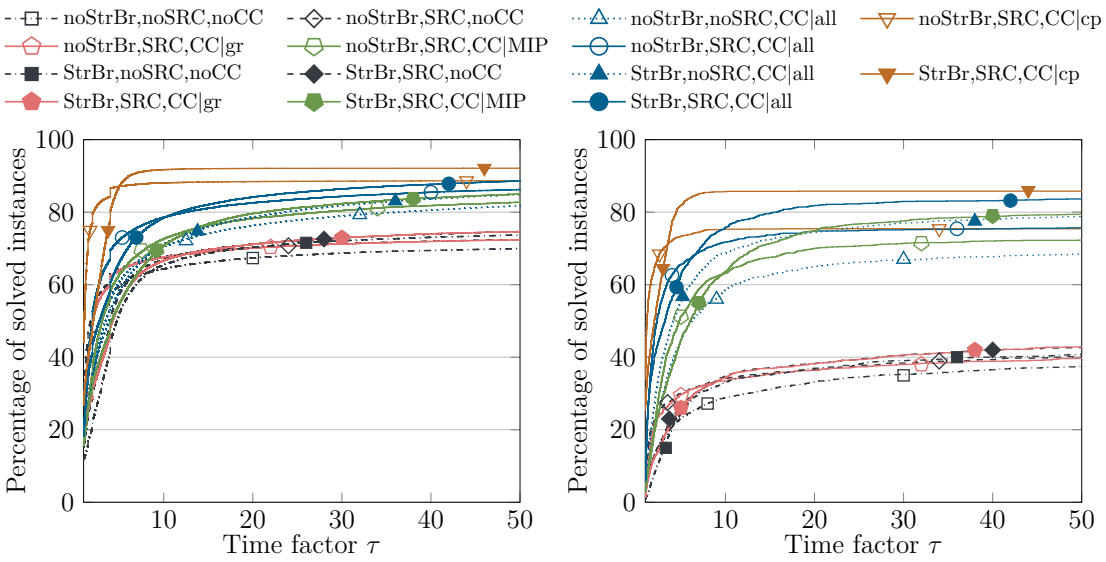


Figure 2.5: Performance profiles of different variants of our BPC algorithm for the H&W instances (left) and the M&Ö instances (right)

with CPLEX. Note further that they allow one hour of computation time for each of the two phases of their algorithm. We still count unsolved instances with 3,600 seconds in the average solution times also for their approach. Note finally that their computer is about 30% slower than ours according to the single-thread performance reported on www.passmark.com.

Table 2.3 indicates that our BPC clearly outperforms the method of Muter and Öncan (2015). While they solve 55, 71, and 75 out of the 270 instances for strategies traversal, return, and midpoint, respectively, our BPC is able to solve more than three times the number of instances, namely 208, 237, and 244. A meaningful comparison of computation times is only possible for a small number of instance groups. For the majority of groups, the averages for the method of Muter and Öncan (2015) are dominated by the time limit (notice that no results for individual instances are available).

Table 2.4 summarizes the results of our BPC for the benchmarks M&Ö and H&W and all considered routing strategies aggregated by capacity Q. As expected, instances with larger capacities (and more orders) are harder to solve. Regarding the routing strategies, traversal seems to be the most difficult for our BPC. Compared to the other strategies, substantially less optima are provided and the average computation times are much higher. Routing strategies return, largest gap, combined, and optimal appear to be of very comparable difficulty, while midpoint seems to be slightly easier. For all routing strategies except traversal, the M&Ö instances

			Traversal				Return				Midpoint			
			M&Ö†		Our method‡		M&Ö†		Our method‡		M&Ö†		Our method‡	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
24	20	10	10	7.0	10	0.2	10	7.0	10	0.1	10	5.0	10	0.0
	30	10	7	1,095.4	10	0.7	9	377.1	10	1.1	9	373.5	10	0.4
	40	10	8	807.2	10	8.6	7	1,105.2	10	1.9	10	35.0	10	1.6
	50	10	4	2,205.6	10	14.8	6	1,491.6	10	13.0	6	1,472.4	10	3.0
	60	10	6	1,516.8	10	11.5	6	1,519.2	10	3.0	7	1,210.2	10	5.2
	70	10	2	2,901.6	10	37.9	5	1,938.5	10	73.8	3	2,564.7	10	17.3
	80	10	1	3,315.1	10	392.4	2	2,944.2	10	128.3	4	2,325.2	10	39.2
	90	10	2	2,928.0	8	1,097.6	3	2,621.1	10	570.1	3	2,625.3	10	269.3
	100	10	2	2,959.4	10	493.0	2	2,936.0	10	274.8	4	2,362.0	9	582.6
Subtot.		90	42	1,970.7	88	228.5	50	1,660.0	90	118.5	56	1,441.5	89	102.1
36	20	10	6	1,450.8	10	2.1	9	396.0	10	0.6	8	744.8	10	0.3
	30	10	4	2,170.4	10	25.4	7	1,154.2	10	5.6	6	1,478.4	10	2.5
	40	10	0	3,600.0	10	43.5	1	3,255.5	10	30.8	0	3,600.0	10	7.2
	50	10	0	3,600.0	9	649.2	0	3,600.0	10	59.3	0	3,600.0	10	27.1
	60	10	0	3,600.0	10	805.8	0	3,600.0	10	137.9	0	3,600.0	10	85.1
	70	10	0	3,600.0	8	1,125.5	0	3,600.0	10	791.9	0	3,600.0	10	160.4
	80	10	0	3,600.0	5	2,141.7	0	3,600.0	8	1,829.8	0	3,600.0	10	989.0
	90	10	0	3,600.0	6	2,295.7	0	3,600.0	10	1,098.1	0	3,600.0	10	1,402.3
	100	10	0	3,600.0	4	2,394.8	0	3,600.0	5	2,971.8	0	3,600.0	8	2,107.6
Subtot.		90	10	3,202.4	72	1,053.7	17	2,934.0	83	769.5	14	3,047.0	88	531.3
48	20	10	3	2,535.0	10	3.6	4	2,193.6	10	1.2	5	1,866.5	10	0.5
	30	10	0	3,600.0	10	146.5	0	3,600.0	10	21.8	0	3,600.0	10	46.4
	40	10	0	3,600.0	9	612.7	0	3,600.0	10	32.0	0	3,600.0	10	57.3
	50	10	0	3,600.0	6	2,285.0	0	3,600.0	10	613.0	0	3,600.0	10	187.0
	60	10	0	3,600.0	6	1,677.9	0	3,600.0	10	363.6	0	3,600.0	10	268.6
	70	10	0	3,600.0	3	2,951.5	0	3,600.0	5	2,208.9	0	3,600.0	6	2,286.7
	80	10	0	3,600.0	1	3,316.5	0	3,600.0	5	2,381.6	0	3,600.0	7	2,197.2
	90	10	0	3,600.0	1	3,276.8	0	3,600.0	1	3,387.3	0	3,600.0	3	3,160.0
	100	10	0	3,600.0	2	3,039.8	0	3,600.0	3	3,140.0	0	3,600.0	1	3,387.8
Subtot.		90	3	3,481.7	48	1,923.4	4	3,443.7	64	1,349.9	5	3,407.4	67	1,287.9
Total		270	55	2,884.9	208	1,068.5	71	2,679.2	237	746.0	75	2,632.0	244	640.4

Table 2.3: Comparison of our BPC algorithm with the approach of Muter and Öncan (2015) for routing strategies traversal, return and midpoint on the `M&Ö` instances

†: Windows computer with an Intel Xeon X5460 processor, single thread score (`www.passmark.com`): 1370

‡: Linux computer with an Intel Xeon Gold 6126 processor, single thread score (`www.passmark.com`): 2019

		Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
Panel A: M&Ö instances													
24	90	88	228.5	90	118.5	89	102.1	89	87.8	90	108.2	88	131.3
36	90	72	1,053.7	83	769.5	88	531.3	80	714.2	81	861.7	79	802.0
48	90	48	1,923.4	64	1,349.9	67	1,287.9	66	1,355.1	63	1,458.4	65	1,451.2
Total	270	208	1,068.5	237	746.0	244	640.4	235	719.0	234	809.4	232	794.8
Panel B: H&W instances													
30	1,440	1,420	55.6	1,440	0.1	1,440	0.1	1,440	0.1	1,438	5.2	1,440	1.6
45	1,440	1,352	307.8	1,419	98.0	1,438	20.7	1,434	39.3	1,419	104.1	1,424	78.9
60	1,440	1,020	1,203.4	1,356	380.8	1,384	276.2	1,374	348.2	1,355	428.5	1,346	456.0
75	1,440	720	1,960.2	1,247	820.8	1,262	732.6	1,228	848.8	1,249	798.0	1,204	923.9
Total	5,760	4,512	881.8	5,462	325.0	5,524	257.4	5,476	309.1	5,461	333.9	5,414	365.1

Table 2.4: Summary results of our BPC algorithm for the M&Ö and H&W instances and all routing strategies

are overall more difficult (less instances solved and larger computation times) than the H&W instances. Following the trend for the largest M&Ö and H&W instances, only a limited number of the larger-sized instances from benchmarks M&Ö-ext, W&G-g, and W&G-u can be solved to optimality (see Tables 2.14–2.16 in Appendix 2.F). We can also observe, that the instances with uniform order weights seem slightly easier for our BPC than those with general weights.

A detailed analysis averaged by capacity of our BPC is provided in Table 2.5 for routing strategies traversal and optimal. The latter is representative also for the remaining strategies return, midpoint, largest gap and combined. The additional columns are the average time for solving the LP relaxation in seconds (t^{LP}), the average optimality gap with respect to the BKS of the LP relaxation (Gp), the average optimality gap with respect to the BKS before the first node resulting from a Ryan-and-Foster branching is solved (Gp^{RF}), the average number of B&B nodes solved (Nds), and the average number of CCs (CC) and SRCs (SRC) added.

The most striking observation when comparing the two routing strategies is the difference in the number of solved B&B nodes. On average, this number is twice as large on the M&Ö instances and seven times as large on the H&W instances for traversal than for optimal. This is the main reason, why traversal is more difficult to solve than the other strategies. The solution time for a single node, on the other hand, is much smaller for traversal than for optimal and the other strategies as indicated by the average time for solving the LP relaxation. This can be explained by the fact that the distance function for traversal is computationally the least expensive of all strategies.

Overall, Table 2.5 reveals that the times needed for solving the LP relaxations are very short (note that the maximum for any of the instances is 84 seconds and only six instances require more than 60 seconds) and the average optimality gaps

		Traversal								Optimal							
Q	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
Panel A: M&Ö instances																	
24	90	88	228.5	0.2	0.98	0.31	2,346	21	42	88	131.3	0.3	0.80	0.23	1,070	12	36
36	90	72	1,053.7	1.2	2.12	0.54	2,633	41	91	79	802.0	2.7	1.74	0.40	1,716	38	69
48	90	48	1,923.4	5.3	2.98	0.68	1,750	55	113	65	1,451.2	16.6	2.74	0.66	558	67	99
Tot.	270	208	1,068.5	2.2	2.03	0.51	2,243	39	82	232	794.8	6.5	1.76	0.43	1,114	39	68
Panel B: H&W instances																	
30	1,440	1,420	55.6	0.0	0.30	0.03	533	8	1	1,440	1.6	0.0	0.22	0.01	45	2	1
45	1,440	1,352	307.8	0.1	0.57	0.23	7,103	21	27	1,424	78.9	0.2	0.40	0.16	1,396	10	21
60	1,440	1,020	1,203.4	0.4	0.96	0.34	20,562	21	54	1,346	456.0	0.7	0.77	0.29	3,147	22	40
75	1,440	720	1,960.2	1.0	1.29	0.36	21,534	28	80	1,204	923.9	2.1	1.19	0.38	2,569	37	57
Tot.	5,760	4,512	881.8	0.4	0.78	0.24	12,433	19	41	5,414	365.1	0.7	0.65	0.21	1,789	18	30

Table 2.5: Detailed results of our BPC algorithm for the M&Ö and H&W instances and routing strategies traversal and optimal

are rather small: less than 2% for the LP relaxation, and less than 0.5% after adding cuts. This is also true when considering the individual instance groups for all routing strategies and benchmark sets from the literature as shown by the detailed results in Appendix 2.F. Comparing the two benchmark sets, we see that LP times and gaps are larger for the M&Ö instances than for the H&W instances, which can be an explanation why the former are overall harder to solve.

The main insights taken from Table 2.5 are also valid for the larger-sized instances M&Ö-ext, W&G-g, and W&G-u. Note that with increasing instance sizes, the times for solving the LP relaxation also strongly increases. In fact, for the largest instances of the W&G-g and W&G-u benchmarks, our BPC was consistently not able to solve the LP relaxation within the time limit.

2.4.4 Computational Analysis of BPC-based Heuristics

We now analyze our BPC-based heuristics SC and BPC-DF and compare it to the state-of-the-art heuristic approach of Žulj *et al.* (2018), who proposed a hybrid of an *adaptive large neighborhood search* and a *tabu search* (ALNS/TS) for the routing strategies traversal and largest gap. They report results only for subsets of the H&W benchmark and for the large-scale ZKS instances.

Tables 2.6–2.8 summarize the comparison on these instances. The results for both BPC-based heuristics are obtained using a hard time limit of two minutes indicated by the suffix *-2*. The tables report the average gap with respect to the best-known lower bound (Gp) and the average computation time in seconds ($t[s]$). On all subsets of instances considered by Žulj *et al.* (2018), we are able to drastically improve on their gaps with both types of heuristics. For example, for the H&W instances with CBD and the largest gap strategy they obtain an average gap

H&W CBD/largest gap instances							H&W UDD/largest gap instances						
	SC-2‡		BPC-DF-2‡		ALNS/TS§			SC-2‡		BPC-DF-2‡		ALNS/TS§	
n	Gp	t[s]	Gp	t[s]	Gp	t[s]	n	Gp	t[s]	Gp	t[s]	Gp	t[s]
40	0.10	0.8	0.00	2.2	0.24	11.2	40	0.12	2.7	0.01	8.6	0.20	10.9
60	0.09	5.0	0.02	21.2	0.63	34.6	60	0.11	13.9	0.06	36.1	0.53	32.1
80	0.19	22.3	0.07	48.9	0.85	75.3	80	0.38	36.4	0.20	57.0	0.84	72.0
100	0.35	44.8	0.17	66.4	1.00	141.9	100	0.56	53.6	0.34	69.7	0.92	133.5
Total	0.18	18.2	0.07	34.7	0.68	65.7	Total	0.29	26.7	0.15	42.8	0.62	62.1

Table 2.6: Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the largest gap strategy on a subset of the H&W instances

‡: Linux computer with an Intel Xeon Gold 6126 processor, single thread score (www.passmark.com): 2019

§: Windows computer with an Intel Core i7-3770 processor, single thread score (www.passmark.com): 2071

of 0.68% while SC-2 and BPC-DF-2 achieve gaps of 0.18% and 0.07%, respectively. For largest gap, these reductions even come with much shorter computation times for both SC-2 and BPC-DF-2. For the large-scale ZKS instances, we are again able to reduce the average gap from 2.56%, which represents the instance-wise best out of three runs of the ALNS/TS, to 0.65% and 1.10% for SC-2 and BPC-DF-2, respectively. Surprisingly, our BPC-based heuristics seem to scale much better than the ALNS/TS of Žulj *et al.* (2018), because the latter gaps are obtained with an average computation time of 48.8 seconds (SC-2) and 115.8 seconds (BPC-DF-2) compared to 1,570.4 seconds (ALNS/TS).

Table 2.9 provides a very aggregated summary by benchmark set of the BPC-based heuristics investigating the impact of the allowed computation time on SC and BPC-DF. We report results for both heuristics and with hard time limits of two, three, and five minutes indicated with a corresponding suffix. For both types of heuristics, the average gaps strictly decrease with increasing time limit, i.e., there is a direct trade-off between allowed computation time and solution quality. On all benchmark sets, BPC-DF consumes on average more of the allowed computation time than SC. However, BPC-DF also seems to benefit more from larger time limits in the sense that it is able to generate larger improvements in solution quality than SC with increasing computation time. Overall, BPC-DF performs better than SC for the M&Ö and H&W instances, while for the ZKS instances the picture is reversed.

With the computations carried out to obtain the results of Table 2.9, we are also able to improve on hundreds of BKS. For the H&W benchmark, there are 2,720 instances for which BKS have been reported by Žulj *et al.* (2018). We confirm 763 BKS and provide 1,955 new BKS. Only for two instances, we were not able

H&W CBD/traversal instances							H&W UDD/traversal instances						
	SC-2		BPC-DF-2		ALNS/TS			SC-2		BPC-DF-2		ALNS/TS	
n	Gp	t[s]	Gp	t[s]	Gp	t[s]	n	Gp	t[s]	Gp	t[s]	Gp	t[s]
20	0.10	0.1	0.00	0.6	0.05	0.4							
30	0.07	0.4	0.01	9.2	0.42	1.2	40	0.06	0.9	0.02	40.4	0.29	2.1
40	0.08	1.1	0.01	21.4	0.35	2.4	60	0.08	6.4	0.11	61.1	0.62	6.2
50	0.10	2.8	0.04	40.9	0.83	4.5	80	0.09	17.2	0.21	73.1	0.86	14.5
60	0.08	6.2	0.05	50.5	0.91	6.9	100	0.16	36.9	0.22	87.4	1.06	26.7
Total	0.09	2.1	0.02	24.5	0.51	3.1	Total	0.10	15.3	0.14	65.5	0.71	12.4

Table 2.7: Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the traversal strategy on a subset of the H&W instances

		SC-2		BPC-DF-2		ALNS/TS	
Q	n	Gp	t[s]	Gp	t[s]	Gp	t[s]
6	200	0.05	2.3	0.08	108.1	0.99	221.2
	300	0.03	8.0	0.11	108.1	1.18	747.9
	400	0.03	18.6	0.23	109.8	1.60	1,737.9
	500	0.02	40.2	0.48	120.0	1.84	3,388.3
	600	0.02	53.7	1.04	120.0	1.91	5,616.7
9	200	0.16	28.3	1.48	120.0	2.45	248.6
12	200	1.49	119.6	1.73	120.0	4.49	289.1
15	200	3.41	119.6	3.68	120.0	6.01	313.2
Total		0.65	48.8	1.10	115.8	2.56	1,570.4

Table 2.8: Comparison of our BPC-based heuristics SC-2 and BPC-DF-2 with the ALNS/TS of Žulj *et al.* (2018) for the traversal strategy on the large-scale ZKS instances

	SC heuristic						BPC-DF heuristic					
	SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Class	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
M&Ö	0.96	41.8	0.83	56.5	0.70	82.0	0.43	60.7	0.29	85.7	0.17	132.6
H&W	0.17	16.4	0.15	20.6	0.13	26.8	0.08	34.8	0.07	48.6	0.05	73.9
ZKS	0.48	52.1	0.41	68.9	0.36	98.6	0.84	118.1	0.63	177.0	0.51	294.7

Table 2.9: Summary results of our BPC-based heuristics with different time limits

to reach the previously reported BKS. For the ZKS benchmark, we improve the BKS for all 80 instances that have been considered before, i.e., those for routing strategy traversal.

For the larger instances of the benchmarks M&Ö-ext, W&G-g, and W&G-u, the considered hard time limits of two, three, or five minutes are sufficient for the straightforward BPC-based heuristics to consistently provide high-quality solutions. Indeed, in many cases, the provided solution is simply the best one of those provided by the randomized savings heuristic used for initializing the RMP (see Appendix 2.D) with gaps around 20 to 25% with respect to the best-known lower bound. More refined strategies like, e.g., premature termination of the CG process, seem necessary to tackle those instances with BPC-based heuristic approaches.

2.4.5 Comparison of Routing Strategies

Systematic studies on the quality of different routing strategies have been carried out, e.g., in (Petersen 1997, Roodbergen and de Koster 2001a, Hwang *et al.* 2004). To the best of our knowledge, the only evaluation of routing strategies in combination with optimal order batching decisions has recently been performed by Bahçeci and Öncan (2022) on the small-scale B&Ö instances. In Table 2.10, we analyze the quality of the considered routing strategies on the larger M&Ö and H&W instances. The table reports for each strategy the percentage increase in traveled distance compared to the optimal routing averaged by capacity. For the H&W instances, we further differentiate the two storage assignment policies CBD and UDD. Notice that the averages in Table 2.10 are taken over all instances using the BKS whenever an instance is not solved to proven optimality. Because of the very small gaps of these BKS, the impact on the results is marginal.

Table 2.10 reveals that the picking capacity is a major influencing factor for the quality of the routing heuristics relative to the optimal routing. Strategies return, midpoint, and largest gap become worse with increasing capacity, while strategies traversal and combined become better. The number of orders does not seem to have an impact on the relative quality of the heuristics (see detailed results in Appendix 2.F). Overall, the combined strategy provides the best results of the routing heuristics with travel distances only 2-3% longer than with optimal routing for large capacities. Furthermore, the largest gap strategy performs quite well for small capacities, while traversal does so for large capacities. Strategies return and midpoint perform rather poorly in general. Regarding the storage assignment, strategies largest gap and midpoint perform relatively better for CBD, while traversal, return, and combined perform relatively better for UDD.

Summing up, when applying a good routing heuristic, the loss compared to an optimal routing is around 2% in the best case (for large capacities and uniformly distributed demands) and around 5% on average. With a poor routing heuristic,

Q	Traversal	Return	Midpoint	Largest gap	Combined
Panel A: M&Ö instances					
24	10.4%	32.8%	9.9%	5.8%	3.7%
36	7.1%	34.9%	13.5%	8.2%	2.5%
48	5.3%	36.5%	17.1%	10.8%	1.8%
Total	7.6%	34.7%	13.4%	8.2%	2.7%
Panel B: H&W UDD instances					
30	17.4%	52.7%	15.4%	8.9%	7.2%
45	10.0%	53.9%	20.5%	12.5%	4.2%
60	7.5%	55.4%	24.6%	15.9%	2.9%
75	6.3%	56.9%	28.1%	19.0%	2.2%
Total	10.2%	54.7%	22.1%	14.0%	4.1%
Panel C: H&W CBD instances					
30	19.2%	52.2%	9.4%	5.4%	8.7%
45	12.0%	52.4%	12.4%	7.2%	5.7%
60	9.1%	53.2%	15.2%	9.3%	4.1%
75	7.5%	54.1%	18.0%	11.3%	3.2%
Total	11.8%	53.0%	13.7%	8.3%	5.4%

Table 2.10: Percentage increase in total traveled distances compared to the optimal routing strategy

the loss can be around 10–20% in many cases, going up to over 50% for the return strategy.

2.5 Conclusions

In this paper, we have proposed an exact *branch-price-and-cut* (BPC) algorithm for the *order batching problem* (OBP). A main building block of the approach is the representation of the *column generation* (CG) pricing problem as a *shortest path problem with resource constraints* (SPPRC), which can be adapted to handle the implications from non-robust valid inequalities and branching decisions. The SPPRC pricing problems are solved by means of an effective dynamic programming labeling algorithm that relies on strong completion bounds. To strengthen the underlying set-partitioning formulation of the OBP, two families of non-robust valid inequalities are used. Moreover, we have presented two heuristic approaches to the OBP that are based on the proposed BPC.

The focus of this paper has been on the OBP in a rectangular single-block parallel-aisles warehouse and routing strategies traversal, return, midpoint, largest gap, combined, and optimal. The proposed BPC and the derived heuristics, how-

ever, are much more generic. They can immediately be applied to variants of the OBP with different warehouse layouts and routing strategies, or including additional aspects such as scattered storage or a decoupling of picker and picking cart, whenever the corresponding picker routing problem for a given batch can be solved and the travel distances of the batches are given by a monotone function in the comprised orders. Even more, the proposed methods or slightly modified variants might be viable approaches to other optimization problems featuring a knapsack substructure with complex cost function such as the *job grouping problem* (Tang and Denardo 1988) or *bin packing problems* with general costs (Anily *et al.* 1994, Hu *et al.* 2018) that are relevant, e.g., in flexible manufacturing or courier logistics.

In an extensive computational campaign, we have highlighted the competitiveness of the proposed methods. Our BPC clearly outperforms the state-of-the-art exact approach of Muter and Öncan (2015) for the routing strategies (traversal, midpoint, return) and instances considered in their paper: it is faster by about two orders of magnitude and provides more than three times the number of proven optima (201 vs. 689 of 810 instances). Overall, our BPC is able to solve 90% of the benchmarks of Henn and Wäscher (2012) and Muter and Öncan (2015) and a small number of the large-scale instances from the benchmark of Žulj *et al.* (2018). Only a small fraction of these instances has been solved to optimality before. The two BPC-based heuristics substantially improve on the gaps reported for the state-of-the-art heuristic approach of Žulj *et al.* (2018) for the routing strategies (traversal, largest gap) and instances considered in their paper. For the Henn and Wäscher (2012) instances, computation times are comparable. For the large-scale Žulj *et al.* (2018) instances, our heuristics are faster by more than one order of magnitude on average. Overall, our heuristics improve almost 2,000 out of the 2,720 best-known solutions reported by Žulj *et al.* (2018), and confirm all remaining ones except for two instances.

As indicated by the computational results, the proposed CG method is very powerful and can solve the LP relaxation of the set-partitioning formulation very quickly. The BPC spends most of its time in the search tree trying to find an optimal integer solution and to prove its optimality. Viable avenues of future research may thus focus on techniques to raise the dual bounds more effectively and on more refined strategies to identify high-quality solutions within the BPC.

Another promising avenue of future research is to consider integrated optimization problems with an order-batching and picker-routing component such as the *joint planning of order batching, picker routing, and sequencing* (e.g., van Gils *et al.* 2019, Cano *et al.* 2020). The proposed BPC can serve as a central building block for exact and heuristic solution approaches to these problems.

Bibliography

Aerts, B., Cornelissens, T., and Sörensen, K. (2021). The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, **129**, 105168.

Anily, S., Bramel, J., and Simchi-Levi, D. (1994). Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operations Research*, **42**(2), 287–298.

Archetti, C., Bouchard, M., and Desaulniers, G. (2011). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, **45**(3), 285–298.

Azadeh, K., de Koster, R., and Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, **53**(4), 917–945.

Bahçeci, U. and Öncan, T. (2022). An evaluation of several combinations of routing and storage location assignment policies for the order batching problem. *International Journal of Production Research*, **60**(19), 5892–5911.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, **115**, 351–385.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Boysen, N., de Koster, R., and Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, **277**(2), 396–411.

Cano, J. A., Correa-Espinal, A. A., and Gómez-Montoya, R. A. (2020). Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems. *Journal of King Saud University – Engineering Sciences*, **32**(3), 219–228.

Caron, F., Marchet, G., and Perego, A. (2000). Optimal layout in low-level picker-to-part systems. *International Journal of Production Research*, **38**(1), 101–117.

Çelik, M. and Süral, H. (2014). Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE Transactions*, **46**(3), 283–300.

de Koster, R., van der Poort, E., and Wolters, M. (1999a). Efficient orderbatching methods in warehouses. *International Journal of Production Research*, **37**(7), 1479–1504.

de Koster, R., Roodbergen, K. J., and van Voorden, R. (1999b). Reduction of walking time in the distribution center of De Bijenkorf. In M. Speranza and P. Stähly, editors, *New Trends in Distribution Logistics*, pages 215–234. Springer, Berlin.

de Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, **182**(2), 481–501.

Establish Inc. (2013). Establish Davis logistics costs and service 2013. Presentation. CSCMPS Annual Global Conference, Denver.

Frazelle, E. (2001). *World-Class Warehousing and Material Handling*. McGraw-Hill Book Company, New York.

Gademann, N. and van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, **37**(1), 63–75.

Gademann, N., van den Berg, J., and van der Hoff, H. (2001). An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, **33**(5), 385–398.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 436–451.

Goetschalckx, M. and Ratliff, H. (1988). Order picking in an aisle. *IIE Transactions*, **20**(1), 53–62.

Grosse, E. H., Glock, C. H., and Ballester-Ripoll, R. (2014). A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. *International Journal of Operations and Quantitative Management*, **20**(2), 65–83.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Gu, J., Goetschalckx, M., and McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, **203**(3), 539–549.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, **25**(4), 76–87.

Henn, S. and Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, **222**(3), 484–494.

Henn, S., Koch, S., and Wäscher, G. (2012). Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini, editor, *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London.

Heßler, K. and Irnich, S. (2021). A branch-and-cut algorithm for the soft-clustered vehicle-routing problem. *Discrete Applied Mathematics*, **288**, 218–234.

Heßler, K. and Irnich, S. (2022). A note on the linearity of Ratliff and Rosenthal's algorithm for optimal picker routing. *Operations Research Letters*, **50**(2), 155–159.

Heßler, K., Gschwind, T., and Irnich, S. (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, **271**(2), 401–419.

Hintsch, T. and Irnich, S. (2020). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, **280**(1), 164–178.

Hong, S., Johnson, A. L., and Peters, B. A. (2012). Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, **44**(2), 88–106.

Hu, Q., Wei, L., and Lim, A. (2018). The two-dimensional vector packing problem with general costs. *Omega*, **74**, 59–69.

Hwang, H. and Kim, D. G. (2005). Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. *International Journal of Production Research*, **43**(17), 3657–3670.

Hwang, H., Oh, Y. H., and Lee, Y. K. (2004). An evaluation of routing policies for order-picking operations in low-level picker-to-part system. *International Journal of Production Research*, **42**(18), 3873–3889.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer Science & Business Media, Boston.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Marchet, G., Melacini, M., and Perotti, S. (2015). Investigating order picking system adoption: A case-study-based approach. *International Journal of Logistics Research and Applications*, **18**(1), 82–98.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Michel, R. (2016). 2016 Warehouse/DC Operations Survey: Ready to confront complexity. Supply Chain Management Review. `https://www.scmr.com/article/2016_warehouse_dc_operations_survey_ready_to_confront_complexity`, November 8.

Muter, İ. and Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, **47**(7), 728–738.

Öncan, T. (2015). MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. *European Journal of Operational Research*, **243**(1), 142–155.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, **100**, 117–127.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. *Decision Sciences Institute Proceedings*, **3**, 1614–1616.

Petersen, C. G. (1997). An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, **17**(11), 1098–1111.

Petersen, C. G. and Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, **92**(1), 11–19.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

Richards, G. (2017). *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. Kogan Page, London, 3rd edition.

Roodbergen, K. J. (2001). *Layout and routing methods for warehouses*. Ph.D. thesis, Erasmus University Rotterdam, Rotterdam, the Netherlands.

Roodbergen, K. J. and de Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, **39**(9), 1865–1883.

Roodbergen, K. J. and de Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, **133**(1), 32–43.

Tang, C. S. and Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part II: Minimization of the number of switching instants. *Operations Research*, **36**(5), 778–784.

Thia, F. (2008). MySQL Foodmart Database. Pentaho Wiki. `http://pentaho.dlpage.phi-integration.com/mondrian/mysql-foodmart-database`, May 8.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons, Hoboken, NJ, 4th edition.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2016). Modelling and solving the joint order batching and picker routing problem in inventories. In R. Cerulli, S. Fujishige, and A. R. Mahjoub, editors, *Combinatorial Optimization*, volume 9849 of *Lecture Notes in Computer Science*, pages 81–97, Cham, Switzerland. Springer International Publishing.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, **262**(3), 817–834.

van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, **277**(3), 814–830.

Žulj, I., Kramer, S., and Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, **264**(2), 653–664.

Appendix

2.A Detailed Description of Routing Strategies

In the following, we thoroughly describe the routing strategies return, midpoint, traversal, largest gap, composite, combined, and optimal for the rectangular parallel aisles single-block warehouse specified in Section 2.2. The descriptions provide all necessary details to clarify our exact interpretation of the strategies and to allow reproduction of our results. For completeness, we also provide a description of the mixed strategy that has recently been proposed by Bahçeci and Öncan (2022). We use the terms *required location* and *required aisle* for those storage locations and picking aisles, respectively, in which there is at least one item to pick, given a set of orders. Figures 2.6a–2.6i depict exemplary picker routes for picking the batch comprising orders 2, 4, and 5 and each routing strategy. The items and corresponding storage locations are highlighted in blue and the picker route following each strategy is shown with a dashed line.

Return Starting at the depot, the picker moves along the front cross aisle to the rightmost required aisle, i.e., the required aisle furthest from the depot. On the way, the picker enters each required aisle from the front cross aisle, travels towards the required location closest to the back cross aisle, makes a U-turn, and exits the aisle again to the front cross aisle. After exiting the rightmost required aisle, the picker returns to the depot on the front cross aisle.

Midpoint The warehouse is virtually divided into a front and a back part such that all storage locations closer to the front (back) cross aisle are assigned to the front (back) part. Storage locations that are exactly in the middle between front and back cross aisle are assigned to the front part. Starting at the depot, the picker moves along the front cross aisle to the leftmost required aisle, i.e., the required aisle closest to the depot. This aisle is traversed completely by entering from the front and exiting to the back cross aisle. The picker then moves along the back cross aisle to the rightmost required aisle. Similar to the return strategy, the picker enters from the back cross aisle all aisles that contain at least one required location in the back part of the warehouse, travels towards the required location closest to the middle, makes a U-turn, and exits the aisle again to the back cross aisle. The rightmost required aisle is traversed completely and the picker returns along the front cross aisle to the depot. All aisles between the left- and rightmost that contain at least one required location in the front part of the warehouse are visited from the front cross aisle in the analog fashion as those of the back part. In the special case that there is only a single required aisle, the route is the same as in the return strategy.

Traversal Starting at the depot, the picker moves horizontally to the rightmost required aisle. Each required aisle is traversed completely so that the picker enters from and exits to different cross aisles. After each traversal, the horizontal movement to the rightmost required aisle is continued on the opposite cross aisle. If the number of required aisles is even, the picker traverses the last one from the back to the front cross aisle and travels back to the depot on the front cross aisle. If the number of required aisles is odd, the picker visits the last one in a return fashion entering from and exiting to the front cross aisle and travels back to the depot on the front cross aisle.

Largest Gap This strategy is similar to the midpoint strategy. If there is only a single required aisle, the route is the same as in the return strategy. Otherwise, the picker travels horizontally on the front cross aisle from the depot to the leftmost required aisle, traverses this aisle completely, continues horizontally on the back cross aisle to the rightmost required aisle, traverses this aisle completely, and travels back to the depot on the front cross aisle. The other required aisles are visited on the way from/to the depot in a return fashion either from and to only one of the cross aisles or from and to both cross aisles, depending on the largest gap in this aisle. The largest gap in an aisle is the largest value of any of the following: (i) the distance between the front cross aisle and its closest required location, (ii) the distance between the back cross aisle and its closest required location, or (iii) the distance between any pair of required locations for which no third required location is closer to both locations. The aisle is then visited such that the largest gap in this aisle is not traversed. In Figure 2.6d, the largest gaps are highlighted in orange. If the largest gap is between the front (back) cross aisle and the required location closest to it as in the second (fourth) aisle, then a return from and to the back (front) cross aisle is performed. If the largest gap is between any two required locations (third aisle), then returns from and to both cross aisles are performed.

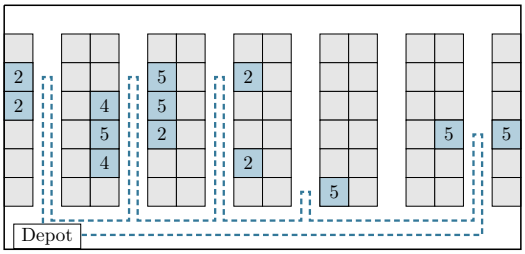
Composite The composite strategy combines elements of the traversal and return strategies. Starting from the depot and on the front cross aisle, the picker moves horizontally to the rightmost required aisle visiting all required aisles on the way and returns on the front cross aisle to the depot. Each required aisle is either traversed completely (changing from the front to the back cross aisle and vice versa) or visited in a return fashion entering from and exiting to the same cross aisle. The choice on how to visit an aisle is made individually for each aisle in a pure greedy fashion. There are two interpretations in the literature. For a given required aisle and the cross aisle on which the picker arrives at this aisle, Petersen (1995) chooses traversal or return based on which of the two gives a shorter distance between the farthest required location from the current cross aisle in the

current aisle and the farthest required location from the current cross aisle in the next required aisle. Roodbergen (2001) and Scholz and Wäscher (2017), on the other hand, choose traversal whenever the distance between the farthest required location and the current cross aisle is more than half of the distance of a full traversal. Otherwise, they choose a return visit. In both interpretations, the rightmost required aisle has to be visited such that the picker exits to the front cross aisle, i.e., performing a traversal if the picker arrives at the rightmost require aisle on the back cross aisle and performing a return visit otherwise.

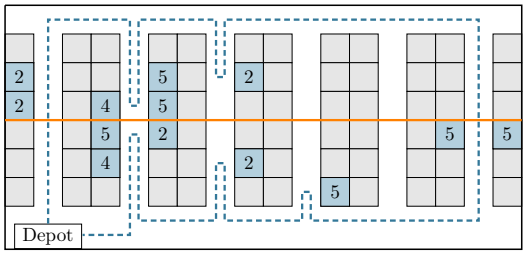
Combined The combined strategy is an enhanced version of the composite strategy. The only difference is that the choice whether an aisle is visited with a traversal or a return is not made individually for each aisle in a greedy fashion. Instead, these visits are performed such that the best possible route using only these in-aisle visits results. To this end, a simple DP algorithm can be used. We refer to Roodbergen (2001) for details.

Mixed The mixed routing strategy is similar to the midpoint and largest gap strategies. It adds elements of the return strategy to the midpoint strategy. The only difference from the midpoint strategy is that the required aisles between the leftmost and rightmost required aisles can be visited according to either the midpoint strategy, i.e., return visits to and from both cross aisles up to the middle of the aisles, or the return strategy, i.e., a single return visit to and from the same cross aisle. An alternative description is as follows. The mixed strategy differs from the largest gap strategy by allowing visits from and to both cross aisles only in the case that the gap that is not traversed is between two required locations that are on different parts (front and back) of the warehouse. We refer to Bahçeci and Öncan (2022) for details.

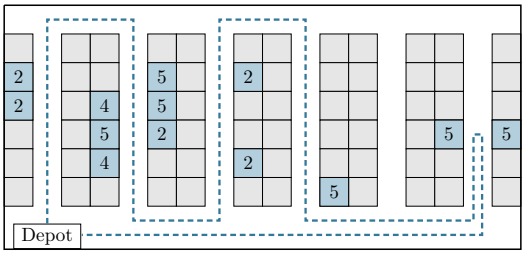
Optimal The optimal strategy follows a distance-minimal route of all possible picker routes which can in principle be computed by solving a *traveling salesman problem* (TSP) over the required locations. For a rectangular parallel-aisles single-block warehouse, the problem can be solved in linear time (linear in the sum of the number of aisles and the number of required locations) by means of a DP approach. We refer to Ratliff and Rosenthal (1983) for a detailed description of this DP algorithm. Note that the optimal strategy allows all possible in-aisle visits, i.e., traversal, a single return visit from the front or back cross aisle, and a double return visit from the front and back cross aisles as in the largest gap strategy, as well as all possible traversals from one required aisle to the next.



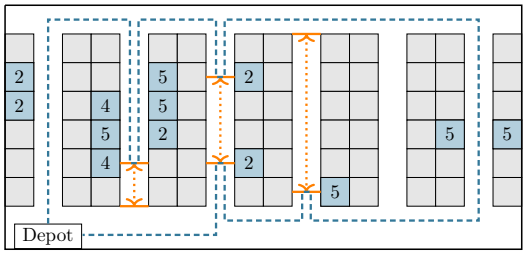
(a) Return strategy, $c_b = 68$



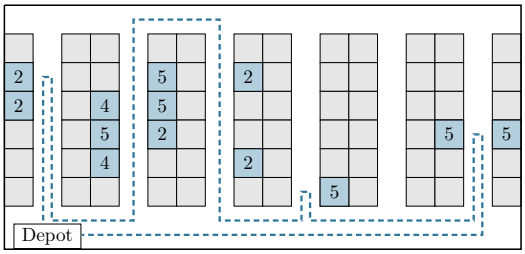
(b) Midpoint strategy, $c_b = 66$



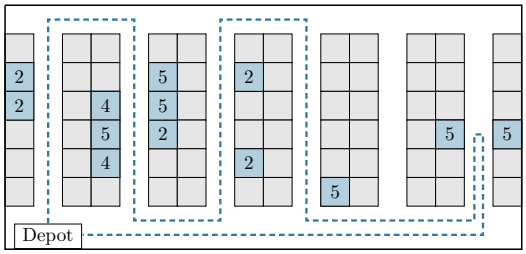
(c) Traversal strategy, $c_b = 64$



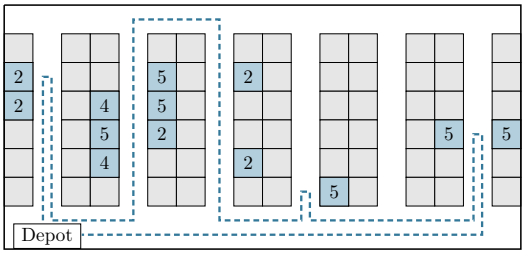
(d) Largest gap strategy, $c_b = 64$



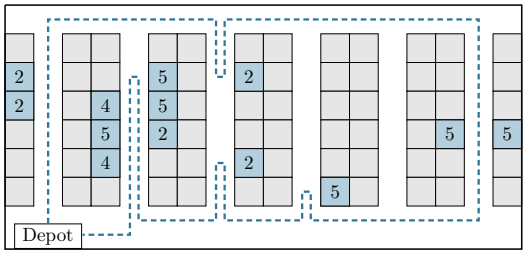
(e) Composite strategy (Petersen 1995), $c_b = 62$



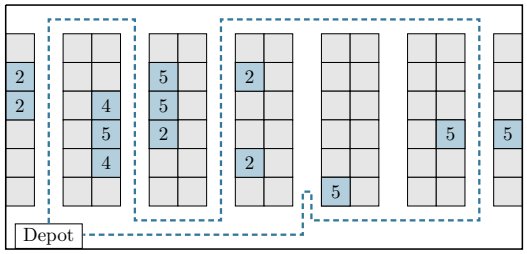
(f) Composite strategy (Roodbergen 2001), $c_b = 64$



(g) Combined strategy, $c_b = 62$



(h) Mixed strategy, $c_b = 64$



(i) Optimal strategy, $c_b = 60$

Figure 2.6: Picker routes for batch $b = \{2, 4, 5\}$ and different routing strategies

2.B Proof of Proposition 2.1

Proposition 2.1. *The routing strategies return, midpoint, traversal, largest gap, combined, mixed, and optimal are monotone.*

Proof. Let b_1 and b_2 be two feasible batches with $b_1 \subseteq b_2$. We need to show that $c_{b_1} \leq c_{b_2}$ holds for the routing strategies.

Let R denote the set of additional required locations of b_2 compared to b_1. Without loss of generality, we assume in the following that $R \neq \varnothing$ (otherwise $c_{b_1} \leq c_{b_2}$ obviously holds for all routing strategies). Note further that for any batch and all strategies except optimal, the total horizontal distance traveled on the cross aisles is exactly twice the distance from the depot to the rightmost required aisle. Thus, this distance strictly increases if there is an additional required location in R located in an aisle further from the depot than the rightmost required aisle of b_1. Otherwise it stays the same. For these strategies, it suffices to consider the distances traveled within the required aisles (including the distances to enter from/exit to the cross aisles) in the following.

Return Any additional required location in R that is located in a required aisle of b_1 but not closer to the back cross aisle than each required location of b_1 in this aisle does not change the distance traveled within the respective aisle. Any additional required location in R that is located either in a required aisle of b_1 and closer to the back cross aisle than each required location of b_1 in this aisle or in an aisle that is not required in b_1 strictly increases the distance traveled within the respective aisle. Thus, $c_{b_1} \leq c_{b_2}$ obviously holds for the return strategy.

Midpoint Consider first the special case of a single required aisle in b_1. If all additional required locations in R are also located in this aisle, then $c_{b_1} \leq c_{b_2}$ follows with the same arguments as for the return strategy. If there is at least one additional required location in R located in a different aisle, then the left- and rightmost required aisles are both traversed completely for b_2 implying a traveled distance of $2(L + 2a)$ within these aisles. The maximum possible distance traveled within the single required aisle of b_1 is $2(L - \ell/2 + a)$ if a required location is the one closest to the back cross aisle, so that $c_{b_1} \leq c_{b_2}$ also holds in this case.

Consider now the general case with multiple required aisles in b_1. If the left- or rightmost required aisles are not identical for b_1 and b_2, then different aisles are traversed completely in b_1 and b_2. The distances traveled, however, do not change. Then, $c_{b_1} \leq c_{b_2}$ immediately follows with similar arguments as in the return strategy.

Traversal We need to distinguish several cases:

(i) If the number of required aisles is even for b_1 and b_2, then any aisle traversed in b_1 is also traversed in b_2 and $c_{b_1} \leq c_{b_2}$ obviously holds.

(ii) If the number of required aisles is odd for b_1 and b_2 and the rightmost required aisle is identical for b_1 and b_2, then any aisle traversed in b_1 is also traversed in b_2 and, with the same arguments as for the return strategy, the distance traveled in the rightmost required aisle for b_2 cannot be smaller than the distance for b_1 so that $c_{b_1} \leq c_{b_2}$ has to hold.

(iii) If the number of required aisles is odd for b_1 and b_2 but the rightmost required aisle is not identical for b_1 and b_2, then any aisle traversed in b_1 is also traversed in b_2 and there are at least two additional required aisles in b_2. At least one of these is also traversed completely in b_2 (the other might be the rightmost required aisle of b_2 which is not traversed completely). In addition, the rightmost required aisle of b_1 is also traversed completely in b_2 instead of the return visit in b_1. For b_2, the total distance traveled within these two aisles is thus $2(L + 2a)$. For b_1, the maximum possible distance traveled within its rightmost required aisle is $2(L - \ell/2 + a)$ if a required location is the one closest to the back cross aisle, so that $c_{b_1} \leq c_{b_2}$ also holds in this case.

(iv) If the number of required aisles is odd for b_1 but even for b_2, then any aisle traversed in b_1 is also traversed in b_2, there is at least one additional required aisle in b_2, and the rightmost aisle is traversed completely for b_2 instead of the return visit for b_1. This additional required aisle as well as the rightmost required aisle of b_1 are traversed completely in b_2 resulting in a traveled distance of $2(L + 2a)$ within these two aisles. For b_1, the maximum possible distance traveled within its rightmost required aisle is $2(L - \ell/2 + a)$ if a required location is the one closest to the back cross aisle, so that $c_{b_1} \leq c_{b_2}$ also holds in this case.

(v) If the number of required aisles is even for b_1 but odd for b_2, then any aisle traversed in b_1 is also traversed in b_2 except for the rightmost required aisle of b_1, there is at least one additional required aisle in b_2, and the rightmost aisle is visited in return fashion for b_2 instead of the complete traversal for b_1. The traversal of the additional required aisle in b_2 obviously implies the same distance as the complete traversal of the rightmost required aisle in b_1, and it immediately follows that $c_{b_1} \leq c_{b_2}$.

Largest Gap The special case of a single required aisle in b_1 follows with the exact same arguments as for the midpoint strategy. For the general case of multiple required aisles in b_1, recall that required aisles are visited such that the largest gap between pairs of required locations or cross aisles and required

locations is not traveled. Any additional required location in R can clearly only decrease the largest gap in the corresponding aisle so that the distance traveled within this aisle can only increase. The relation $c_{b_1} \leq c_{b_2}$ then follows with similar arguments as for the midpoint strategy.

Combined The combined strategy allows visiting aisles either in return fashion or by complete traversal. It chooses the distance-minimal picker route using only these two in-aisle visits. Now, any additional required location in R does not impact the distance of a complete traversal and can only increase the distance traveled in a return visit of the corresponding aisle. Thus, $c_{b_1} \leq c_{b_2}$ obviously holds for the combined strategy.

Mixed The mixed strategy allows visiting aisles either in return or in midpoint fashion choosing for each aisle the shorter of the two possibilities. Any additional required location within an aisle can only increase the distance traveled for visiting this aisle in both return and midpoint fashion. Then, $c_{b_1} \leq c_{b_2}$ follows by the same arguments as for the midpoint and the largest gap strategies.

Optimal Recall that the optimal strategy follows a distance-minimal route that is equivalent to an optimal TSP tour over the required locations. Because the distances between all storage locations satisfy the triangle inequality, any additional required location in R can never decrease the length of an optimal TSP tour and we immediately have $c_{b_1} \leq c_{b_2}$.

□

2.C Non-Monotonicity of Composite Routing Strategy

Figure 2.7 provides a small example showing that the composite strategy is not monotone, for neither of the interpretations by Petersen (1995) and by Roodbergen (2001) and Scholz and Wäscher (2017).

For batch b_1 comprising only order 6, both variants of the composite strategy result in a picker route of length $c_{b_1} = 48$ depicted with a blue dashed line. For batch $b_2 = \{6, 7\}$, the corresponding picker route of both variants is indicated with a red dotted line and has a length of $c_{b_2} = 46$. Because $b_1 \subseteq b_2$ but $c_{b_1} > c_{b_2}$, the composite strategy is obviously not monotone.

2.D Algorithm Design Choices

In the following, we give some details on additional design choices made in our BPC algorithm. We also present the specific values used for the parameters of

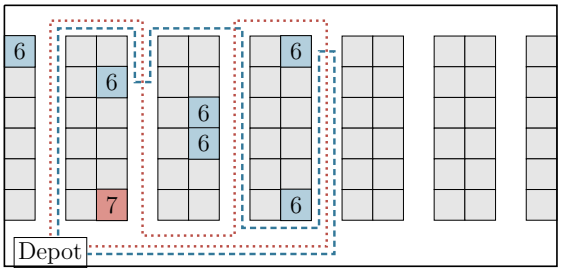


Figure 2.7: Picker routes for both interpretations of the composite strategy and batches $b_1 = \{6\}$ (in blue) and $b_2 = \{6, 7\}$ (in red), $c_{b_1} = 48$, $c_{b_2} = 46$

the algorithm. These values were obtained in pretests on a small subset of the instances used in our main computational study.

Initialization of RMP We initialize the RMP with a subset Ω' of feasible batches obtained from a variant of the well-known savings heuristic by Clarke and Wright (1964). The heuristic first calculates for each pair of customer orders, the savings in travel distance if the customer orders are picked in one picking route instead of two individual routes. Starting with individual batches for each order, the heuristic then iterates over the savings in non-decreasing order and combines the current batches of the two corresponding customer orders to one larger batch, if feasible. To randomize the heuristic, the savings are multiplied with a number randomly drawn from the interval $[0.85, 1.15]$. The heuristic is run several times and all batches contained in any of the heuristic solutions are added to Ω'.

We further initialize the RMP with a lower bound on the number of pickers needed by adding the corresponding inequality (2.6) for $S = O$ using $\kappa^3(O)$. Notice that in this case, no additional resource is needed in the labeling algorithm because the dual price ρ_O has to be subtracted once from all batches.

Pricing Problem Solution In principle, any sorting of the orders can be used when constructing graph G for the SPPRC representation of the pricing problem. The sorting, however, has a substantial impact on the solution time of the pricing problem. In our BPC, the orders are sorted non-increasingly by relative profit π_o/q_o or, after branching, by a generalized version that takes the maximum relative profit multiplied by the maximum capacity consumption of the feasible combinations of an order group. With this sorting, negative reduced-cost batches can often be identified early in the labeling allowing an early termination. Moreover, labels with positive reduced costs tend to be discarded early because of the small-valued completion bounds resulting from this sorting.

Cutting and Branching Strategy In the BPC, branching on the number of pickers, if fractional, is given priority over cutting. Furthermore, cuts are only separated in the root node or in its two child nodes if they result from a branching on the number of pickers.

The overall separation strategy is to first separate CCs with the greedy and connected component heuristics. If they fail to identify any violated cuts, we separate SRCs by enumeration. The computationally costly MIP-based separation of CCs is only invoked when the other separation procedures fail. Moreover, we set a hard time limit of five seconds for each call to the MIP.

To contain the size of the B&B tree, we apply strong branching at the second stage of the branching scheme. The strong branching procedure considers a candidate set of order pairs (o_1, o_2) with fractional $f_{o_1 o_2}$. For each pair, a rough evaluation of both child nodes is performed solving only the RMP with the corresponding branching constraint without any CG. The decision on which candidate branching is performed is taken according to the product rule (Achterberg 2007). At the root node, the maximum size of the candidate set is 25 and we decrease the size by two for each level of the B&B tree. We select the pairs (o_1, o_2) for which $f_{o_1 o_2}$ is closest to 0.5 to enter the candidate set.

2.E Benchmark Instances

In the following, we provide a description of the considered benchmark sets by Henn and Wäscher (2012) (`H&W`), Muter and Öncan (2015) (`M&Ö` and `M&Ö-ext`), and Žulj *et al.* (2018) (`ZKS`) as well as the newly introduced instances (`W&G-g` and `W&G-u`). The same warehouse layouts are shared by the `H&W` and `ZKS` instances and by the `M&Ö`, `M&Ö-ext`, `W&G-g`, and `W&G-u` instances, respectively.

The `H&W` and `ZKS` benchmarks consider a rectangular single-block warehouse with 10 parallel picking aisles and 45 storage locations on both sides of each aisle. Each storage location has a length of one unit. Picking an item takes place in the vertical middle of the corresponding storage location and does not require any horizontal distance to be traveled. When entering/leaving an aisle from/to one of the cross aisles, the order picker moves one unit in vertical direction. Thus, a complete traversal of a picking aisle amounts to 47 units. The depot is located on the front cross aisle in front of the leftmost aisle. There is no additional distance to enter/leave the depot to/from the front cross aisle. The distance between two consecutive picking aisles is five units.

The `M&Ö`, `M&Ö-ext`, `W&G-g`, and `W&G-u` benchmark sets assume a single-block layout with 10 parallel picking aisles, 10 storage locations of length one on both sides of each aisle, and a single depot located on the front cross aisle in front of the leftmost aisle. Picking is performed as in the `H&W` and `ZKS` instances. The horizontal distance between two consecutive picking aisles is 2.4 units. Unfortunately, we were

not able to get any information about the distances needed to enter/leave an aisle from/to one of the cross aisles or to enter/leave the depot for the M&Ö instances. We interpreted these distances as specified for the H&W and ZKS instances also for the benchmark sets M&Ö, M&Ö-ext, W&G-g, and W&G-u.

The H&W benchmark considers two different scenarios with respect to storage assignment: *class-based demand* (CBD) and *uniformly distributed demand* (UDD). For CDB, items are assigned to storage locations according to their demand frequencies: high-demand items in the leftmost aisle, medium-demand items in subsequent aisles, and low-demand items in the right half of the warehouse. For UDD, items are randomly assigned to storage locations. Henn and Wäscher (2012) originally introduced separate instances for strategies traversal and largest gap resulting in the four subclasses CBD/traversal, CBD/largest gap, UDD/traversal and UDD/largest gap. Obviously, all instances can be used for all routing strategies. For all subclasses, there are 40 instances for each combination of capacity $Q \in \{30, 45, 60, 75\}$ and number of orders $n \in \{20, 30, ..., 100\}$. The number of items per order is uniformly distributed in $\{5, ..., 25\}$. The complete benchmark comprises 5,760 instances.

The ZKS benchmark comprises groups of 10 instances for the (n, Q)-pairs $(200, 6)$, $(200, 9)$, $(200, 12)$, $(200, 15)$, $(300, 6)$, $(400, 6)$, $(500, 6)$, and $(600, 6)$ and order sizes uniformly distributed in $\{1, ..., 5\}$.

The original M&Ö benchmark consists of instance groups characterized by capacity $Q \in \{24, 36, 48\}$ and number of orders $n \in \{20, 30, ..., 100\}$. The sizes of the orders are randomly drawn from $\{2, ..., 10\}$ and the individual items are randomly distributed in the warehouse. Each group comprises 10 instances resulting in a total of 270 instances. We additionally consider the M&Ö instances with larger values for the capacities, namely $Q \in \{60, 72\}$, referred to as benchmark set M&Ö-ext which comprises another 180 instances.

The W&G-g benchmark comprises groups of 10 instances where each group is characterized by a capacity $Q \in \{24, 36, 48, 60, 72\}$ and a number of orders $n \in \{125, 150, ..., 250\}$. The order sizes are randomly drawn from $\{2, ..., 10\}$ and the individual items are randomly distributed in the warehouse. The benchmark comprises a total of 300 instances.

The W&G-u benchmark comprises groups of 10 instances where each group is characterized by a capacity $Q \in \{24, 36, 48, 60, 72\}$ and a number of orders $n \in \{100, 150, 200, 250\}$. The orders have a uniform size of six items and the individual items are randomly distributed in the warehouse. The benchmark comprises a total of 200 instances.

2.F Detailed Computational Results

In this section, we report detailed computational results of our BPC algorithm and the BPC-based heuristics for the six considered routing strategies traversal, return, midpoint, largest gap, combined and optimal. We report results for the three benchmark sets from the literature by Muter and Öncan (2015) (`M&Ö`), Henn and Wäscher (2012) (`H&W`), and Žulj *et al.* (2018) (`ZKS`). Furthermore, we report results for the extended Muter and Öncan (2015) instances with enlarged capacities (`M&Ö-ext`) and the newly created benchmark instances with general (`W&G-g`) and uniform (`W&G-u`) order weights. Finally, we provide a comparison of the routing strategies in terms of total traveled distances. Instance-by-instance results of our main BPC and the two BPC-based heuristics together with the best-known solution are provided at `https://logistik.wiwi.uni-kl.de/obp-bpc-detailedresults`.

Summary Results of BPC Algorithm

Tables 2.11–2.16 provide summary results for the six benchmark sets and all routing strategies aggregated by capacity Q and number of orders n. They report the number of instances solved to optimality within the time limit of one hour (Opt) and the average solution time in seconds ($t[s]$).

Detailed Results of BPC Algorithm

Tables 2.17–2.52 provide detailed results for the six benchmark sets and all routing strategies aggregated by capacity Q and number of orders n. They report the number of instances solved to optimality within the time limit of one hour (Opt), the average solution time in seconds ($t[s]$), the average time for solving the LP relaxation in seconds (t^{LP}), the average optimality gap with respect to the best known solution of the LP relaxation (Gp), the average optimality gap with respect to the best-known solution before the first node resulting from a Ryan-and-Foster branching is solved (Gp^{RF}), the average number of B&B nodes solved (Nds), and the average number of CCs (CC) and SRCs (SRC) added. In cases where no average could be computed for a given group, e.g., because the LP relaxation could not be solved for one of the comprised instances, the respective cell is left blank.

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
24	20	10	10	0.2	10	0.1	10	0.0	10	0.1	10	0.1	10	0.1
	30	10	10	0.7	10	1.1	10	0.4	10	0.9	10	0.4	10	0.7
	40	10	10	8.6	10	1.9	10	1.6	10	2.3	10	3.7	10	2.2
	50	10	10	14.8	10	13.0	10	3.0	10	8.1	10	9.2	10	6.3
	60	10	10	11.5	10	3.0	10	5.2	10	12.5	10	8.3	10	7.1
	70	10	10	37.9	10	73.8	10	17.3	10	53.3	10	118.5	10	59.3
	80	10	10	392.4	10	128.3	10	39.2	10	69.5	10	51.0	10	40.5
	90	10	8	1,097.6	10	570.1	10	269.3	9	457.4	10	406.2	9	557.6
	100	10	10	493.0	10	274.8	9	582.6	10	186.5	10	376.6	9	508.3
Subtot.		90	88	228.5	90	118.5	89	102.1	89	87.8	90	108.2	88	131.3
36	20	10	10	2.1	10	0.6	10	0.3	10	0.2	10	0.4	10	0.4
	30	10	10	25.4	10	5.6	10	2.5	10	3.2	10	2.8	10	6.5
	40	10	10	43.5	10	30.8	10	7.2	10	14.7	10	26.1	10	29.7
	50	10	9	649.2	10	59.3	10	27.1	10	116.2	10	163.2	10	59.9
	60	10	10	805.8	10	137.9	10	85.1	10	71.5	10	272.1	10	156.3
	70	10	8	1,125.5	10	791.9	10	160.4	10	373.2	10	737.4	10	359.3
	80	10	5	2,141.7	8	1,829.8	10	989.0	10	929.4	8	1,977.1	8	1,755.7
	90	10	6	2,295.7	10	1,098.1	10	1,402.3	7	1,941.4	9	1,425.8	7	1,969.3
	100	10	4	2,394.8	5	2,971.8	8	2,107.6	3	2,977.9	4	3,150.5	4	2,881.0
Subtot.		90	72	1,053.7	83	769.5	88	531.3	80	714.2	81	861.7	79	802.0
48	20	10	10	3.6	10	1.2	10	0.5	10	1.0	10	1.4	10	0.9
	30	10	10	146.5	10	21.8	10	46.4	10	35.6	10	49.1	10	96.5
	40	10	9	612.7	10	32.0	10	57.3	10	47.8	10	41.3	10	134.2
	50	10	6	2,285.0	10	613.0	10	187.0	10	402.9	9	1,202.6	9	1,030.0
	60	10	6	1,677.9	10	363.6	10	268.6	10	589.4	10	539.0	10	560.2
	70	10	3	2,951.5	5	2,208.9	6	2,286.7	5	2,254.9	7	1,746.4	7	1,887.1
	80	10	1	3,316.5	5	2,381.6	7	2,197.2	5	3,031.7	7	2,345.8	7	2,471.7
	90	10	1	3,276.8	1	3,387.3	3	3,160.0	3	2,828.2	0	3,600.0	1	3,574.5
	100	10	2	3,039.8	3	3,140.0	1	3,387.8	3	3,004.0	0	3,600.0	1	3,305.2
Subtot.		90	48	1,923.4	64	1,349.9	67	1,287.9	66	1,355.1	63	1,458.4	65	1,451.2
Total		270	208	1,068.5	237	746.0	244	640.4	235	719.0	234	809.4	232	794.8

Table 2.11: Summary results of our BPC algorithm for the M&Ö instances

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
30	20	160	160	0.0	160	0.0	160	0.0	160	0.0	160	0.0	160	0.0
	30	160	160	0.0	160	0.0	160	0.0	160	0.0	160	0.0	160	0.0
	40	160	160	0.1	160	0.1	160	0.0	160	0.0	160	0.0	160	0.0
	50	160	160	0.1	160	0.0	160	0.0	160	0.0	160	0.0	160	0.0
	60	160	158	45.2	160	0.1	160	0.1	160	0.1	160	0.1	160	0.1
	70	160	158	45.2	160	0.2	160	0.1	160	0.1	160	0.2	160	0.1
	80	160	155	129.2	160	0.2	160	0.1	160	0.2	159	22.9	160	13.9
	90	160	154	157.5	160	0.2	160	0.1	160	0.2	159	22.8	160	0.2
	100	160	155	123.1	160	0.3	160	0.1	160	0.3	160	0.3	160	0.4
Subtot.		1,440	1,420	55.6	1,440	0.1	1,440	0.1	1,440	0.1	1,438	5.2	1,440	1.6
45	20	160	160	0.2	160	0.1	160	0.1	160	0.1	160	0.1	160	0.1
	30	160	160	11.0	160	0.8	160	0.4	160	0.7	160	1.0	160	0.5
	40	160	160	34.1	160	1.6	160	1.1	160	2.6	160	1.7	160	2.1
	50	160	157	113.5	160	6.6	160	2.3	159	24.5	160	17.2	160	5.7
	60	160	152	221.5	159	35.2	160	6.9	160	7.9	159	81.6	160	15.7
	70	160	150	334.3	158	80.2	160	10.7	160	24.9	158	105.2	159	56.4
	80	160	145	491.4	156	146.5	160	22.0	160	32.1	157	157.0	157	142.6
	90	160	136	745.1	158	178.9	160	37.2	159	78.9	157	176.0	157	140.7
	100	160	132	818.8	148	432.2	158	105.9	156	182.5	148	396.6	151	346.6
Subtot.		1,440	1,352	307.8	1,419	98.0	1,438	20.7	1,434	39.3	1,419	104.1	1,424	78.9
60	20	160	160	1.3	160	0.3	160	0.2	160	0.2	160	0.3	160	0.3
	30	160	159	49.5	160	1.9	160	1.9	160	1.9	160	2.2	160	2.0
	40	160	149	325.4	160	8.0	160	5.0	160	8.3	160	12.9	160	11.9
	50	160	138	657.8	159	53.5	160	25.0	160	45.2	158	70.5	159	64.7
	60	160	130	932.1	159	132.8	158	127.6	159	142.9	157	212.4	159	130.8
	70	160	102	1,503.0	155	333.3	153	283.7	154	323.7	153	411.9	152	403.6
	80	160	79	2,031.4	146	626.3	147	501.5	142	693.6	147	671.5	135	951.9
	90	160	64	2,432.1	134	890.4	147	608.5	143	765.3	137	1,018.8	141	996.3
	100	160	39	2,898.1	123	1,381.1	139	932.1	136	1,152.7	123	1,456.0	120	1,542.8
Subtot.		1,440	1,020	1,203.4	1,356	380.8	1,384	276.2	1,374	348.2	1,355	428.5	1,346	456.0
75	20	160	160	36.8	160	0.4	160	0.3	160	0.5	160	0.8	160	0.9
	30	160	140	598.3	160	4.8	160	3.7	160	4.0	160	5.9	160	5.4
	40	160	130	881.6	160	28.3	160	18.8	160	26.0	160	29.4	160	36.1
	50	160	93	1,731.2	160	139.2	160	100.4	159	157.5	158	153.5	160	162.9
	60	160	83	2,031.8	155	360.1	157	304.6	155	382.2	155	375.6	157	457.1
	70	160	50	2,633.9	152	721.9	144	778.8	143	892.4	139	898.9	144	878.2
	80	160	36	2,966.8	122	1,473.2	134	1,088.1	125	1,426.8	129	1,364.3	108	1,835.8
	90	160	15	3,362.6	111	1,901.0	109	1,857.3	101	2,036.9	111	1,854.2	98	2,099.8
	100	160	13	3,399.2	67	2,758.6	78	2,441.8	65	2,712.8	77	2,499.2	57	2,838.7
Subtot.		1,440	720	1,960.2	1,247	820.8	1,262	732.6	1,228	848.8	1,249	798.0	1204	923.9
Total		5,760	4,512	881.8	5,462	325.0	5,524	257.4	5,476	309.1	5,461	333.9	5,414	365.1

Table 2.12: Summary results of our BPC algorithm for the `H&W` instances

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
6	200	10	2	2,884.8	6	1,648.2	2	2,978.5	4	2,172.0	4	2,513.2	3	2,524.3
	300	10	1	3,253.7	2	3,011.2	1	3,250.6	0	3,600.0	1	3,243.9	1	3,257.2
	400	10	1	3,259.1	0	3,600.0	0	3,600.0	1	3,265.4	0	3,600.0	0	3,600.0
	500	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	1	3,384.2	0	3,600.0
	600	10	1	3,581.2	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
9	200	10	0	3,600.0	1	3,344.2	0	3,600.0	0	3,600.0	3	2,675.6	2	2,991.5
12	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
15	200	10	0	3,600.0	1	3,510.9	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Total		80	5	3,422.4	10	3,239.3	3	3,478.6	5	3,379.7	9	3,277.1	6	3,346.6

Table 2.13: Summary results of our BPC algorithm for the `ZKS` instances

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
60	20	10	10	13.3	10	1.2	10	2.3	10	1.7	10	4.5	10	7.7
	30	10	10	330.1	10	99.2	10	38.6	10	60.6	10	215.0	10	444.3
	40	10	6	1,590.0	10	139.1	10	81.7	10	169.9	10	294.5	9	745.6
	50	10	7	1,791.3	9	535.5	10	850.1	10	557.8	8	1,387.5	8	1,112.2
	60	10	4	2,596.3	5	2,358.5	6	1,760.1	6	2,339.4	5	2,389.4	6	2,403.0
	70	10	2	3,090.8	7	1,685.2	3	2,823.2	3	3,145.2	5	2,645.1	2	3,220.8
	80	10	0	3,600.0	1	3,371.3	2	3,054.8	0	3,600.0	2	3,107.1	0	3,600.0
	90	10	1	3,262.8	2	3,063.7	1	3,357.8	0	3,600.0	1	3,317.4	1	3,402.8
	100	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		90	40	2,208.3	54	1,650.4	52	1,729.8	49	1,897.2	51	1,884.5	46	2,059.6
72	20	10	10	3.8	10	4.4	10	5.0	10	7.0	10	9.8	10	29.8
	30	10	10	368.8	10	74.6	10	226.4	10	249.0	10	195.5	10	1,010.6
	40	10	8	841.6	10	980.8	9	509.8	10	717.2	9	901.6	10	487.2
	50	10	4	2,728.1	6	1,760.4	8	1,525.1	7	1,798.4	6	1,692.2	8	1,417.6
	60	10	6	2,141.7	3	2,863.8	5	2,326.9	2	3,272.1	2	3,074.3	6	2,216.9
	70	10	2	2,903.7	2	3,227.4	2	3,187.4	0	3,600.0	3	2,825.8	0	3,600.0
	80	10	0	3,600.0	1	3,462.0	1	3,529.1	0	3,600.0	0	3,600.0	0	3,600.0
	90	10	1	3,356.4	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	100	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	1	3,316.4	0	3,600.0
Subtot.		90	41	2,171.6	42	2,174.8	45	2,056.6	39	2,271.5	41	2,135.1	44	2,173.6
Total		180	81	2,189.9	96	1,912.6	97	1,893.2	88	2,084.4	92	2,009.8	90	2,116.6

Table 2.14: Summary results of our BPC algorithm for the `M&Ö-ext` instances

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
24	125	10	5	2,259.2	8	1,423.7	6	1,694.5	7	1,557.1	7	1,974.4	7	1,730.9
	150	10	4	2,564.5	7	1,845.8	7	2,458.0	7	2,043.3	5	2,624.6	5	2,410.4
	175	10	3	2,733.7	1	3,508.7	5	3,076.1	3	3,169.5	3	3,164.7	4	2,671.1
	200	10	0	3,600.0	0	3,600.0	1	3,484.8	0	3,600.0	0	3,600.0	0	3,600.0
	225	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	1	3,576.3	0	3,600.0
Subtot.		60	12	3,059.6	16	2,929.7	19	2,985.6	17	2,928.3	16	3,090.0	16	2,935.4
36	125	10	0	3,600.0	1	3,489.9	4	2,985.8	1	3,367.5	1	3,518.5	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	175	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	225	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		60	0	3,600.0	1	3,581.6	4	3,497.6	1	3,561.3	1	3,586.4	0	3,600.0
48	125	10	1	3,445.3	0	3,600.0	1	3,561.2	0	3,600.0	0	3,600.0	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	175	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	225	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		60	1	3,574.2	0	3,600.0	1	3,593.5	0	3,600.0	0	3,600.0	0	3,600.0
60	125	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	175	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	225	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		60	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
72	125	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	175	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	225	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		60	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Total		300	13	3,486.8	17	3,462.3	24	3,455.3	18	3,457.9	17	3,495.3	16	3,467.1

Table 2.15: Summary results of our BPC algorithm for the `W&G-g` instances

			Traversal		Return		Midpoint		Largest gap		Combined		Optimal	
Q	n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]	Opt	t[s]
24	100	10	10	77.8	10	34.0	10	37.8	10	24.5	10	28.5	10	54.0
	150	10	7	1,447.1	10	777.8	10	577.4	9	679.2	8	1,538.9	10	1,118.0
	200	10	6	2,896.3	9	1,399.0	8	1,908.6	5	2,548.2	6	2,394.1	3	3,150.8
	250	10	1	3,391.7	1	3,290.1	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		40	24	1,953.2	30	1,375.2	28	1,531.0	24	1,713.0	24	1,890.4	23	1,980.7
36	100	10	7	2,235.3	6	2,793.7	8	1,737.4	9	1,809.8	5	2,249.7	7	2,174.5
	150	10	2	3,006.6	1	3,336.6	0	3,600.0	1	3,380.1	1	3,415.4	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		40	9	3,110.5	7	3,332.6	8	3,134.4	10	3,097.5	6	3,216.3	7	3,243.6
48	100	10	2	3,320.0	3	3,251.1	1	3,538.3	0	3,600.0	1	3,577.4	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		40	2	3,530.0	3	3,512.8	1	3,584.6	0	3,600.0	1	3,594.3	0	3,600.0
60	100	10	1	3,542.6	0	3,600.0	0	3,600.0	0	3,600.0	1	3,423.7	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		40	1	3,585.6	0	3,600.0	0	3,600.0	0	3,600.0	1	3,555.9	0	3,600.0
72	100	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	150	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	200	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
	250	10	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Subtot.		40	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0	0	3,600.0
Total		200	36	3,155.9	40	3,084.1	37	3,090.0	34	3,122.1	32	3,171.4	30	3,204.9

Table 2.16: Summary results of our BPC algorithm for the `W&G-u` instances

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.2	0.0	1.68	0.25	5	13	12
	30	10	10	0.7	0.0	1.81	0.31	12	21	30
	40	10	10	8.6	0.0	1.01	0.28	258	18	37
	50	10	10	14.8	0.1	1.11	0.44	107	33	41
	60	10	10	11.5	0.1	0.82	0.32	95	10	43
	70	10	10	37.9	0.2	0.77	0.30	212	32	53
	80	10	10	392.4	0.2	0.62	0.31	5,047	17	53
	90	10	8	1,097.6	0.3	0.56	0.35	13,403	22	48
	100	10	10	493.0	0.4	0.44	0.22	1,977	27	59
Subtotal		90	88	228.5	0.2	0.98	0.31	2,346	21	42
36	20	10	10	2.1	0.0	4.57	0.26	6	29	48
	30	10	10	25.4	0.1	2.13	0.60	47	41	70
	40	10	10	43.5	0.3	2.17	0.49	54	44	86
	50	10	9	649.2	0.6	2.18	0.73	1,129	58	98
	60	10	10	805.8	0.9	2.16	0.65	2,457	35	88
	70	10	8	1,125.5	1.3	1.54	0.54	2,714	41	101
	80	10	5	2,141.7	2.0	1.66	0.54	4,834	50	104
	90	10	6	2,295.7	2.5	1.46	0.53	5,530	42	116
	100	10	4	2,394.8	3.1	1.20	0.56	6,922	29	110
Subtotal		90	72	1,053.7	1.2	2.12	0.54	2,633	41	91
48	20	10	10	3.6	0.1	6.29	0.14	5	33	60
	30	10	10	146.5	0.4	5.00	0.50	50	81	101
	40	10	9	612.7	0.9	2.65	0.60	1,297	37	118
	50	10	6	2,285.0	2.3	3.06	0.84	2,539	72	112
	60	10	6	1,677.9	3.7	1.64	0.70	1,885	22	122
	70	10	3	2,951.5	5.2	2.28	0.89	2,344	62	126
	80	10	1	3,316.5	7.7	2.02	0.90	4,711	41	123
	90	10	1	3,276.8	13.4	2.33	0.82	1,333	85	127
	100	10	2	3,039.8	14.0	1.56	0.74	1,590	63	128
Subtotal		90	48	1,923.4	5.3	2.98	0.68	1,750	55	113
Total		270	208	1,068.5	2.2	2.03	0.51	2,243	39	82

Table 2.17: Detailed results of our BPC algorithm for the M&Ö instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.1	0.0	1.38	0.10	3	6	11
	30	10	10	1.1	0.0	1.67	0.30	18	25	30
	40	10	10	1.9	0.0	0.87	0.24	24	16	36
	50	10	10	13.0	0.1	0.83	0.24	62	32	41
	60	10	10	3.0	0.1	0.59	0.16	18	9	41
	70	10	10	73.8	0.2	0.68	0.27	305	34	44
	80	10	10	128.3	0.3	0.54	0.28	932	14	47
	90	10	10	570.1	0.3	0.52	0.29	7,851	10	44
	100	10	10	274.8	0.4	0.46	0.25	1,209	27	45
Subtotal		90	90	118.5	0.2	0.84	0.24	1,158	19	38
36	20	10	10	0.6	0.0	3.61	0.11	4	28	38
	30	10	10	5.6	0.1	1.69	0.20	9	41	48
	40	10	10	30.8	0.3	2.14	0.33	31	45	62
	50	10	10	59.3	0.5	1.61	0.36	41	67	76
	60	10	10	137.9	0.9	1.85	0.42	166	42	80
	70	10	10	791.9	1.3	1.30	0.44	450	65	80
	80	10	8	1,829.8	2.0	1.53	0.50	3,322	67	86
	90	10	10	1,098.1	2.6	1.22	0.40	2,111	46	76
	100	10	5	2,971.8	3.3	1.21	0.55	5,732	53	90
Subtotal		90	83	769.5	1.2	1.79	0.37	1,318	51	71
48	20	10	10	1.2	0.1	5.04	0.00	2	37	25
	30	10	10	21.8	0.5	4.04	0.17	6	69	70
	40	10	10	32.0	1.2	2.26	0.31	14	49	86
	50	10	10	613.0	2.8	2.70	0.51	72	111	106
	60	10	10	363.6	4.3	1.47	0.39	60	60	103
	70	10	5	2,208.9	6.7	2.02	0.65	354	101	105
	80	10	5	2,381.6	11.4	1.93	0.87	1,478	79	109
	90	10	1	3,387.3	13.0	2.43	1.00	1,090	112	122
	100	10	3	3,140.0	16.7	1.75	0.92	1,000	88	118
Subtotal		90	64	1,349.9	6.3	2.63	0.54	453	78	94
Total		270	237	746.0	2.6	1.75	0.38	976	49	67

Table 2.18: Detailed results of our BPC algorithm for the M&Ö instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.0	0.0	0.81	0.04	2	4	3
	30	10	10	0.4	0.0	1.08	0.13	6	12	21
	40	10	10	1.6	0.0	0.70	0.22	23	9	28
	50	10	10	3.0	0.1	0.63	0.21	17	12	33
	60	10	10	5.2	0.1	0.70	0.23	31	8	42
	70	10	10	17.3	0.2	0.64	0.25	82	14	38
	80	10	10	39.2	0.3	0.49	0.25	276	6	43
	90	10	10	269.3	0.4	0.49	0.28	2,429	16	42
	100	10	9	582.6	0.4	0.47	0.31	5,873	8	42
Subtotal		90	89	102.1	0.2	0.67	0.21	971	10	32
36	20	10	10	0.3	0.0	2.58	0.10	3	15	26
	30	10	10	2.5	0.2	1.09	0.13	4	21	41
	40	10	10	7.2	0.3	1.62	0.26	16	25	56
	50	10	10	27.1	0.8	1.36	0.31	30	34	68
	60	10	10	85.1	1.2	1.50	0.43	140	22	68
	70	10	10	160.4	1.9	1.09	0.46	254	26	73
	80	10	10	989.0	2.8	1.42	0.52	2,111	37	80
	90	10	10	1,402.3	3.7	1.19	0.48	3,509	25	86
	100	10	8	2,107.6	4.0	0.94	0.50	6,743	16	77
Subtotal		90	88	531.3	1.7	1.42	0.35	1,423	24	64
48	20	10	10	0.5	0.1	4.19	0.01	2	32	30
	30	10	10	46.4	0.7	3.41	0.33	13	73	84
	40	10	10	57.3	1.7	1.89	0.36	25	44	80
	50	10	10	187.0	4.6	2.44	0.51	73	71	100
	60	10	10	268.6	6.9	1.46	0.62	272	21	100
	70	10	6	2,286.7	12.8	2.00	0.91	1,819	67	104
	80	10	7	2,197.2	18.9	1.64	0.78	1,602	38	109
	90	10	3	3,160.0	25.6	2.13	1.01	1,385	65	112
	100	10	1	3,387.8	31.9	2.11	1.53	2,451	47	106
Subtotal		90	67	1,287.9	11.5	2.36	0.67	849	51	92
Total		270	244	640.4	4.4	1.48	0.41	1,081	28	63

Table 2.19: Detailed results of our BPC algorithm for the M&Ö instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.1	0.0	1.16	0.02	2	7	9
	30	10	10	0.9	0.0	1.22	0.19	12	14	27
	40	10	10	2.3	0.1	0.77	0.19	34	12	28
	50	10	10	8.1	0.1	0.74	0.29	51	13	38
	60	10	10	12.5	0.2	0.67	0.25	94	6	41
	70	10	10	53.3	0.2	0.63	0.26	261	16	42
	80	10	10	69.5	0.3	0.49	0.27	487	6	46
	90	10	9	457.4	0.4	0.49	0.28	5,550	12	43
	100	10	10	186.5	0.5	0.41	0.26	1,114	8	43
Subtotal		90	89	87.8	0.2	0.73	0.22	845	10	35
36	20	10	10	0.2	0.1	2.46	0.00	1	12	14
	30	10	10	3.2	0.2	1.22	0.20	6	23	41
	40	10	10	14.7	0.4	1.72	0.23	22	30	59
	50	10	10	116.2	1.2	1.48	0.42	166	40	70
	60	10	10	71.5	1.5	1.49	0.38	167	24	69
	70	10	10	373.2	2.5	1.16	0.51	444	27	73
	80	10	10	929.4	3.8	1.45	0.50	1,712	40	78
	90	10	7	1,941.4	4.4	1.21	0.51	6,519	22	73
	100	10	3	2,977.9	5.8	1.23	0.76	6,745	18	78
Subtotal		90	80	714.2	2.2	1.49	0.39	1,754	26	62
48	20	10	10	1.0	0.2	4.16	0.04	2	24	26
	30	10	10	35.6	1.1	3.59	0.24	10	65	84
	40	10	10	47.8	2.5	1.97	0.32	23	38	80
	50	10	10	402.9	7.2	2.52	0.60	187	60	99
	60	10	10	589.4	12.4	1.48	0.69	448	26	99
	70	10	5	2,254.9	18.9	1.99	0.88	1,162	62	100
	80	10	5	3,031.7	31.2	1.60	0.79	2,119	32	101
	90	10	3	2,828.2	37.2	2.44	1.28	1,580	67	103
	100	10	3	3,004.0	50.5	1.80	1.26	1,290	44	105
Subtotal		90	66	1,355.1	17.9	2.40	0.68	758	46	89
Total		270	235	719.0	6.8	1.54	0.43	1,119	28	62

Table 2.20: Detailed results of our BPC algorithm for the M&Ö instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.1	0.0	1.53	0.19	5	8	14
	30	10	10	0.4	0.0	1.50	0.11	6	16	18
	40	10	10	3.7	0.0	0.93	0.32	38	18	34
	50	10	10	9.2	0.1	0.92	0.33	54	25	34
	60	10	10	8.3	0.1	0.65	0.29	63	5	38
	70	10	10	118.5	0.2	0.73	0.33	1,320	22	40
	80	10	10	51.0	0.3	0.56	0.28	229	12	47
	90	10	10	406.2	0.4	0.46	0.27	3,806	11	47
	100	10	10	376.6	0.4	0.43	0.24	1,208	24	53
Subtotal		90	90	108.2	0.2	0.86	0.26	748	16	36
36	20	10	10	0.4	0.0	3.50	0.06	3	24	28
	30	10	10	2.8	0.1	1.53	0.24	8	29	50
	40	10	10	26.1	0.3	2.13	0.43	40	46	62
	50	10	10	163.2	0.6	1.93	0.51	121	65	78
	60	10	10	272.1	0.9	1.95	0.52	510	49	82
	70	10	10	737.4	1.4	1.39	0.53	1,499	46	85
	80	10	8	1,977.1	2.3	1.66	0.61	4,992	47	87
	90	10	9	1,425.8	2.6	1.25	0.45	2,761	38	88
	100	10	4	3,150.5	3.3	1.34	0.69	8,890	39	95
Subtotal		90	81	861.7	1.3	1.86	0.45	2,092	43	73
48	20	10	10	1.4	0.1	5.48	0.09	3	34	45
	30	10	10	49.1	0.5	4.66	0.40	19	72	100
	40	10	10	41.3	1.1	2.29	0.38	32	48	95
	50	10	9	1,202.6	2.9	3.14	0.72	315	98	109
	60	10	10	539.0	4.1	1.59	0.57	311	48	112
	70	10	7	1,746.4	6.7	2.03	0.64	518	82	115
	80	10	7	2,345.8	9.1	1.78	0.68	3,816	48	118
	90	10	0	3,600.0	12.0	2.62	1.17	2,247	91	120
	100	10	0	3,600.0	15.2	1.96	1.21	2,587	55	125
Subtotal		90	63	1,458.4	5.8	2.84	0.65	1,094	64	104
Total		270	234	809.4	2.4	1.85	0.45	1,311	41	71

Table 2.21: Detailed results of our BPC algorithm for the M&Ö instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	20	10	10	0.1	0.0	1.36	0.15	3	8	14
	30	10	10	0.7	0.0	1.53	0.20	9	15	25
	40	10	10	2.2	0.1	0.89	0.27	22	14	37
	50	10	10	6.3	0.2	0.75	0.23	38	18	36
	60	10	10	7.1	0.2	0.59	0.24	42	7	36
	70	10	10	59.3	0.3	0.69	0.26	342	20	44
	80	10	10	40.5	0.5	0.51	0.23	163	10	45
	90	10	9	557.6	0.6	0.49	0.29	6,205	7	46
	100	10	9	508.3	0.7	0.41	0.23	2,803	13	41
Subtotal		90	88	131.3	0.3	0.80	0.23	1,070	12	36
36	20	10	10	0.4	0.1	3.30	0.05	2	24	22
	30	10	10	6.5	0.3	1.48	0.20	8	38	50
	40	10	10	29.7	0.7	2.04	0.37	30	47	64
	50	10	10	59.9	1.4	1.61	0.42	74	38	69
	60	10	10	156.3	1.9	1.76	0.44	199	38	76
	70	10	10	359.3	2.9	1.35	0.47	443	43	82
	80	10	8	1,755.7	4.8	1.56	0.51	4,018	46	76
	90	10	7	1,969.3	5.3	1.31	0.49	4,764	34	84
	100	10	4	2,881.0	6.8	1.26	0.65	5,905	30	94
Subtotal		90	79	802.0	2.7	1.74	0.40	1,716	38	69
48	20	10	10	0.9	0.2	4.89	0.01	2	34	22
	30	10	10	96.5	1.4	4.31	0.31	11	83	87
	40	10	10	134.2	3.4	2.45	0.45	38	55	95
	50	10	9	1,030.0	7.2	3.07	0.73	228	96	110
	60	10	10	560.2	10.3	1.50	0.55	283	43	100
	70	10	7	1,887.1	17.7	2.16	0.79	466	83	114
	80	10	7	2,471.7	27.7	1.67	0.58	1,498	54	123
	90	10	1	3,574.5	38.3	2.51	1.18	1,091	84	118
	100	10	1	3,305.2	43.5	2.07	1.35	1,401	67	121
Subtotal		90	65	1,451.2	16.6	2.74	0.66	558	67	99
Total		270	232	794.8	6.5	1.76	0.43	1,114	39	68

Table 2.22: Detailed results of our BPC algorithm for the M&Ö instances and the optimal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.71	0.04	3	1	0
	30	160	160	0.0	0.0	0.42	0.04	5	2	1
	40	160	160	0.1	0.0	0.30	0.04	14	6	1
	50	160	160	0.1	0.0	0.28	0.02	7	4	1
	60	160	158	45.2	0.0	0.27	0.03	966	8	1
	70	160	158	45.2	0.0	0.21	0.03	387	10	2
	80	160	155	129.2	0.0	0.20	0.03	1,295	12	2
	90	160	154	157.5	0.0	0.16	0.02	1,230	10	1
	100	160	155	123.1	0.0	0.17	0.02	888	16	2
Subtotal		1,440	1,420	55.6	0.0	0.30	0.03	533	8	1
45	20	160	160	0.2	0.0	1.15	0.28	13	8	10
	30	160	160	11.0	0.0	0.73	0.29	933	9	15
	40	160	160	34.1	0.0	0.61	0.24	1,845	14	19
	50	160	157	113.5	0.0	0.58	0.26	5,001	22	26
	60	160	152	221.5	0.1	0.47	0.23	7,458	21	29
	70	160	150	334.3	0.1	0.43	0.21	10,293	25	31
	80	160	145	491.4	0.2	0.41	0.20	10,162	27	36
	90	160	136	745.1	0.2	0.38	0.19	13,483	33	38
	100	160	132	818.8	0.3	0.36	0.19	14,740	30	41
Subtotal		1,440	1,352	307.8	0.1	0.57	0.23	7,103	21	27
60	20	160	160	1.3	0.0	2.03	0.35	71	15	25
	30	160	159	49.5	0.0	1.33	0.36	2,911	12	31
	40	160	149	325.4	0.1	1.05	0.39	11,555	16	43
	50	160	138	657.8	0.2	0.88	0.36	21,302	15	50
	60	160	130	932.1	0.3	0.80	0.34	21,348	21	55
	70	160	102	1,503.0	0.4	0.73	0.34	29,825	24	60
	80	160	79	2,031.4	0.6	0.66	0.32	33,341	26	68
	90	160	64	2,432.1	0.9	0.61	0.31	33,915	25	75
	100	160	39	2,898.1	1.2	0.59	0.29	30,792	32	81
Subtotal		1,440	1,020	1,203.4	0.4	0.96	0.34	20,562	21	54
75	20	160	160	36.8	0.0	2.30	0.29	1,512	18	45
	30	160	140	598.3	0.1	1.79	0.39	16,026	17	53
	40	160	130	881.6	0.2	1.55	0.38	19,846	21	67
	50	160	93	1,731.2	0.4	1.41	0.42	27,441	27	72
	60	160	83	2,031.8	0.7	1.12	0.40	26,043	26	83
	70	160	50	2,633.9	1.1	0.99	0.36	28,428	30	93
	80	160	36	2,966.8	1.5	0.87	0.35	27,565	35	96
	90	160	15	3,362.6	2.2	0.86	0.36	24,390	39	105
	100	160	13	3,399.2	2.8	0.75	0.33	22,555	39	108
Subtotal		1,440	720	1,960.2	1.0	1.29	0.36	21,534	28	80
Total		5,760	4,512	881.8	0.4	0.78	0.24	12,433	19	41

Table 2.23: Detailed results of our BPC algorithm for the `H&W` instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.50	0.02	2	1	0
	30	160	160	0.0	0.0	0.27	0.01	3	1	1
	40	160	160	0.1	0.0	0.23	0.03	9	3	1
	50	160	160	0.0	0.0	0.19	0.01	3	2	1
	60	160	160	0.1	0.0	0.17	0.01	5	3	1
	70	160	160	0.2	0.0	0.13	0.01	7	4	1
	80	160	160	0.2	0.0	0.12	0.01	17	3	1
	90	160	160	0.2	0.0	0.10	0.01	10	3	2
	100	160	160	0.3	0.0	0.11	0.01	7	4	2
Subtotal		1,440	1,440	0.1	0.0	0.20	0.01	7	3	1
45	20	160	160	0.1	0.0	0.90	0.15	8	7	7
	30	160	160	0.8	0.0	0.62	0.19	64	9	14
	40	160	160	1.6	0.0	0.50	0.19	61	10	16
	50	160	160	6.6	0.0	0.50	0.21	262	16	20
	60	160	159	35.2	0.1	0.39	0.19	1,294	12	24
	70	160	158	80.2	0.1	0.37	0.18	2,553	17	25
	80	160	156	146.5	0.1	0.33	0.17	3,636	15	27
	90	160	158	178.9	0.2	0.30	0.16	3,526	15	28
	100	160	148	432.2	0.2	0.31	0.17	8,045	12	32
Subtotal		1,440	1,419	98.0	0.1	0.47	0.18	2,161	13	21
60	20	160	160	0.3	0.0	1.70	0.18	8	17	19
	30	160	160	1.9	0.0	1.14	0.26	72	19	28
	40	160	160	8.0	0.1	0.89	0.28	106	20	34
	50	160	159	53.5	0.2	0.77	0.31	1,209	21	40
	60	160	159	132.8	0.3	0.70	0.31	2,314	27	43
	70	160	155	333.3	0.4	0.65	0.30	5,415	26	45
	80	160	146	626.3	0.6	0.61	0.31	7,452	24	48
	90	160	134	890.4	0.8	0.54	0.30	8,265	23	53
	100	160	123	1,381.1	1.0	0.56	0.33	10,414	27	54
Subtotal		1,440	1,356	380.8	0.4	0.84	0.29	3,917	23	40
75	20	160	160	0.4	0.0	2.11	0.09	4	21	24
	30	160	160	4.8	0.1	1.68	0.26	22	27	45
	40	160	160	28.3	0.3	1.45	0.34	102	33	53
	50	160	160	139.2	0.5	1.31	0.37	992	38	56
	60	160	155	360.1	0.8	1.07	0.38	2,263	39	59
	70	160	152	721.9	1.3	0.95	0.38	3,866	37	65
	80	160	122	1,473.2	1.9	0.94	0.43	5,625	44	70
	90	160	111	1,901.0	2.7	0.91	0.46	6,157	42	71
	100	160	67	2,758.6	3.3	0.99	0.59	9,209	42	77
Subtotal		1,440	1,247	820.8	1.2	1.26	0.37	3,138	36	58
Total		5,760	5,462	325.0	0.4	0.69	0.21	2,306	18	30

Table 2.24: Detailed results of our BPC algorithm for the `H&W` instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.48	0.03	3	1	0
	30	160	160	0.0	0.0	0.26	0.01	2	1	0
	40	160	160	0.0	0.0	0.20	0.01	2	1	1
	50	160	160	0.0	0.0	0.16	0.01	2	1	0
	60	160	160	0.1	0.0	0.16	0.01	3	2	1
	70	160	160	0.1	0.0	0.12	0.01	3	2	1
	80	160	160	0.1	0.0	0.11	0.01	5	3	1
	90	160	160	0.1	0.0	0.09	0.00	3	2	1
	100	160	160	0.1	0.0	0.08	0.00	3	3	1
Subtotal		1,440	1,440	0.1	0.0	0.18	0.01	3	2	1
45	20	160	160	0.1	0.0	0.69	0.12	5	5	6
	30	160	160	0.4	0.0	0.49	0.15	17	6	11
	40	160	160	1.1	0.0	0.40	0.15	45	6	16
	50	160	160	2.3	0.0	0.40	0.15	72	8	19
	60	160	160	6.9	0.1	0.33	0.16	227	6	21
	70	160	160	10.7	0.1	0.32	0.16	218	5	23
	80	160	160	22.0	0.2	0.30	0.15	332	7	27
	90	160	160	37.2	0.2	0.30	0.15	529	6	27
	100	160	158	105.9	0.3	0.29	0.16	1,683	5	30
Subtotal		1,440	1,438	20.7	0.1	0.39	0.15	347	6	20
60	20	160	160	0.2	0.0	1.08	0.11	4	13	15
	30	160	160	1.9	0.0	0.88	0.25	43	12	26
	40	160	160	5.0	0.1	0.72	0.30	57	13	31
	50	160	160	25.0	0.2	0.63	0.30	586	12	36
	60	160	158	127.6	0.3	0.63	0.34	2,784	16	40
	70	160	153	283.7	0.5	0.58	0.33	5,145	13	40
	80	160	147	501.5	0.6	0.54	0.32	6,486	12	42
	90	160	147	608.5	0.9	0.50	0.31	5,916	14	46
	100	160	139	932.1	1.2	0.49	0.32	7,246	11	48
Subtotal		1,440	1,384	276.2	0.4	0.67	0.29	3,141	13	36
75	20	160	160	0.3	0.0	1.48	0.08	3	18	21
	30	160	160	3.7	0.1	1.32	0.24	23	20	39
	40	160	160	18.8	0.3	1.17	0.34	128	26	47
	50	160	160	100.4	0.6	1.09	0.43	684	30	50
	60	160	157	304.6	0.9	0.94	0.44	2,179	27	56
	70	160	144	778.8	1.5	0.86	0.47	5,038	24	57
	80	160	134	1,088.1	2.2	0.78	0.45	4,933	29	59
	90	160	109	1,857.3	3.1	0.84	0.52	7,098	28	62
	100	160	78	2,441.8	4.0	0.90	0.61	8,481	27	66
Subtotal		1,440	1,262	732.6	1.4	1.04	0.40	3,174	25	51
Total		5,760	5,524	257.4	0.5	0.57	0.21	1,666	11	27

Table 2.25: Detailed results of our BPC algorithm for the `H&W` instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.49	0.01	2	1	0
	30	160	160	0.0	0.0	0.26	0.00	2	1	0
	40	160	160	0.0	0.0	0.22	0.01	2	1	1
	50	160	160	0.0	0.0	0.19	0.01	3	2	0
	60	160	160	0.1	0.0	0.18	0.01	4	2	1
	70	160	160	0.1	0.0	0.12	0.01	3	2	1
	80	160	160	0.2	0.0	0.12	0.01	28	2	1
	90	160	160	0.2	0.0	0.11	0.01	6	3	1
	100	160	160	0.3	0.0	0.11	0.01	12	3	1
Subtotal		1,440	1,440	0.1	0.0	0.20	0.01	7	2	1
45	20	160	160	0.1	0.0	0.71	0.12	6	6	7
	30	160	160	0.7	0.0	0.50	0.17	55	6	12
	40	160	160	2.6	0.0	0.40	0.14	214	8	16
	50	160	159	24.5	0.1	0.37	0.15	1,097	8	19
	60	160	160	7.9	0.1	0.31	0.14	220	7	21
	70	160	160	24.9	0.1	0.29	0.15	779	8	24
	80	160	160	32.1	0.2	0.28	0.15	635	7	26
	90	160	159	78.9	0.2	0.28	0.14	1,488	7	27
	100	160	156	182.5	0.3	0.27	0.15	3,177	6	30
Subtotal		1,440	1,434	39.3	0.1	0.38	0.15	852	7	20
60	20	160	160	0.2	0.0	1.15	0.13	5	13	16
	30	160	160	1.9	0.1	0.90	0.29	43	12	25
	40	160	160	8.3	0.1	0.75	0.30	132	14	32
	50	160	160	45.2	0.2	0.63	0.30	1,021	12	35
	60	160	159	142.9	0.4	0.60	0.32	2,380	15	39
	70	160	154	323.7	0.6	0.55	0.31	4,587	13	40
	80	160	142	693.6	0.9	0.53	0.32	7,339	15	41
	90	160	143	765.3	1.2	0.48	0.30	6,327	13	47
	100	160	136	1,152.7	1.5	0.48	0.32	7,115	12	47
Subtotal		1,440	1,374	348.2	0.6	0.67	0.29	3,217	13	36
75	20	160	160	0.5	0.0	1.56	0.11	4	18	23
	30	160	160	4.0	0.2	1.31	0.26	22	19	40
	40	160	160	26.0	0.4	1.19	0.37	165	25	46
	50	160	159	157.5	0.8	1.10	0.42	1,008	28	49
	60	160	155	382.2	1.4	0.91	0.42	2,170	26	53
	70	160	143	892.4	2.4	0.86	0.46	4,194	27	56
	80	160	125	1,426.8	3.0	0.79	0.47	5,623	29	57
	90	160	101	2,036.9	4.7	0.87	0.55	5,710	27	60
	100	160	65	2,712.8	6.1	0.88	0.60	6,904	30	64
Subtotal		1,440	1,228	848.8	2.1	1.05	0.41	2,867	25	50
Total		5,760	5,476	309.1	0.7	0.58	0.21	1,736	12	27

Table 2.26: Detailed results of our BPC algorithm for the `H&W` instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.57	0.02	2	0	0
	30	160	160	0.0	0.0	0.34	0.02	2	2	0
	40	160	160	0.0	0.0	0.24	0.02	5	2	1
	50	160	160	0.0	0.0	0.23	0.01	2	3	1
	60	160	160	0.1	0.0	0.21	0.01	10	4	1
	70	160	160	0.2	0.0	0.17	0.01	24	4	1
	80	160	159	22.9	0.0	0.16	0.01	123	5	1
	90	160	159	22.8	0.0	0.14	0.01	35	5	1
	100	160	160	0.3	0.0	0.14	0.01	14	5	1
Subtotal		1,440	1,438	5.2	0.0	0.24	0.02	24	3	1
45	20	160	160	0.1	0.0	0.93	0.20	18	7	8
	30	160	160	1.0	0.0	0.60	0.22	77	7	12
	40	160	160	1.7	0.0	0.52	0.19	58	10	18
	50	160	160	17.2	0.1	0.49	0.23	933	12	21
	60	160	159	81.6	0.1	0.40	0.20	3,769	13	24
	70	160	158	105.2	0.1	0.35	0.19	3,345	14	24
	80	160	157	157.0	0.2	0.35	0.19	3,995	13	29
	90	160	157	176.0	0.2	0.31	0.17	3,176	17	29
	100	160	148	396.6	0.3	0.30	0.17	7,161	10	31
Subtotal		1,440	1,419	104.1	0.1	0.47	0.20	2,504	12	22
60	20	160	160	0.3	0.0	1.76	0.17	7	16	20
	30	160	160	2.2	0.1	1.22	0.30	68	17	29
	40	160	160	12.9	0.1	0.98	0.32	200	22	36
	50	160	158	70.5	0.2	0.76	0.29	1,958	20	40
	60	160	157	212.4	0.4	0.73	0.32	4,234	24	44
	70	160	153	411.9	0.5	0.66	0.30	6,443	28	48
	80	160	147	671.5	0.7	0.61	0.29	6,860	27	51
	90	160	137	1,018.8	1.0	0.56	0.31	9,098	23	57
	100	160	123	1,456.0	1.3	0.53	0.30	10,721	27	58
Subtotal		1,440	1,355	428.5	0.5	0.87	0.29	4,399	23	43
75	20	160	160	0.8	0.0	2.23	0.15	8	21	31
	30	160	160	5.9	0.2	1.73	0.29	34	23	47
	40	160	160	29.4	0.3	1.43	0.34	99	30	55
	50	160	158	153.5	0.6	1.34	0.35	980	35	61
	60	160	155	375.6	1.0	1.09	0.37	2,474	38	65
	70	160	139	898.9	1.7	0.96	0.37	5,198	36	74
	80	160	129	1,364.3	2.4	0.88	0.35	5,179	45	79
	90	160	111	1,854.2	3.0	0.83	0.36	6,282	44	82
	100	160	77	2,499.2	4.3	0.85	0.43	7,245	42	89
Subtotal		1,440	1,249	798.0	1.5	1.26	0.34	3,055	35	65
Total		5,760	5,461	333.9	0.5	0.71	0.21	2,495	18	32

Table 2.27: Detailed results of our BPC algorithm for the `H&W` instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
30	20	160	160	0.0	0.0	0.55	0.02	2	1	0
	30	160	160	0.0	0.0	0.28	0.00	2	1	0
	40	160	160	0.0	0.0	0.22	0.01	4	2	1
	50	160	160	0.0	0.0	0.20	0.00	2	1	0
	60	160	160	0.1	0.0	0.19	0.01	3	3	1
	70	160	160	0.1	0.0	0.14	0.01	9	2	1
	80	160	160	13.9	0.0	0.14	0.01	360	3	1
	90	160	160	0.2	0.0	0.13	0.01	5	3	1
	100	160	160	0.4	0.0	0.12	0.01	17	3	1
Subtotal		1,440	1,440	1.6	0.0	0.22	0.01	45	2	1
45	20	160	160	0.1	0.0	0.79	0.16	9	7	7
	30	160	160	0.5	0.0	0.51	0.17	25	8	13
	40	160	160	2.1	0.0	0.43	0.15	82	10	17
	50	160	160	5.7	0.1	0.41	0.18	204	13	20
	60	160	160	15.7	0.1	0.34	0.18	393	10	23
	70	160	159	56.4	0.2	0.31	0.16	1,487	11	24
	80	160	157	142.6	0.2	0.29	0.16	3,175	10	26
	90	160	157	140.7	0.3	0.28	0.15	2,121	12	27
	100	160	151	346.6	0.4	0.27	0.16	5,071	9	31
Subtotal		1,440	1,424	78.9	0.2	0.40	0.16	1,396	10	21
60	20	160	160	0.3	0.0	1.48	0.15	6	17	20
	30	160	160	2.0	0.1	1.03	0.26	36	17	28
	40	160	160	11.9	0.2	0.84	0.31	174	22	34
	50	160	159	64.7	0.3	0.69	0.30	1,164	19	38
	60	160	159	130.8	0.5	0.66	0.31	1,272	23	44
	70	160	152	403.6	0.7	0.60	0.31	4,359	26	44
	80	160	135	951.9	1.0	0.58	0.32	8,010	23	48
	90	160	141	996.3	1.4	0.51	0.31	5,375	22	51
	100	160	120	1,542.8	1.8	0.51	0.31	7,930	24	55
Subtotal		1440	1346	456.0	0.7	0.77	0.29	3,147	22	40
75	20	160	160	0.9	0.1	1.98	0.12	5	25	26
	30	160	160	5.4	0.2	1.56	0.25	21	28	41
	40	160	160	36.1	0.5	1.35	0.35	100	36	53
	50	160	160	162.9	0.9	1.22	0.38	798	36	54
	60	160	157	457.1	1.5	1.03	0.39	1,759	40	60
	70	160	144	878.2	2.3	0.91	0.40	3,246	38	64
	80	160	108	1,835.8	3.2	0.88	0.43	5,349	47	69
	90	160	98	2,099.8	4.5	0.88	0.48	5,230	43	71
	100	160	57	2,838.7	5.7	0.95	0.59	6,614	44	77
Subtotal		1,440	1,204	923.9	2.1	1.19	0.38	2,569	37	57
Total		5,760	5,414	365.1	0.7	0.65	0.21	1,789	18	30

Table 2.28: Detailed results of our BPC algorithm for the `H&W` instances and the optimal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	2	2,884.8	0.1	0.18	0.06	19,134	128	76
	300	10	1	3,253.7	0.3	0.13	0.03	9,538	147	77
	400	10	1	3,259.1	0.6	0.15	0.03	6,987	140	101
	500	10	0	3,600.0	1.3	0.16	0.03	3,615	152	123
	600	10	1	3,581.2	1.9	0.15	0.02	1,088	151	128
9	200	10	0	3,600.0	0.6	0.38	0.18	3,905	140	121
12	200	10	0	3,600.0	3.9	0.80	0.55	8,663	39	128
15	200	10	0	3,600.0	11.9	1.54	1.09	3,531	61	128
Total		80	5	3,422.4	2.6	0.44	0.25	7,058	120	110

Table 2.29: Detailed results of our BPC algorithm for the ZKS instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	6	1,648.2	0.1	0.24	0.07	10,474	92	63
	300	10	2	3,011.2	0.3	0.15	0.04	11,391	133	82
	400	10	0	3,600.0	0.7	0.15	0.05	4,407	155	120
	500	10	0	3,600.0	1.2	0.13	0.05	2,706	150	128
	600	10	0	3,600.0	1.9	0.13	0.04	1,080	138	128
9	200	10	1	3,344.2	0.5	0.41	0.15	4,235	92	123
12	200	10	0	3,600.0	1.6	0.60	0.34	3,552	35	128
15	200	10	1	3,510.9	5.8	0.82	0.49	1,862	20	128
Total		80	10	3,239.3	1.5	0.33	0.15	4,963	102	112

Table 2.30: Detailed results of our BPC algorithm for the ZKS instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	2	2,978.5	0.1	0.18	0.06	29,924	111	71
	300	10	1	3,250.6	0.4	0.16	0.03	19,045	147	76
	400	10	0	3,600.0	0.8	0.16	0.04	10,661	147	112
	500	10	0	3,600.0	1.6	0.16	0.04	3,335	160	127
	600	10	0	3,600.0	2.5	0.16	0.04	1,597	160	128
9	200	10	0	3,600.0	0.9	0.38	0.20	5,817	128	128
12	200	10	0	3,600.0	3.8	0.55	0.31	6,797	41	128
15	200	10	0	3,600.0	13.2	1.46	1.05	2,573	64	128
Total		80	3	3,478.6	2.9	0.40	0.22	9,969	120	112

Table 2.31: Detailed results of our BPC algorithm for the ZKS instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	4	2,172.0	0.2	0.19	0.06	20,578	110	80
	300	10	0	3,600.0	0.3	0.17	0.04	21,082	160	77
	400	10	1	3,265.4	0.9	0.16	0.04	5,841	147	113
	500	10	0	3,600.0	1.6	0.17	0.04	3,282	160	128
	600	10	0	3,600.0	2.5	0.15	0.04	1,402	160	128
9	200	10	0	3,600.0	0.8	0.35	0.18	7,359	128	128
12	200	10	0	3,600.0	4.2	0.55	0.30	8,070	37	128
15	200	10	0	3,600.0	14.2	1.47	1.06	2,533	76	128
Total		80	5	3,379.7	3.1	0.40	0.22	8,768	122	114

Table 2.32: Detailed results of our BPC algorithm for the ZKS instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	4	2,513.2	0.2	0.16	0.07	15,785	116	57
	300	10	1	3,243.9	0.4	0.15	0.04	17,634	120	71
	400	10	0	3,600.0	0.8	0.13	0.03	8,213	146	112
	500	10	1	3,384.2	1.4	0.12	0.03	3,880	158	120
	600	10	0	3,600.0	2.3	0.12	0.04	1,155	159	128
9	200	10	3	2,675.6	0.8	0.40	0.16	8,015	75	117
12	200	10	0	3,600.0	3.0	0.50	0.29	6,920	37	128
15	200	10	0	3,600.0	9.9	1.15	0.83	3,238	33	128
Total		80	9	3,277.1	2.4	0.34	0.19	8,105	106	108

Table 2.33: Detailed results of our BPC algorithm for the ZKS instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
6	200	10	3	2,524.3	0.2	0.20	0.08	17,991	105	58
	300	10	1	3,257.2	0.5	0.16	0.05	20,512	129	80
	400	10	0	3,600.0	0.9	0.13	0.04	11,327	146	118
	500	10	0	3,600.0	1.8	0.10	0.04	3,772	147	120
	600	10	0	3,600.0	2.7	0.12	0.04	1,568	155	128
9	200	10	2	2,991.5	1.1	0.34	0.12	4,529	108	116
12	200	10	0	3,600.0	4.9	0.52	0.33	5,249	37	128
15	200	10	0	3,600.0	18.5	1.19	0.89	2,203	39	128
Total		80	6	3,346.6	3.8	0.35	0.20	8,394	108	109

Table 2.34: Detailed results of our BPC algorithm for the ZKS instances and the optimal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	13.3	0.1	6.16	0.06	4	41	48
	30	10	10	330.1	0.9	3.35	0.34	144	60	85
	40	10	6	1,590.0	2.0	3.20	0.41	298	75	126
	50	10	7	1,791.3	4.8	4.40	0.84	827	64	128
	60	10	4	2,596.3	9.6	3.98	0.94	999	71	124
	70	10	2	3,090.8	15.7	2.15	0.75	1,266	72	128
	80	10	0	3,600.0	26.2	2.39	0.97	1,203	56	128
	90	10	1	3,262.8	36.7	2.35	0.68	1,126	44	128
	100	10	0	3,600.0	48.0	2.24	0.88	683	73	128
Subtotal		90	40	2,208.3	16.0	3.36	0.65	728	62	114
72	20	10	10	3.8	0.2	7.55	0.02	3	60	23
	30	10	10	368.8	1.3	8.97	0.48	25	108	126
	40	10	8	841.6	3.8	6.59	0.48	47	86	116
	50	10	4	2,728.1	13.6	5.92		224	100	93
	60	10	6	2,141.7	24.9	4.58	0.63	234	83	128
	70	10	2	2,903.7	42.8	3.00		275	89	120
	80	10	0	3,600.0	60.1	3.61	0.91	386	91	128
	90	10	1	3,356.4	84.4	3.03	0.84	226	114	128
	100	10	0	3,600.0	104.4	1.99	0.51	344	69	128
Subtotal		90	41	2,171.6	37.3	5.03		196	89	110
Total		180	81	2,189.9	26.6	4.19		462	75	112

Table 2.35: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	1.2	0.2	4.34	0.00	2	29	5
	30	10	10	99.2	1.5	3.15	0.06	4	83	86
	40	10	10	139.1	4.6	2.67	0.24	13	66	102
	50	10	9	535.5	11.2	3.33	0.36	29	83	121
	60	10	5	2,358.5	23.2	3.85	0.66	121	130	124
	70	10	7	1,685.2	32.5	1.93	0.57	133	102	124
	80	10	1	3,371.3	62.0	2.64	1.25	457	114	127
	90	10	2	3,063.7	76.6	2.33	0.82	730	73	127
	100	10	0	3,600.0	95.2	2.58	1.25	428	114	128
Subtotal		90	54	1,650.4	34.1	2.98	0.58	213	88	105
72	20	10	10	4.4	0.4	7.30	0.00	2	81	14
	30	10	10	74.6	4.2	7.75	0.06	3	121	72
	40	10	10	980.8	17.2	6.01	0.31	32	117	97
	50	10	6	1,760.4	62.7	4.45	0.56	60	133	128
	60	10	3	2,863.8	131.0	4.86		135	127	128
	70	10	2	3,227.4	169.6	3.20	1.14	142	130	128
	80	10	1	3,462.0	321.3	4.21	1.56	166	131	128
	90	10	0	3,600.0	496.7	4.50	2.29	89	146	128
	100	10	0	3,600.0	545.9	4.63	2.87	115	132	128
Subtotal		90	42	2,174.8	194.3	5.21		83	124	106
Total		180	96	1,912.6	114.2	4.10		148	106	105

Table 2.36: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	2.3	0.3	3.96	0.00	2	35	24
	30	10	10	38.6	2.5	2.02	0.07	5	62	60
	40	10	10	81.7	8.8	2.36	0.30	18	52	102
	50	10	10	850.1	26.8	3.32	0.58	65	72	114
	60	10	6	1,760.1	49.7	3.47	0.84	207	91	119
	70	10	3	2,823.2	87.0	2.15	0.90	472	80	117
	80	10	2	3,054.8	146.5	2.42	1.27	472	68	123
	90	10	1	3,357.8	196.5	2.53	1.15	658	58	120
	100	10	0	3,600.0	227.6	2.69	1.63	405	101	123
Subtotal		90	52	1,729.8	82.8	2.77	0.75	256	69	100
72	20	10	10	5.0	0.6	6.02	0.00	2	63	15
	30	10	10	226.4	8.5	6.34	0.16	7	110	87
	40	10	9	509.8	34.0	4.67	0.25	23	93	104
	50	10	8	1,525.1	169.4	4.00	0.50	42	123	127
	60	10	5	2,326.9	332.7	4.67	1.39	90	120	124
	70	10	2	3,187.4	539.2	3.39		84	115	117
	80	10	1	3,529.1	1,050.3	4.95	2.53	87	114	128
	90	10	0	3,600.0	1,395.7	4.73	2.74	47	143	128
	100	10	0	3,600.0	1,656.8			71	114	114
Subtotal		90	45	2,056.6	576.4			50	110	105
Total		180	97	1,893.2	329.6			153	90	103

Table 2.37: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	1.7	0.4	4.08	0.00	2	23	23
	30	10	10	60.6	4.4	2.15	0.12	4	49	73
	40	10	10	169.9	16.1	2.55	0.37	19	50	113
	50	10	10	557.8	49.8	3.43	0.66	98	62	110
	60	10	6	2,339.4	87.4	3.62	0.94	360	98	112
	70	10	3	3,145.2	148.0	2.35		457	88	110
	80	10	0	3,600.0	319.2	2.57	1.38	368	64	117
	90	10	0	3,600.0	314.9	2.75	1.42	694	54	112
	100	10	0	3,600.0	392.3	2.57	1.48	361	95	120
Subtotal		90	49	1,897.2	148.0	2.90		263	65	99
72	20	10	10	7.0	0.8	6.04	0.00	2	69	11
	30	10	10	249.0	18.7	6.48	0.11	9	100	66
	40	10	10	717.2	60.4	4.59	0.27	16	90	91
	50	10	7	1,798.4	300.1	4.38		63	113	116
	60	10	2	3,272.1	723.2	4.79		148	105	111
	70	10	0	3,600.0	1,109.0	3.92		103	106	103
	80	10	0	3,600.0	2,192.1			44	95	93
	90	10	0	3,600.0	2,614.2			8	118	69
	100	10	0	3,600.0	2,880.7			18	74	62
Subtotal		90	39	2,271.5	1,099.9			46	97	80
Total		180	88	2,084.4	624.0			154	81	90

Table 2.38: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	4.5	0.2	5.77	0.02	2	43	33
	30	10	10	215.0	1.6	3.58	0.27	9	75	122
	40	10	10	294.5	3.9	3.09	0.33	25	72	122
	50	10	8	1,387.5	9.7	4.00	0.63	152	82	122
	60	10	5	2,389.4	17.3	3.99	0.60	360	112	125
	70	10	5	2,645.1	27.3	2.33	0.75	560	109	128
	80	10	2	3,107.1	42.3	2.33	0.82	623	98	126
	90	10	1	3,317.4	52.3	2.51	0.95	1205	60	125
	100	10	0	3,600.0	66.7	2.33	0.95	563	110	128
Subtotal		90	51	1,884.5	24.6	3.33	0.59	389	84	114
72	20	10	10	9.8	0.4	7.60	0.00	2	71	34
	30	10	10	195.5	4.2	8.45	0.18	7	109	71
	40	10	9	901.6	10.9	6.52	0.39	48	123	104
	50	10	6	1,692.2	39.7	4.51	0.56	144	106	128
	60	10	2	3,074.3	78.3	4.80	0.78	220	116	128
	70	10	3	2,825.8	118.0	3.14	0.92	116	119	128
	80	10	0	3,600.0	190.8	3.95	0.92	221	119	128
	90	10	0	3,600.0	234.9	4.60	2.22	120	146	128
	100	10	1	3,316.4	280.2	3.08	1.37	149	114	128
Subtotal		90	41	2,135.1	106.4	5.18	0.82	114	114	109
Total		180	92	2,009.8	65.5	4.25	0.70	252	99	112

Table 2.39: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
60	20	10	10	7.7	0.5	5.44	0.00	2	44	24
	30	10	10	444.3	4.0	3.35	0.29	10	78	116
	40	10	9	745.6	12.1	3.14	0.57	80	68	110
	50	10	8	1,112.2	32.3	3.67	0.53	68	82	108
	60	10	6	2,403.0	61.0	3.85	0.67	244	110	121
	70	10	2	3,220.8	79.6	2.44		522	103	123
	80	10	0	3,600.0	155.5	2.58	1.15	464	95	124
	90	10	1	3,402.8	154.1	2.44	0.98	721	66	127
	100	10	0	3,600.0	215.7	2.44	1.09	311	120	128
Subtotal		90	46	2,059.6	79.4	3.26		269	85	109
72	20	10	10	29.8	1.1	7.90	0.00	2	95	42
	30	10	10	1,010.6	13.8	8.16	0.34	42	127	84
	40	10	10	487.2	38.7	5.81	0.26	15	116	127
	50	10	8	1,417.6	154.9	4.31	0.50	54	112	126
	60	10	6	2,216.9	306.0	4.59		60	117	115
	70	10	0	3,600.0	377.1	3.67		148	122	94
	80	10	0	3,600.0	734.3	4.50		124	125	116
	90	10	0	3,600.0	883.5	5.06		59	146	112
	100	10	0	3,600.0	1,070.4	5.25		91	112	115
Subtotal		90	44	2,173.6	397.7	5.47		66	119	104
Total		180	90	2,116.6	238.6	4.37		168	102	106

Table 2.40: Detailed results of our BPC algorithm for the `M&Ö-ext` instances and the optimal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	5	2,259.2	0.7	0.57	0.30	19,047	20	71
	150	10	4	2,564.5	1.1	0.44	0.26	14,542	23	81
	175	10	3	2,733.7	2.0	0.36	0.24	12,180	18	83
	200	10	0	3,600.0	2.5	0.44	0.30	6,009	38	92
	225	10	0	3,600.0	5.6	0.45	0.32	5,569	36	113
	250	10	0	3,600.0	5.0	0.48	0.36	2,742	60	123
Subtotal		60	12	3,059.6	2.8	0.46	0.30	10,015	32	94
36	125	10	0	3,600.0	5.9	1.12	0.57	6,002	51	119
	150	10	0	3,600.0	9.6	0.95	0.51	5,023	28	126
	175	10	0	3,600.0	15.0	1.29	0.98	3,349	27	128
	200	10	0	3,600.0	21.1	1.21	0.89	1,936	51	128
	225	10	0	3,600.0	35.0	1.07	0.83	1,303	38	128
	250	10	0	3,600.0	43.2	1.22	0.94	994	41	128
Subtotal		60	0	3,600.0	21.6	1.15	0.79	3,101	39	126
48	125	10	1	3,445.3	32.9	1.72	0.82	1,579	69	128
	150	10	0	3,600.0	47.5	1.48	1.03	1,625	50	128
	175	10	0	3,600.0	74.4	1.25	0.98	1,873	39	128
	200	10	0	3,600.0	106.0	1.44	1.03	510	84	128
	225	10	0	3,600.0	178.6	1.84	1.43	550	44	128
	250	10	0	3,600.0	222.9	1.50	1.29	627	25	128
Subtotal		60	1	3,574.2	110.4	1.54	1.10	1,127	52	128
60	125	10	0	3,600.0	99.2	2.04	0.88	483	87	128
	150	10	0	3,600.0	168.2	2.23	1.21	462	70	128
	175	10	0	3,600.0	263.6	2.59		220	86	111
	200	10	0	3,600.0	385.6	3.13		281	84	102
	225	10	0	3,600.0	971.7			105	97	115
	250	10	0	3,600.0	808.0	3.48	2.87	96	87	128
Subtotal		60	0	3,600.0	449.4			275	85	119
72	125	10	0	3,600.0	290.4	2.90	1.53	194	85	128
	150	10	0	3,600.0	816.3			134	76	102
	175	10	0	3,600.0	2,194.3			75	25	64
	200	10	0	3,600.0	2,100.3			68	39	64
	225	10	0	3,600.0	3,130.3			8	38	38
	250	10	0	3,600.0	2,840.8			33	22	64
Subtotal		60	0	3,600.0	1,895.4			85	48	77
Total		300	13	3,486.8	495.9			2,921	51	109

Table 2.41: Detailed results of our BPC algorithm for the `W&G-g` instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	8	1,423.7	0.7	0.48	0.27	5,781	23	61
	150	10	7	1,845.8	1.1	0.39	0.25	5,286	16	62
	175	10	1	3,508.7	1.5	0.43	0.29	9,694	40	68
	200	10	0	3,600.0	2.1	0.44	0.32	5,054	48	76
	225	10	0	3,600.0	4.2	0.46	0.34	3,945	38	84
	250	10	0	3,600.0	4.9	0.48	0.38	3,494	35	83
Subtotal		60	16	2,929.7	2.4	0.45	0.31	5,542	33	72
36	125	10	1	3,489.9	5.8	1.14	0.56	2,254	81	111
	150	10	0	3,600.0	9.5	1.34	0.84	2,239	61	112
	175	10	0	3,600.0	14.9	1.27	0.92	2,074	56	115
	200	10	0	3,600.0	21.2	1.48	1.12	884	71	123
	225	10	0	3,600.0	33.3	1.43	1.17	765	52	127
	250	10	0	3,600.0	39.8	1.49	1.26	660	56	128
Subtotal		60	1	3,581.6	20.7	1.36	0.98	1,479	63	119
48	125	10	0	3,600.0	39.9	1.74	0.94	1,015	98	124
	150	10	0	3,600.0	58.9	1.46	1.00	1,114	69	124
	175	10	0	3,600.0	103.1	1.82	1.48	596	74	126
	200	10	0	3,600.0	130.7	1.99	1.58	284	117	128
	225	10	0	3,600.0	222.6	2.91	2.44	236	93	128
	250	10	0	3,600.0	260.9	1.94	1.66	168	96	128
Subtotal		60	0	3,600.0	136.0	1.98	1.52	569	91	126
60	125	10	0	3,600.0	242.5	3.07	2.00	275	131	128
	150	10	0	3,600.0	461.2	2.82	1.94	367	83	128
	175	10	0	3,600.0	1,203.4			106	103	102
	200	10	0	3,600.0	1,405.8			136	70	102
	225	10	0	3,600.0	2,814.4			14	43	38
	250	10	0	3,600.0	3,046.7			10	35	38
Subtotal		60	0	3,600.0	1,529.0			151	77	90
72	125	10	0	3,600.0	1,938.5			68	92	90
	150	10	0	3,600.0	2,974.9			18	56	38
	175	10	0	3,600.0	3,408.6			10	10	26
	200	10	0	3,600.0	3,600.0			0	0	0
	225	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		60	0	3,600.0	3,187.0			16	26	26
Total		300	17	3,462.3	975.0			1,552	58	87

Table 2.42: Detailed results of our BPC algorithm for the `W&G-g` instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	6	1,694.5	0.8	0.50	0.32	9,695	16	54
	150	10	7	2,458.0	1.2	0.37	0.23	11,454	22	59
	175	10	5	3,076.1	2.0	0.34	0.24	7,997	25	66
	200	10	1	3,484.8	2.6	0.36	0.27	7,086	23	71
	225	10	0	3,600.0	5.3	0.43	0.34	6,644	11	80
	250	10	0	3,600.0	5.6	0.42	0.33	3,475	21	95
Subtotal		60	19	2,985.6	2.9	0.40	0.29	7,725	19	71
36	125	10	4	2,985.8	8.4	0.92	0.52	3,840	29	96
	150	10	0	3,600.0	12.9	1.12	0.70	3,319	29	101
	175	10	0	3,600.0	19.7	1.25	0.94	2,121	33	110
	200	10	0	3,600.0	28.8	1.10	0.81	1,444	23	117
	225	10	0	3,600.0	47.0	1.23	1.02	1,284	20	124
	250	10	0	3,600.0	52.7	1.41	1.18	826	26	123
Subtotal		60	4	3,497.6	28.3	1.17	0.86	2,139	27	112
48	125	10	1	3,561.2	67.0	1.47	0.86	1,670	36	113
	150	10	0	3,600.0	95.2	1.98	1.54	1,115	39	122
	175	10	0	3,600.0	150.9	2.03	1.79	825	36	125
	200	10	0	3,600.0	238.1	1.97	1.61	360	68	128
	225	10	0	3,600.0	348.8	2.46	2.12	390	31	128
	250	10	0	3,600.0	425.2	2.18	1.91	248	54	128
Subtotal		60	1	3,593.5	220.9	2.01	1.64	768	44	124
60	125	10	0	3,600.0	593.4	2.42	1.55	287	97	126
	150	10	0	3,600.0	941.7	3.10	2.37	280	50	125
	175	10	0	3,600.0	1,993.7			76	83	102
	200	10	0	3,600.0	2,026.9	7.16	6.69	77	73	127
	225	10	0	3,600.0	3,393.6			5	12	20
	250	10	0	3,600.0	3,452.8			2	16	26
Subtotal		60	0	3,600.0	2,067.0			121	55	88
72	125	10	0	3,600.0	3,055.6			32	38	49
	150	10	0	3,600.0	3,591.8			0	2	13
	175	10	0	3,600.0	3,600.0			0	0	0
	200	10	0	3,600.0	3,600.0			0	0	0
	225	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		60	0	3,600.0	3,507.9			5	7	10
Total		300	24	3,455.3	1,165.4			2,152	30	81

Table 2.43: Detailed results of our BPC algorithm for the `W&G-g` instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	7	1,557.1	0.9	0.45	0.25	7,845	15	49
	150	10	7	2,043.3	1.3	0.42	0.27	7,625	14	62
	175	10	3	3,169.5	2.1	0.42	0.32	7,396	22	66
	200	10	0	3,600.0	2.9	0.36	0.27	8,043	17	73
	225	10	0	3,600.0	4.8	0.44	0.33	4,892	14	86
	250	10	0	3,600.0	6.6	0.43	0.37	4,093	10	87
Subtotal		60	17	2,928.3	3.1	0.42	0.30	6,649	16	70
36	125	10	1	3,367.5	10.3	1.01	0.60	4,941	26	88
	150	10	0	3,600.0	15.3	1.15	0.76	3,505	19	94
	175	10	0	3,600.0	24.9	1.38	1.08	1,838	26	109
	200	10	0	3,600.0	33.1	1.26	0.99	1,360	24	118
	225	10	0	3,600.0	51.2	1.32	1.13	1,139	24	121
	250	10	0	3,600.0	61.8	1.41	1.20	729	20	125
Subtotal		60	1	3,561.3	32.8	1.26	0.96	2,252	23	109
48	125	10	0	3,600.0	96.6	2.05	1.42	1,181	48	113
	150	10	0	3,600.0	142.8	1.93	1.48	964	34	122
	175	10	0	3,600.0	219.2	1.78	1.52	542	37	127
	200	10	0	3,600.0	318.2	1.91	1.55	327	58	128
	225	10	0	3,600.0	662.2	2.52	2.17	314	30	126
	250	10	0	3,600.0	578.8	2.31	2.06	191	53	128
Subtotal		60	0	3,600.0	336.3	2.08	1.70	587	43	124
60	125	10	0	3,600.0	985.3	3.15	2.33	227	79	122
	150	10	0	3,600.0	1,520.5	3.48	2.74	166	52	128
	175	10	0	3,600.0	2,421.3			57	91	103
	200	10	0	3,600.0	3,035.0			16	65	70
	225	10	0	3,600.0	3,555.9			0	7	13
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		60	0	3,600.0	2,519.7			78	49	72
72	125	10	0	3,600.0	3,542.2			3	4	26
	150	10	0	3,600.0	3,600.0			0	0	0
	175	10	0	3,600.0	3,600.0			0	0	0
	200	10	0	3,600.0	3,600.0			0	0	0
	225	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		60	0	3,600.0	3,590.4			0	1	4
Total		300	18	3,457.9	1,296.4			1,913	26	76

Table 2.44: Detailed results of our BPC algorithm for the `W&G-g` instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	7	1,974.4	0.8	0.56	0.30	8,877	32	57
	150	10	5	2,624.6	1.2	0.41	0.25	9,788	26	76
	175	10	3	3,164.7	1.8	0.37	0.25	7,062	26	77
	200	10	0	3,600.0	2.8	0.41	0.29	5,734	34	85
	225	10	0	3,600.0	5.1	0.46	0.34	4,413	29	93
	250	10	1	3,576.3	6.0	0.46	0.36	2,895	53	112
Subtotal		60	16	3,090.0	2.9	0.45	0.30	6,462	33	84
36	125	10	1	3,518.5	6.9	1.15	0.59	3,545	55	117
	150	10	0	3,600.0	10.4	1.20	0.67	2,953	48	117
	175	10	0	3,600.0	15.6	1.38	1.01	2,443	28	128
	200	10	0	3,600.0	22.2	1.39	1.01	962	57	126
	225	10	0	3,600.0	35.6	1.11	0.87	1,012	33	128
	250	10	0	3,600.0	42.0	1.31	1.07	784	36	128
Subtotal		60	1	3,586.4	22.1	1.26	0.87	1,950	43	124
48	125	10	0	3,600.0	30.3	1.72	0.93	1,351	71	128
	150	10	0	3,600.0	50.4	1.36	0.92	1,278	52	126
	175	10	0	3,600.0	76.1	1.83	1.48	771	46	128
	200	10	0	3,600.0	112.7	1.73	1.34	386	80	128
	225	10	0	3,600.0	181.2	2.55	2.07	326	75	128
	250	10	0	3,600.0	205.2	2.18	1.91	303	60	128
Subtotal		60	0	3,600.0	109.3	1.90	1.44	736	64	128
60	125	10	0	3,600.0	142.0	2.41	1.33	446	108	128
	150	10	0	3,600.0	543.7			442	46	115
	175	10	0	3,600.0	992.0			185	77	102
	200	10	0	3,600.0	1,137.3			151	57	102
	225	10	0	3,600.0	1,332.3			56	98	102
	250	10	0	3,600.0	1,913.5			46	56	77
Subtotal		60	0	3,600.0	1,010.1			221	74	105
72	125	10	0	3,600.0	849.0			125	113	113
	150	10	0	3,600.0	1,495.6			60	110	90
	175	10	0	3,600.0	2,158.2			49	55	77
	200	10	0	3,600.0	2,425.2			27	66	77
	225	10	0	3,600.0	3,514.7			0	2	13
	250	10	0	3,600.0	3,299.9			4	18	51
Subtotal		60	0	3,600.0	2,290.4			44	61	70
Total		300	17	3,495.3	687.0			1,882	55	102

Table 2.45: Detailed results of our BPC algorithm for the `W&G-g` instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	125	10	7	1,730.9	1.2	0.48	0.24	8,323	19	59
	150	10	5	2,410.4	2.0	0.38	0.24	8,146	13	69
	175	10	4	2,671.1	2.9	0.43	0.30	5,237	25	66
	200	10	0	3,600.0	3.9	0.46	0.34	5,483	26	84
	225	10	0	3,600.0	7.0	0.44	0.32	4,440	20	89
	250	10	0	3,600.0	8.1	0.52	0.44	3,172	23	98
Subtotal		60	16	2,935.4	4.2	0.45	0.31	5,800	21	78
36	125	10	0	3,600.0	12.5	1.21	0.65	2,387	65	112
	150	10	0	3,600.0	19.2	1.22	0.72	2,138	44	113
	175	10	0	3,600.0	28.2	1.41	1.03	1,552	31	123
	200	10	0	3,600.0	39.3	1.40	1.05	861	45	124
	225	10	0	3,600.0	59.7	1.26	1.03	871	27	128
	250	10	0	3,600.0	70.2	1.59	1.36	647	32	128
Subtotal		60	0	3,600.0	38.2	1.35	0.97	1,409	41	122
48	125	10	0	3,600.0	79.8	1.86	1.13	915	68	127
	150	10	0	3,600.0	110.1	1.55	1.12	818	54	128
	175	10	0	3,600.0	184.6	2.12	1.77	528	50	128
	200	10	0	3,600.0	263.8	2.07	1.65	266	81	128
	225	10	0	3,600.0	385.8	3.15	2.69	268	60	128
	250	10	0	3,600.0	473.1	2.19	1.92	175	70	128
Subtotal		60	0	3,600.0	249.5	2.16	1.71	495	64	128
60	125	10	0	3,600.0	1,051.2			210	90	102
	150	10	0	3,600.0	729.0	2.69	1.78	248	62	128
	175	10	0	3,600.0	1,806.6			84	68	90
	200	10	0	3,600.0	2,131.9			59	64	90
	225	10	0	3,600.0	2,829.9			10	66	60
	250	10	0	3,600.0	3,209.1			2	49	31
Subtotal		60	0	3,600.0	1,959.6			102	66	83
72	125	10	0	3,600.0	2,676.0			37	52	39
	150	10	0	3,600.0	3,094.1			19	34	38
	175	10	0	3,600.0	3,530.3			2	8	26
	200	10	0	3,600.0	3,600.0			0	0	0
	225	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		60	0	3,600.0	3,350.1			10	16	17
Total		300	16	3,467.1	1,120.3			1,563	42	86

Table 2.46: Detailed results of our BPC algorithm for the `W&G-g` instances and the optimal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	77.8	0.3	0.36	0.21	688	0	61
	150	10	7	1,447.1	1.0	0.37	0.18	4,133	28	74
	200	10	6	2,896.3	3.1	0.21	0.12	10,114	2	102
	250	10	1	3,391.7	5.4	0.20	0.10	2,057	84	122
Subtotal		40	24	1,953.2	2.5	0.28	0.15	4,248	28	90
36	100	10	7	2,235.3	2.3	1.12	0.57	4,202	31	118
	150	10	2	3,006.6	7.0	0.62	0.47	6,439	0	128
	200	10	0	3,600.0	17.0	0.87	0.55	3,571	19	128
	250	10	0	3,600.0	35.5	0.72	0.59	2,368	10	128
Subtotal		40	9	3,110.5	15.4	0.83	0.54	4,145	15	125
48	100	10	2	3320.0	9.7	1.59	0.70	1,478	33	128
	150	10	0	3,600.0	30.7	1.41	0.89	607	65	128
	200	10	0	3,600.0	84.6	1.05	0.90	957	0	128
	250	10	0	3,600.0	172.2	1.56	1.21	487	13	128
Subtotal		40	2	3,530.0	74.3	1.40	0.93	882	28	128
60	100	10	1	3,542.6	27.9	0.94	0.62	1,000	1	128
	150	10	0	3,600.0	117.5	1.07	0.83	597	0	128
	200	10	0	3,600.0	278.4	1.65	1.44	365	0	128
	250	10	0	3,600.0	599.2	4.97	4.84	241	0	128
Subtotal		40	1	3,585.6	255.7	2.16	1.93	551	0	128
72	100	10	0	3,600.0	58.4	2.25	0.90	317	38	128
	150	10	0	3,600.0	253.6	2.94	2.07	160	44	128
	200	10	0	3,600.0	670.7	4.37	3.71	56	78	128
	250	10	0	3,600.0	1,438.7	6.65	6.29	21	109	128
Subtotal		40	0	3,600.0	605.3	4.05	3.24	139	67	128
Total		200	36	3,155.9	190.7	1.75	1.36	1,993	28	120

Table 2.47: Detailed results of our BPC algorithm for the `W&G-u` instances and the traversal strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	34.0	0.3	0.30	0.19	118	0	42
	150	10	10	777.8	0.8	0.42	0.18	856	43	57
	200	10	9	1,399.0	2.2	0.24	0.15	3,727	1	71
	250	10	1	3,290.1	3.5	0.38	0.24	1,186	50	88
Subtotal		40	30	1,375.2	1.7	0.33	0.19	1,472	24	65
36	100	10	6	2,793.7	2.3	1.06	0.52	2,544	61	97
	150	10	1	3,336.6	7.6	0.63	0.50	4,493	0	98
	200	10	0	3,600.0	15.9	1.09	0.71	1,478	18	120
	250	10	0	3,600.0	30.9	1.11	0.88	513	54	126
Subtotal		40	7	3,332.6	14.2	0.97	0.65	2,257	33	110
48	100	10	3	3,251.1	11.9	1.79	0.77	786	85	124
	150	10	0	3,600.0	39.1	1.72	1.29	429	92	126
	200	10	0	3,600.0	94.9	1.42	1.26	626	0	126
	250	10	0	3,600.0	168.6	1.95	1.65	339	14	128
Subtotal		40	3	3,512.8	78.6	1.72	1.24	545	48	126
60	100	10	0	3,600.0	60.1	1.47	1.16	597	0	125
	150	10	0	3,600.0	193.2	1.66	1.47	368	0	126
	200	10	0	3,600.0	494.8	2.20	2.02	212	0	128
	250	10	0	3,600.0	873.5	2.40	2.26	127	0	128
Subtotal		40	0	3,600.0	405.4	1.93	1.73	326	0	127
72	100	10	0	3,600.0	287.2	3.53	1.71	123	101	128
	150	10	0	3,600.0	984.1	5.73	4.66	55	144	128
	200	10	0	3,600.0	2,408.6	9.80		3	151	103
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		40	0	3,600.0	1,820.0			45	99	90
Total		200	40	3,084.1	464.0			929	41	104

Table 2.48: Detailed results of our BPC algorithm for the `W&G-u` instances and the return strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	37.8	0.4	0.34	0.19	176	0	42
	150	10	10	577.4	1.0	0.44	0.18	1,381	14	58
	200	10	8	1,908.6	3.0	0.27	0.17	5,843	0	68
	250	10	0	3,600.0	4.6	0.38	0.25	2,710	22	90
Subtotal		40	28	1,531.0	2.3	0.36	0.20	2,528	9	65
36	100	10	8	1,737.4	3.4	0.96	0.50	2,719	38	88
	150	10	0	3,600.0	11.1	0.90	0.76	4,311	0	92
	200	10	0	3,600.0	23.9	1.30	0.99	1,713	9	105
	250	10	0	3,600.0	44.3	1.31	1.11	778	19	123
Subtotal		40	8	3,134.4	20.7	1.12	0.84	2,380	17	102
48	100	10	1	3,538.3	23.4	1.75	0.94	1,370	50	120
	150	10	0	3,600.0	79.4	2.06	1.60	618	54	122
	200	10	0	3,600.0	172.2	1.95	1.80	424	0	122
	250	10	0	3,600.0	353.7	2.53	2.27	247	13	128
Subtotal		40	1	3,584.6	157.2	2.07	1.65	665	29	123
60	100	10	0	3,600.0	152.7	1.79	1.50	315	0	114
	150	10	0	3,600.0	578.8	2.45	2.26	146	0	121
	200	10	0	3,600.0	1,348.5	3.17	3.01	86	0	125
	250	10	0	3,600.0	2,522.8	7.98	7.83	28	0	128
Subtotal		40	0	3,600.0	1,150.7	3.85	3.65	144	0	122
72	100	10	0	3,600.0	994.5	4.57	3.14	83	72	126
	150	10	0	3,600.0	3,600.0			0	0	0
	200	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		40	0	3,600.0	2,948.6			21	18	32
Total		200	37	3,090.0	855.9			1,148	15	89

Table 2.49: Detailed results of our BPC algorithm for the `W&G-u` instances and the midpoint strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	24.5	0.4	0.28	0.14	77	0	48
	150	10	9	679.2	1.2	0.36	0.14	1,834	20	60
	200	10	5	2,548.2	3.3	0.30	0.21	6,710	0	74
	250	10	0	3,600.0	5.9	0.35	0.22	2,674	24	91
Subtotal		40	24	1,713.0	2.7	0.32	0.18	2,824	11	68
36	100	10	9	1,809.8	4.7	0.93	0.48	2,302	35	80
	150	10	1	3,380.1	14.7	0.77	0.63	2,994	0	99
	200	10	0	3,600.0	31.9	1.32	0.98	1,357	12	111
	250	10	0	3,600.0	57.9	1.28	1.09	643	24	123
Subtotal		40	10	3,097.5	27.3	1.08	0.80	1,824	18	103
48	100	10	0	3,600.0	40.6	2.02	1.21	1,070	56	118
	150	10	0	3,600.0	132.3	2.07	1.61	465	52	126
	200	10	0	3,600.0	282.9	2.00	1.85	323	0	123
	250	10	0	3,600.0	514.8	2.66	2.43	183	9	126
Subtotal		40	0	3,600.0	242.7	2.19	1.78	510	29	123
60	100	10	0	3,600.0	321.2	2.13	1.89	182	0	105
	150	10	0	3,600.0	1,181.9	2.93	2.77	99	0	112
	200	10	0	3,600.0	2,612.8	19.71	19.59	18	0	122
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		40	0	3,600.0	1,929.0			75	0	85
72	100	10	0	3,600.0	2,386.0	17.64		7	65	82
	150	10	0	3,600.0	3,600.0			0	0	0
	200	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		40	0	3,600.0	3,296.5			2	16	21
Total		200	34	3,122.1	1,099.6			1,047	15	80

Table 2.50: Detailed results of our BPC algorithm for the `W&G-u` instances and the largest gap strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	28.5	0.4	0.27	0.15	119	0	45
	150	10	8	1,538.9	1.0	0.46	0.20	5,158	26	56
	200	10	6	2,394.1	3.0	0.26	0.17	7,607	0	75
	250	10	0	3,600.0	5.0	0.39	0.25	3,206	25	100
Subtotal		40	24	1,890.4	2.3	0.35	0.19	4,022	13	69
36	100	10	5	2,249.7	2.6	1.05	0.52	2,736	40	112
	150	10	1	3,415.4	8.0	0.67	0.53	4,927	0	110
	200	10	0	3,600.0	18.0	1.14	0.76	1,532	13	124
	250	10	0	3,600.0	35.1	1.21	0.98	592	37	128
Subtotal		40	6	3,216.3	15.9	1.02	0.70	2,447	23	119
48	100	10	1	3,577.4	10.2	2.01	0.98	1,581	60	127
	150	10	0	3,600.0	34.1	1.67	1.14	538	78	128
	200	10	0	3,600.0	75.5	1.18	1.04	850	0	128
	250	10	0	3,600.0	141.2	1.62	1.28	384	17	128
Subtotal		40	1	3,594.3	65.3	1.62	1.11	838	38	128
60	100	10	1	3,423.7	39.9	1.39	1.10	776	0	126
	150	10	0	3,600.0	128.2	1.72	1.49	472	0	128
	200	10	0	3,600.0	290.3	1.72	1.55	297	0	128
	250	10	0	3,600.0	555.7	2.08	1.95	175	0	128
Subtotal		40	1	3,555.9	253.5	1.73	1.52	430	0	127
72	100	10	0	3,600.0	123.4	3.44	1.64	153	70	128
	150	10	0	3,600.0	452.7	3.94	2.67	66	128	128
	200	10	0	3,600.0	1,132.2	5.50	4.78	34	144	128
	250	10	0	3,600.0	2,168.1	6.48	6.15	16	142	128
Subtotal		40	0	3,600.0	969.1	4.84	3.81	67	121	128
Total		200	32	3,171.4	261.2	1.91	1.47	1,561	39	114

Table 2.51: Detailed results of our BPC algorithm for the `W&G-u` instances and the combined strategy

Q	n	Inst	Opt	t[s]	t^{LP}	Gp	Gp^{RF}	Nds	CC	SRC
24	100	10	10	54.0	0.7	0.29	0.17	183	0	54
	150	10	10	1,118.0	1.9	0.41	0.17	2,564	28	54
	200	10	3	3,150.8	4.7	0.28	0.19	7,668	0	73
	250	10	0	3,600.0	8.0	0.43	0.29	2,241	34	85
Subtotal		40	23	1,980.7	3.8	0.35	0.20	3,164	16	66
36	100	10	7	2,174.5	5.8	1.02	0.49	1,810	50	102
	150	10	0	3,600.0	19.2	0.83	0.67	2,292	0	112
	200	10	0	3,600.0	37.2	1.14	0.79	1,008	17	127
	250	10	0	3,600.0	67.9	1.24	1.04	505	36	128
Subtotal		40	7	3,243.6	32.5	1.06	0.75	1,404	26	117
48	100	10	0	3,600.0	32.4	1.83	0.84	820	76	124
	150	10	0	3,600.0	108.5	1.84	1.36	373	85	128
	200	10	0	3,600.0	223.9	1.55	1.40	343	0	128
	250	10	0	3,600.0	406.4	2.35	2.06	202	14	128
Subtotal		40	0	3,600.0	192.8	1.89	1.42	434	44	127
60	100	10	0	3,600.0	161.4	1.57	1.28	261	0	121
	150	10	0	3,600.0	532.4	2.04	1.85	153	0	128
	200	10	0	3,600.0	1,126.5	2.08	1.92	91	0	128
	250	10	0	3,600.0	2,035.2	4.32	4.18	43	0	128
Subtotal		40	0	3,600.0	963.9	2.50	2.31	137	0	126
72	100	10	0	3,600.0	648.4	3.72	1.88	67	92	128
	150	10	0	3,600.0	2,437.4	15.16		10	136	115
	200	10	0	3,600.0	3,600.0			0	0	0
	250	10	0	3,600.0	3,600.0			0	0	0
Subtotal		40	0	3,600.0	2,571.5			19	57	61
Total		200	30	3,204.9	752.9			1,032	29	100

Table 2.52: Detailed results of our BPC algorithm for the `W&G-u` instances and the optimal strategy

Detailed Results of BPC-based Heuristics

Tables 2.53–2.58 provide aggregated results per capacity Q for the proposed BPC-based heuristics on the six benchmark sets and all considered routing strategies. They compare variants of the set-covering heuristic (*SC*) and the depth-first heuristic (*BPC-DF*) with hard time limits of two, three, and five minutes (*-2*, *-3*, *-5*). The average gap with respect to the best-known solution (*Gp*) and the average computation time in seconds ($t[s]$) are reported. In cases where no average could be computed for a given group, e.g., because no lower bound was available for one of the comprised instances, the corresponding cell is left blank.

		SC heuristic						BPC-DF heuristic					
		SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	24	0.09	5.6	0.09	5.6	0.09	5.6	0.02	40.4	0.02	54.8	0.01	78.1
	36	0.50	52.6	0.40	68.2	0.33	92.9	0.22	80.4	0.17	115.6	0.11	181.3
	48	1.63	64.5	1.58	87.8	1.07	129.7	0.67	89.5	0.40	131.8	0.29	214.7
Subtotal		0.74	40.9	0.69	53.9	0.50	76.1	0.30	70.1	0.20	100.8	0.14	158.1
Return	24	0.08	6.1	0.08	6.7	0.08	7.0	0.04	30.3	0.03	38.8	0.01	55.0
	36	0.44	49.9	0.33	65.8	0.31	88.0	0.20	64.2	0.16	91.4	0.08	141.9
	48	2.02	64.7	1.50	89.7	1.41	136.1	1.19	75.2	0.70	108.5	0.36	168.6
Subtotal		0.85	40.2	0.64	54.1	0.60	77.0	0.48	56.6	0.30	79.6	0.15	121.8
Midpoint	24	0.06	4.7	0.06	4.7	0.06	4.7	0.01	24.1	0.01	31.7	0.01	46.5
	36	0.61	45.6	0.56	62.4	0.42	87.1	0.17	60.0	0.15	83.3	0.09	127.5
	48	2.30	70.4	1.89	100.9	1.61	152.0	1.02	77.8	0.62	112.4	0.32	176.4
Subtotal		0.99	40.2	0.84	56.0	0.70	81.3	0.40	54.0	0.26	75.8	0.14	116.8
L. gap	24	0.06	4.1	0.06	4.1	0.06	4.1	0.02	30.6	0.01	38.8	0.01	51.1
	36	0.62	49.9	0.54	65.8	0.44	96.1	0.16	65.8	0.14	93.8	0.09	146.6
	48	2.47	72.4	2.01	100.5	1.75	153.7	1.19	82.0	0.75	117.6	0.45	187.5
Subtotal		1.05	42.1	0.87	56.8	0.75	84.6	0.45	59.4	0.30	83.4	0.18	128.4
Combined	24	0.10	7.4	0.10	7.5	0.10	7.5	0.02	30.7	0.02	39.4	0.01	56.8
	36	0.68	53.9	0.59	71.4	0.50	99.4	0.20	70.9	0.14	100.3	0.11	155.1
	48	2.52	68.7	2.27	97.6	1.91	150.6	0.66	79.7	0.47	115.5	0.28	186.2
Subtotal		1.10	43.4	0.99	58.8	0.84	85.8	0.29	60.5	0.21	85.1	0.13	132.7
Optimal	24	0.09	4.9	0.09	4.9	0.09	4.9	0.02	31.5	0.02	42.0	0.01	56.9
	36	0.75	53.7	0.61	72.7	0.52	104.4	0.27	70.3	0.21	99.1	0.16	152.4
	48	2.31	73.0	2.17	101.3	1.82	151.5	1.73	88.3	1.12	127.9	0.67	204.8
Subtotal		1.05	43.9	0.96	59.6	0.81	86.9	0.67	63.4	0.45	89.7	0.28	138.0
Total		0.96	41.8	0.83	56.5	0.70	82.0	0.43	60.7	0.29	85.7	0.17	132.6

Table 2.53: Comparison of the BPC-based heuristics on the `M&Ö` instances

		SC heuristic						BPC-DF heuristic					
		SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	30	0.00	0.1	0.00	0.1	0.00	0.1	0.00	4.3	0.00	6.0	0.00	9.4
	45	0.06	1.7	0.06	1.7	0.06	1.7	0.02	38.0	0.02	52.3	0.02	78.0
	60	0.10	14.5	0.10	15.8	0.10	16.9	0.11	72.8	0.09	104.9	0.08	166.0
	75	0.23	33.1	0.20	40.2	0.18	49.3	0.25	90.7	0.21	133.4	0.18	216.2
Subtotal		0.10	12.3	0.09	14.4	0.08	17.0	0.10	51.5	0.08	74.2	0.07	117.4
Return	30	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.1
	45	0.06	1.4	0.06	1.4	0.06	1.4	0.01	14.3	0.01	18.5	0.00	25.8
	60	0.15	21.0	0.14	24.0	0.13	27.6	0.08	45.8	0.06	61.7	0.05	89.4
	75	0.51	48.0	0.44	64.1	0.36	90.4	0.21	64.1	0.16	90.7	0.12	139.5
Subtotal		0.18	17.6	0.16	22.4	0.14	29.9	0.07	31.1	0.06	42.7	0.05	63.7
Midpoint	30	0.00	0.0	0.00	0.0	0.00	0.0	0.00	0.0	0.00	0.0	0.00	0.0
	45	0.04	1.0	0.04	1.0	0.04	1.0	0.00	7.9	0.00	9.5	0.00	11.8
	60	0.15	15.6	0.14	17.6	0.13	20.0	0.05	41.2	0.04	55.5	0.03	79.9
	75	0.52	44.5	0.46	58.5	0.39	81.3	0.18	63.3	0.14	89.6	0.11	136.7
Subtotal		0.18	15.3	0.16	19.3	0.14	25.6	0.06	28.1	0.05	38.6	0.04	57.1
L. gap	30	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.2	0.00	0.3	0.00	0.5
	45	0.04	1.1	0.04	1.1	0.04	1.1	0.00	9.6	0.00	11.5	0.00	14.6
	60	0.14	18.5	0.13	21.0	0.12	23.1	0.06	44.6	0.05	61.4	0.04	90.7
	75	0.57	47.1	0.50	62.7	0.43	88.5	0.25	67.0	0.20	95.1	0.15	147.0
Subtotal		0.19	16.7	0.17	21.2	0.15	28.2	0.08	30.4	0.06	42.1	0.05	63.2
Combined	30	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.6	0.00	0.8	0.00	1.2
	45	0.05	1.6	0.05	1.6	0.05	1.7	0.01	17.5	0.01	22.3	0.01	29.9
	60	0.14	22.8	0.13	26.1	0.12	29.4	0.10	50.6	0.08	69.6	0.06	103.8
	75	0.42	43.6	0.35	56.8	0.28	77.0	0.22	65.8	0.18	93.7	0.13	145.6
Subtotal		0.15	17.0	0.13	21.2	0.11	27.0	0.08	33.6	0.07	46.6	0.05	70.1
Optimal	30	0.00	0.1	0.00	0.1	0.00	0.1	0.00	0.2	0.00	0.2	0.00	0.3
	45	0.05	1.5	0.05	1.5	0.05	1.5	0.01	15.1	0.01	18.2	0.01	24.1
	60	0.17	25.4	0.15	30.1	0.13	35.2	0.09	52.3	0.07	72.2	0.06	107.9
	75	0.58	51.3	0.50	68.4	0.42	95.8	0.29	70.0	0.23	99.6	0.16	154.5
Subtotal		0.20	19.6	0.17	25.0	0.15	33.1	0.10	34.4	0.08	47.6	0.06	71.7
Total		0.17	16.4	0.15	20.6	0.13	26.8	0.08	34.8	0.07	48.6	0.05	73.9

Table 2.54: Comparison of the BPC-based heuristics on the `H&W` instances

			SC heuristic						BPC-DF heuristic					
			SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	n	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	6	200	0.05	2.3	0.05	2.3	0.05	2.3	0.08	108.1	0.07	162.1	0.07	270.1
	6	300	0.03	8.0	0.03	8.0	0.03	8.0	0.11	108.1	0.06	162.1	0.05	270.1
	6	400	0.03	18.6	0.03	18.8	0.03	18.9	0.23	109.8	0.17	163.8	0.15	271.7
	6	500	0.02	40.2	0.02	41.8	0.02	41.8	0.48	120.0	0.44	180.0	0.38	300.0
	6	600	0.02	53.7	0.02	59.5	0.02	60.0	1.04	120.0	0.62	180.1	0.38	300.0
	9	200	0.16	28.3	0.16	28.2	0.16	28.3	1.48	120.0	1.25	180.0	1.06	300.0
	12	200	1.49	119.6	1.14	179.4	1.07	293.8	1.73	120.0	1.46	180.0	1.09	300.0
	15	200	3.41	119.6	2.98	179.4	2.69	298.9	3.68	120.0	2.20	180.0	1.67	300.0
	Subtotal		0.65	48.8	0.55	64.7	0.51	94.0	1.10	115.8	0.78	173.5	0.61	289.0
Return	6	200	0.02	2.2	0.02	2.2	0.02	2.2	0.16	108.4	0.15	162.4	0.09	270.4
	6	300	0.02	4.6	0.02	4.6	0.02	4.6	0.16	120.0	0.15	180.0	0.13	300.0
	6	400	0.04	37.9	0.03	39.8	0.03	39.6	0.25	120.0	0.21	180.0	0.19	300.0
	6	500	0.04	52.5	0.03	59.6	0.03	71.6	0.50	120.0	0.34	180.0	0.29	300.0
	6	600	0.04	71.0	0.03	90.3	0.03	120.8	0.51	120.0	0.49	180.0	0.37	300.0
	9	200	0.12	30.3	0.12	30.3	0.12	30.2	0.67	120.0	0.51	180.0	0.50	300.0
	12	200	0.70	116.8	0.40	170.5	0.35	255.7	1.19	120.0	1.16	180.0	0.81	300.0
	15	200	1.52	119.6	1.36	179.3	1.04	298.8	1.60	120.0	1.21	180.0	1.07	300.0
	Subtotal		0.31	54.4	0.25	72.1	0.21	102.9	0.63	118.6	0.53	177.8	0.43	296.3
Midpoint	6	200	0.06	2.5	0.06	2.5	0.06	2.5	0.08	120.0	0.08	180.0	0.07	300.0
	6	300	0.03	8.0	0.03	7.9	0.03	8.0	0.14	108.5	0.12	162.5	0.10	270.5
	6	400	0.03	17.5	0.03	17.6	0.03	17.2	0.40	120.0	0.24	180.0	0.21	300.0
	6	500	0.04	52.8	0.04	64.7	0.04	79.2	0.42	120.0	0.33	180.0	0.29	300.0
	6	600	0.05	79.2	0.03	96.2	0.03	120.0	0.65	120.0	0.55	180.0	0.47	300.0
	9	200	0.18	33.2	0.18	33.1	0.18	33.1	0.89	120.0	0.79	180.0	0.55	300.0
	12	200	1.18	119.6	0.80	174.7	0.41	270.3	1.16	120.0	0.85	180.0	0.63	300.0
	15	200	2.88	119.7	2.56	179.5	2.28	299.1	3.11	120.0	2.09	180.0	1.70	300.0
	Subtotal		0.56	54.1	0.47	72.0	0.38	103.7	0.86	118.6	0.63	177.8	0.50	296.3
L. gap	6	200	0.05	2.8	0.05	2.8	0.05	2.8	0.10	108.6	0.09	162.6	0.08	270.6
	6	300	0.04	7.3	0.04	7.3	0.04	7.3	0.17	120.0	0.13	180.0	0.10	300.0
	6	400	0.03	18.9	0.03	19.0	0.03	18.8	0.25	120.0	0.24	180.0	0.22	300.0
	6	500	0.03	41.9	0.03	42.8	0.03	42.8	0.56	120.0	0.48	180.0	0.36	300.0
	6	600	0.03	67.4	0.03	80.0	0.03	101.6	1.44	120.0	0.58	180.0	0.52	300.0
	9	200	0.15	47.1	0.15	53.3	0.15	65.1	1.03	120.0	0.81	180.0	0.64	300.0
	12	200	0.85	112.9	0.49	166.7	0.51	251.5	1.14	120.0	1.07	180.0	0.85	300.0
	15	200	2.54	119.6	2.30	179.3	1.86	298.8	2.71	120.0	2.42	180.0	1.69	300.0
	Subtotal		0.47	52.2	0.39	68.9	0.34	98.6	0.93	118.6	0.73	177.8	0.56	296.3

Continued on the next page.

Table 2.55: Comparison of the BPC-based heuristics on the ZKS instances

			SC heuristic						BPC-DF heuristic					
			SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	n	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Combined	6	200	0.04	2.7	0.04	2.7	0.04	2.7	0.13	106.0	0.12	153.9	0.11	250.7
	6	300	0.03	7.3	0.03	7.2	0.03	7.2	0.28	120.0	0.20	180.0	0.19	300.0
	6	400	0.03	16.7	0.03	16.7	0.03	16.7	0.23	120.0	0.22	180.0	0.13	300.0
	6	500	0.03	36.6	0.03	41.2	0.03	40.9	0.36	120.0	0.28	180.0	0.22	300.0
	6	600	0.03	82.0	0.03	94.0	0.03	109.6	0.64	120.0	0.53	180.0	0.36	300.0
	9	200	0.13	36.2	0.13	40.6	0.13	40.5	0.65	120.0	0.55	180.0	0.50	300.0
	12	200	0.58	112.0	0.32	156.6	0.24	240.3	1.04	120.0	0.84	180.0	0.72	300.0
	15	200	2.23	119.7	2.20	179.4	2.10	299.0	2.55	120.0	2.01	180.0	1.59	300.0
	Subtotal		0.39	51.7	0.35	67.3	0.33	94.6	0.73	118.3	0.59	176.7	0.48	293.9
Optimal	6	200	0.05	2.8	0.05	2.8	0.05	2.8	0.13	111.3	0.13	165.1	0.10	273.1
	6	300	0.05	12.1	0.05	12.0	0.05	12.2	0.16	120.0	0.15	180.0	0.12	300.0
	6	400	0.03	17.0	0.03	17.0	0.03	16.9	0.27	120.0	0.21	180.0	0.15	300.0
	6	500	0.03	31.3	0.03	31.2	0.03	31.2	0.32	120.0	0.29	180.0	0.24	300.0
	6	600	0.03	78.0	0.03	95.9	0.03	111.3	1.20	120.0	0.47	180.0	0.33	300.0
	9	200	0.09	33.0	0.08	33.2	0.08	33.2	0.87	120.0	0.61	180.0	0.59	300.0
	12	200	0.88	119.3	0.63	177.0	0.41	273.5	1.24	120.0	1.03	180.0	1.02	300.0
	15	200	2.71	119.6	2.51	179.4	2.27	299.0	2.13	120.0	1.37	180.0	1.18	300.0
	Subtotal		0.48	51.6	0.43	68.6	0.37	97.5	0.79	118.9	0.53	178.2	0.46	296.6
Total			0.48	52.1	0.41	68.9	0.36	98.6	0.84	118.1	0.63	177.0	0.51	294.7

Table 2.55: Comparison of the BPC-based heuristics on the ZKS instances (cont.)

		SC heuristic						BPC-DF heuristic					
		SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	60	4.52	84.6	3.99	120.3	3.55	185.3	1.71	94.5	0.98	139.8	0.50	226.4
	72	7.51	87.1	6.18	122.9	5.18	190.8	6.56	100.6	4.05	147.1	1.57	234.8
Subtotal		6.02	85.8	5.08	121.6	4.36	188.1	4.14	97.5	2.51	143.4	1.04	230.6
Return	60	5.93	79.4	4.77	108.8	3.88	161.4	4.61	81.7	2.84	117.3	1.06	181.6
	72	10.23	89.8	9.61	128.8	8.87	201.6	10.63	96.7	9.68	140.7	7.57	226.7
Subtotal		8.08	84.6	7.19	118.8	6.38	181.5	7.62	89.2	6.26	129.0	4.31	204.2
Midpoint	60	8.43	81.3	6.97	115.8	4.75	174.6	8.36	86.2	5.56	123.9	2.95	198.5
	72		95.4		135.5		211.1		98.8		142.0		227.0
Subtotal			88.3		125.7		192.8		92.5		132.9		212.8
L. gap	60	10.20	89.3	9.48	126.4	7.14	197.7	9.51	91.9	8.73	133.8	6.33	212.9
	72		99.0		143.5		229.3		102.1		148.7		238.8
Subtotal			94.2		135.0		213.5		97.0		141.2		225.9
Combined	60	6.51	87.2	5.56	122.9	4.40	186.7	2.96	88.3	1.93	127.3	0.88	202.9
	72	8.79	86.7	8.64	121.4	8.00	190.4	8.55	97.8	7.41	141.6	4.92	227.9
Subtotal		7.65	86.9	7.10	122.1	6.20	188.6	5.75	93.1	4.67	134.4	2.90	215.4
Optimal	60	9.04	89.6	7.74	129.3	5.36	201.1	7.62	94.3	6.24	137.2	3.41	219.9
	72	10.74	112.9	10.18	154.1	9.29	229.8	10.62	106.1	10.05	154.8	8.68	247.5
Subtotal		9.89	101.3	8.96	141.7	7.33	215.4	9.12	100.2	8.15	146.0	6.04	233.7
Total			90.2		127.5		196.7		94.9		137.8		220.4

Table 2.56: Comparison of the BPC-based heuristics on the `M&Ö-ext` instances

		SC heuristic						BPC-DF heuristic					
		SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	24	1.08	111.0	0.81	158.8	0.56	240.3	2.10	120.0	0.96	179.4	0.47	297.0
	36	3.69	119.8	3.10	179.5	2.94	299.1	7.75	120.0	4.44	180.0	1.88	300.0
	48	15.73	119.9	13.63	179.9	7.75	299.6	16.66	120.0	14.36	180.0	9.33	300.1
	60		120.0		180.0		299.9		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			118.1		175.6		287.8		120.0		179.9		299.4
Return	24	0.90	104.2	0.71	148.9	0.52	229.0	0.67	120.0	0.42	179.0	0.34	297.0
	36	4.76	119.8	3.22	179.7	2.93	299.3	7.33	120.1	5.25	180.0	2.38	300.0
	48	15.68	120.0	13.78	179.9	10.90	299.7	16.81	120.0	14.99	180.0	11.09	300.1
	60		120.0		180.0		300.0		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.1
Subtotal			116.8		173.7		285.6		120.0		179.8		299.5
Midpoint	24	0.94	107.5	0.80	151.1	0.67	226.4	0.52	118.7	0.38	177.4	0.33	293.3
	36	5.41	119.8	3.18	179.6	3.08	299.2	7.33	120.1	4.35	180.0	1.82	300.0
	48	18.96	120.0	15.90	179.9	12.85	299.7	20.38	120.0	17.61	180.0	12.83	300.1
	60		120.0		180.0		300.0		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			117.5		174.1		285.1		119.8		179.5		298.7
L. gap	24	1.02	104.2	0.93	148.6	0.70	228.5	0.59	119.2	0.48	178.2	0.31	294.4
	36	5.93	119.8	3.32	179.2	3.20	296.8	7.69	120.0	5.85	180.0	2.30	300.0
	48	18.53	120.0	17.36	180.0	13.38	299.8	19.23	120.0	17.90	180.0	14.79	300.0
	60		120.0		180.0		300.0		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			116.8		173.6		285.0		119.9		179.7		298.9
Combined	24	1.08	112.8	0.89	163.1	0.67	255.0	1.21	120.0	0.73	179.4	0.46	297.5
	36	4.01	119.8	3.45	179.5	3.17	299.1	8.09	120.0	4.05	180.0	2.19	300.0
	48	15.03	119.9	12.41	179.8	9.49	299.6	16.58	120.0	14.19	180.1	10.53	300.1
	60		120.0		180.0		299.9		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.1		300.0
Subtotal			118.5		176.5		290.7		120.0		179.9		299.5
Optimal	24	0.98	111.4	0.90	157.8	0.75	244.2	1.00	120.0	0.58	180.0	0.39	299.5
	36	6.84	119.9	3.68	179.7	3.26	299.3	9.76	120.2	6.72	180.0	2.50	300.0
	48	18.01	120.0	16.48	180.0	13.06	299.8	18.01	120.0	17.33	180.0	14.53	300.1
	60		120.0		180.0		300.0		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.4
Subtotal			118.2		175.5		288.7		120.1		180.0		300.0
Total			117.7		174.8		287.1		120.0		179.8		299.3

Table 2.57: Comparison of the BPC-based heuristics on the `W&G-g` instances

		SC heuristic						BPC-DF heuristic					
		SC-2		SC-3		SC-5		BPC-DF-2		BPC-DF-3		BPC-DF-5	
Routing	Q	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]
Traversal	24	0.15	61.7	0.09	75.7	0.06	90.0	1.04	103.5	0.62	151.9	0.13	247.0
	36	3.63	116.7	3.35	172.4	3.05	278.7	6.00	120.0	1.22	180.0	0.90	299.1
	48	12.79	119.9	11.84	179.7	7.47	299.4	13.26	120.0	8.54	180.1	5.59	300.0
	60	21.50	119.9	19.53	179.8	18.34	299.6	20.10	120.0	16.44	180.0	14.64	300.0
	72	23.12	120.0	22.99	179.9	21.46	299.8	24.17	120.0	22.00	180.0	20.51	300.0
Subtotal		12.24	107.6	11.56	157.5	10.08	253.5	12.91	116.7	9.76	174.4	8.35	289.2
Return	24	0.31	76.0	0.27	102.8	0.14	141.4	0.23	94.6	0.18	138.0	0.12	221.5
	36	4.18	119.6	2.96	179.3	2.60	297.4	5.32	120.0	4.16	180.0	1.51	300.0
	48	12.79	119.9	10.01	179.8	6.47	299.4	13.31	120.0	10.75	180.2	6.65	300.0
	60	20.09	120.0	19.84	179.9	17.12	299.8	18.03	120.0	17.84	180.0	14.28	300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			111.1		164.4		267.6		114.9		171.6		284.3
Midpoint	24	0.38	78.5	0.32	105.7	0.20	157.2	0.21	97.7	0.17	140.0	0.12	217.8
	36	7.26	117.2	3.71	169.8	3.59	272.8	5.96	120.2	4.36	180.0	1.06	298.3
	48	19.30	120.0	17.18	179.9	11.55	299.7	19.09	120.0	17.18	180.0	13.59	300.0
	60	26.44	120.0	25.48	180.0	23.20	299.9	26.44	120.0	26.44	180.0	21.16	300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			111.1		163.1		265.9		115.6		172.0		283.2
L. gap	24	0.39	74.0	0.32	103.8	0.22	158.7	0.22	89.2	0.17	131.1	0.14	208.2
	36	7.28	119.8	3.66	178.4	3.49	292.0	6.73	120.2	5.38	180.0	0.99	300.1
	48	19.34	120.0	18.52	179.9	15.60	299.8	19.71	120.0	18.33	180.0	14.73	300.0
	60		120.0		180.0		300.0		120.0		180.0		300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			110.8		164.4		270.1		113.9		170.2		281.7
Combined	24	0.55	80.5	0.42	114.7	0.26	170.5	0.34	93.9	0.18	136.1	0.12	219.9
	36	3.72	119.8	3.51	178.9	3.17	294.2	6.07	120.0	4.50	180.0	1.51	299.4
	48	11.54	119.9	11.25	179.7	7.57	299.4	14.04	120.0	8.01	180.2	5.47	300.0
	60	22.41	119.9	19.55	179.9	18.67	299.7	20.06	120.0	17.12	180.0	14.28	300.0
	72	25.59	120.0	24.53	180.0	23.39	299.9	25.59	120.0	25.59	180.0	23.87	300.0
Subtotal		12.76	112.0	11.85	166.6	10.61	272.7	13.22	114.8	11.08	171.3	9.05	283.9
Optimal	24	0.61	79.2	0.40	113.5	0.29	172.3	0.90	101.8	0.24	143.9	0.14	225.6
	36	6.94	119.0	4.44	175.9	3.29	285.6	8.40	120.1	5.55	180.0	3.09	300.0
	48	17.79	120.0	17.17	179.9	12.48	299.8	18.97	120.0	17.39	180.0	13.30	300.0
	60	22.10	120.0	21.51	180.0	20.17	299.9	22.10	120.0	22.10	180.0	17.78	300.0
	72		120.0		180.0		300.0		120.0		180.0		300.0
Subtotal			111.6		165.9		271.5		116.4		172.8		285.1
Total			110.7		163.6		266.9		115.4		172.1		284.6

Table 2.58: Comparison of the BPC-based heuristics on the `W&G-u` instances

Comparison of Routing Strategies

Tables 2.59–2.62 provide a comparison of the different routing strategies with respect to the total traveled distances for the benchmark sets M&Ö, H&W, and ZKS aggregated by capacity Q and number of orders n. The columns report the percentage increase in the total traveled distances for the respective routing strategy compared to the optimal strategy. For the comparison, we use the BKS for each instance and routing strategy.

Q	n	Traversal	Return	Midpoint	Largest gap	Combined
24	20	11.8%	33.3%	9.5%	5.6%	4.3%
	30	10.3%	32.3%	9.9%	5.4%	3.3%
	40	11.0%	31.8%	10.3%	6.0%	4.2%
	50	10.8%	34.0%	10.1%	6.3%	4.0%
	60	9.3%	32.2%	10.0%	5.8%	3.3%
	70	10.3%	32.9%	9.3%	5.4%	3.9%
	80	10.1%	32.9%	9.8%	5.9%	3.6%
	90	10.0%	32.7%	9.7%	5.6%	3.3%
	100	9.8%	33.1%	10.1%	5.8%	3.2%
Subtotal		10.4%	32.8%	9.9%	5.8%	3.7%
36	20	8.7%	35.7%	14.1%	7.7%	3.0%
	30	7.1%	35.2%	14.2%	8.4%	2.3%
	40	7.6%	34.2%	13.7%	8.4%	2.9%
	50	7.6%	35.7%	13.7%	8.6%	2.8%
	60	6.6%	34.2%	13.6%	8.2%	2.2%
	70	7.0%	34.8%	13.0%	7.9%	2.6%
	80	6.6%	34.9%	13.2%	8.3%	2.5%
	90	6.6%	34.4%	13.0%	8.0%	2.1%
	100	6.3%	34.7%	13.1%	8.1%	2.2%
Subtotal		7.1%	34.9%	13.5%	8.2%	2.5%
48	20	7.0%	37.0%	18.2%	10.8%	2.6%
	30	5.4%	36.3%	18.0%	11.1%	1.9%
	40	5.6%	35.5%	17.3%	10.9%	1.8%
	50	5.2%	37.2%	16.8%	10.8%	1.9%
	60	4.9%	36.3%	17.8%	11.3%	1.4%
	70	5.2%	36.8%	16.5%	10.5%	1.8%
	80	5.4%	37.2%	16.8%	10.8%	2.0%
	90	4.4%	36.1%	15.9%	10.5%	1.6%
	100	4.1%	36.3%	16.7%	10.4%	1.6%
Subtotal		5.3%	36.5%	17.1%	10.8%	1.8%
Total		7.6%	34.7%	13.4%	8.2%	2.7%

Table 2.59: Percentage increase in total traveled distances compared to the optimal strategy for the M&Ö instances

Q	n	Traversal	Return	Midpoint	Largest gap	Combined
30	20	18.8%	52.9%	15.0%	8.5%	7.6%
	30	17.7%	53.1%	15.2%	8.8%	7.6%
	40	17.1%	52.0%	15.7%	9.1%	7.2%
	50	17.6%	52.6%	15.4%	8.8%	7.1%
	60	17.3%	52.5%	15.5%	8.9%	7.1%
	70	17.4%	52.8%	15.3%	8.9%	7.1%
	80	16.9%	52.5%	15.6%	9.0%	7.0%
	90	17.0%	52.9%	15.5%	8.9%	7.1%
	100	17.2%	52.9%	15.5%	9.0%	7.0%
Subtotal		17.4%	52.7%	15.4%	8.9%	7.2%
45	20	10.4%	53.6%	20.7%	12.5%	4.5%
	30	10.3%	54.1%	20.8%	12.8%	4.3%
	40	10.2%	54.1%	20.7%	12.5%	4.3%
	50	10.0%	54.0%	20.5%	12.7%	4.2%
	60	9.9%	54.2%	20.6%	12.6%	4.2%
	70	10.0%	54.0%	20.5%	12.5%	4.3%
	80	9.8%	53.7%	20.3%	12.4%	4.1%
	90	9.5%	53.6%	20.5%	12.5%	4.0%
	100	9.7%	53.5%	20.1%	12.4%	4.1%
Subtotal		10.0%	53.9%	20.5%	12.5%	4.2%
60	20	7.8%	56.0%	24.7%	15.9%	3.3%
	30	8.0%	56.1%	25.2%	16.1%	3.1%
	40	7.7%	55.4%	25.1%	16.2%	2.8%
	50	7.6%	55.8%	24.6%	16.0%	3.0%
	60	7.5%	55.2%	24.4%	15.7%	3.0%
	70	7.5%	55.2%	24.5%	15.8%	2.9%
	80	7.3%	54.9%	24.6%	15.8%	2.8%
	90	7.1%	55.2%	24.2%	15.7%	2.8%
	100	7.2%	55.2%	24.2%	15.6%	2.8%
Subtotal		7.5%	55.4%	24.6%	15.9%	2.9%
75	20	6.2%	56.8%	28.9%	19.1%	2.6%
	30	6.9%	57.4%	28.8%	19.3%	2.5%
	40	6.6%	57.2%	28.7%	19.2%	2.3%
	50	6.6%	57.0%	27.9%	18.9%	2.3%
	60	6.4%	57.0%	28.1%	19.0%	2.2%
	70	6.3%	57.1%	27.9%	19.0%	2.2%
	80	6.1%	56.8%	27.8%	18.9%	2.0%
	90	5.8%	56.5%	27.6%	18.8%	1.9%
	100	5.7%	56.7%	27.5%	18.7%	1.9%
Subtotal		6.3%	56.9%	28.1%	19.0%	2.2%
Total		10.2%	54.7%	22.1%	14.0%	4.1%

Table 2.60: Percentage increase in total traveled distances compared to the optimal strategy for the `H&W` UDD instances

Q	n	Traversal	Return	Midpoint	Largest gap	Combined
30	20	19.4%	51.9%	9.6%	5.4%	8.9%
	30	19.7%	52.7%	9.2%	5.3%	8.9%
	40	18.7%	52.4%	9.2%	5.3%	8.6%
	50	19.3%	52.1%	9.3%	5.3%	8.7%
	60	19.6%	52.1%	9.4%	5.4%	8.9%
	70	19.1%	52.2%	9.6%	5.4%	8.6%
	80	19.0%	52.4%	9.5%	5.4%	8.7%
	90	18.7%	51.7%	9.5%	5.4%	8.4%
	100	18.9%	52.2%	9.4%	5.3%	8.4%
Subtotal		19.2%	52.2%	9.4%	5.4%	8.7%
45	20	12.7%	52.3%	12.7%	7.4%	5.8%
	30	12.8%	53.2%	12.9%	7.4%	6.1%
	40	11.9%	52.2%	12.3%	7.2%	5.8%
	50	12.1%	52.4%	12.5%	7.3%	5.7%
	60	12.0%	52.6%	12.7%	7.3%	5.7%
	70	11.7%	52.2%	12.5%	7.2%	5.4%
	80	11.8%	52.3%	12.2%	7.0%	5.6%
	90	11.6%	52.1%	12.0%	7.0%	5.3%
	100	11.8%	52.3%	11.9%	7.1%	5.5%
Subtotal		12.0%	52.4%	12.4%	7.2%	5.7%
60	20	9.8%	52.7%	15.7%	9.3%	4.3%
	30	9.7%	53.6%	15.5%	9.4%	4.3%
	40	9.2%	53.9%	15.5%	9.4%	4.1%
	50	9.0%	53.2%	15.3%	9.3%	4.2%
	60	8.9%	52.8%	15.1%	9.1%	4.2%
	70	8.7%	53.3%	15.1%	9.3%	4.1%
	80	8.6%	52.9%	15.1%	9.2%	4.0%
	90	8.7%	53.1%	15.2%	9.3%	4.0%
	100	8.8%	53.2%	14.8%	9.1%	4.0%
Subtotal		9.1%	53.2%	15.2%	9.3%	4.1%
75	20	8.6%	54.6%	19.0%	11.6%	3.7%
	30	8.0%	53.7%	18.7%	11.7%	3.3%
	40	7.7%	54.4%	17.9%	11.4%	3.1%
	50	7.5%	54.2%	18.2%	11.4%	3.2%
	60	6.9%	53.8%	17.9%	11.4%	2.9%
	70	7.1%	54.3%	17.8%	11.3%	3.0%
	80	7.1%	54.2%	17.7%	11.2%	3.1%
	90	7.2%	54.0%	17.4%	11.1%	3.1%
	100	7.1%	53.7%	17.0%	11.0%	3.1%
Subtotal		7.5%	54.1%	18.0%	11.3%	3.2%
Total		11.8%	53.0%	13.7%	8.3%	5.4%

Table 2.61: Percentage increase in total traveled distances compared to the optimal strategy for the `H&W` CBD instances

Q	n	Traversal	Return	Midpoint	Largest gap	Combined
6	200	21.1%	30.7%	6.5%	6.2%	5.5%
	300	21.0%	29.2%	6.8%	6.5%	5.3%
	400	20.8%	28.9%	6.6%	6.3%	5.3%
	500	21.2%	28.5%	6.6%	6.3%	5.3%
	600	20.9%	28.0%	6.6%	6.2%	5.3%
9	200	17.4%	34.1%	6.1%	5.4%	5.2%
12	200	15.3%	38.8%	5.6%	4.5%	4.9%
15	200	14.2%	40.4%	6.1%	4.6%	4.5%
Total		19.0%	32.2%	6.4%	5.7%	5.2%

Table 2.62: Percentage increase in total traveled distances compared to the optimal strategy for the `ZKS` instances

Bibliography

Achterberg, T. (2007). *Constraint Integer Programming.* Ph.D. thesis, Fakultät II – Mathematik und Naturwissenschaften, Technische Universität Berlin, Berlin, Germany.

Bahçeci, U. and Öncan, T. (2022). An evaluation of several combinations of routing and storage location assignment policies for the order batching problem. *International Journal of Production Research*, **60**(19), 5892–5911.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, **12**(4), 568–581.

Henn, S. and Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, **222**(3), 484–494.

Muter, İ. and Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, **47**(7), 728–738.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. *Decision Sciences Institute Proceedings*, **3**, 1614–1616.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

Roodbergen, K. J. (2001). *Layout and routing methods for warehouses.* Ph.D. thesis, Erasmus University Rotterdam, Rotterdam, the Netherlands.

Scholz, A. and Wäscher, G. (2017). Order batching and picker routing in manual order picking systems: The benefits of integrated routing. *Central European Journal of Operations Research*, **25**(2), 491–520.

Žulj, I., Kramer, S., and Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, **264**(2), 653–664.

Chapter 3

Solving the Multi-Block Order Batching Problem with Branch-Price-and-Cut

Julia Wahlen

Abstract

In the realm of warehouse optimization, the *order batching problem* (OBP) involves partitioning customer orders into capacity-constrained batches, such that the total distance traveled in the warehouse for picking all batches is minimized. This paper addresses the OBP in rectangular warehouses consisting of two or more blocks with parallel aisles, expanding upon a *branch-price-and-cut* (BPC) approach previously applied to the OBP in single-block warehouses, as discussed in recent literature. The generic BPC framework can, under certain conditions, handle the complexities of multi-block warehouse layouts and support both optimal and heuristic picker routing strategies. A key contribution of this work is the analysis of the monotonicity properties of routing strategies in multi-block configurations, which are crucial for the application of the BPC method. The extended approach is thoroughly evaluated on publicly available benchmark instances. Computational results demonstrate that both the exact BPC and BPC-based heuristics offer significant improvements for solving the multi-block OBP compared to the current state-of-the-art methods. Specifically, instances with up to 80 orders are solved to proven optimality for five different routing strategies.

3.1 Introduction

The efficiency of order picking operations is a critical determinant of overall warehouse performance, particularly in large-scale distribution centers where order picking can account for a significant portion of operational costs (de Koster *et al.* 2007, Tompkins *et al.* 2010, Richards 2017). Order consolidation is central to optimizing these operations (Gademann and van de Velde 2005). It allows the order pickers to fulfill multiple customer orders in a single picking tour through the warehouse, which reduces both the number of picking tours and the total length of the tours (Hong *et al.* 2012). Consequently, it significantly reduces the overall picking effort compared to single order picking, where each order is fulfilled through a separate, individual tour (de Koster *et al.* 1999b, Petersen and Aase 2004). The arising *order batching problem* (OBP) consists of combining a given set of customer orders, each consisting of one or more individual items, into batches such that each order is assigned to exactly one batch, all batches satisfy the capacity restriction, and the total distance traveled by the pickers in the warehouse to fulfill the batches is minimized. The batch capacity is given by a maximal number of items that fit into a batch. Each batch is handled by an order picker who navigates the warehouse to collect the individual items required for all orders in the batch, following a predefined routing strategy. The problem of determining the shortest route for picking a set of items based on a predefined routing strategy has been coined the *single picker routing problem* (SPRP) and constitutes a special case of the *traveling salesman problem* (TSP, Burkard *et al.* 1998). The OBP for optimal routing, which involves finding the minimum distance tour for each batch, is commonly referred to in the literature as the *joint order batching and picker routing problem* (JOBPRP, e.g., Won and Olafsson 2005, Kulak *et al.* 2012, Valle *et al.* 2016, Briant *et al.* 2020). This terminology emphasizes the simultaneous decisions of assigning orders to batches and determining the shortest TSP tour for each batch. Besides the optimal routing strategy, heuristic routing strategies can be employed to solve the OBP. Those are often preferred in practice, as the resulting routes tend to be more intuitive for the (usually human) pickers to follow (de Koster *et al.* 1999b, Grosse *et al.* 2014). The OBP constitutes an $\mathcal{NP}$-hard optimization problem (Gademann and van de Velde 2005). We refer to Pardo *et al.* (2024) for a comprehensive overview on variants of the OBP and their taxonomy.

Warehouses usually exhibit parallel picking aisles that can be accessed by the pickers from both sides. Traditionally, research on the OBP has focused on a rectangular single-block warehouse layout, where all aisles are connected by two horizontal cross aisles (e.g., Gademann *et al.* 2001, Hong and Kim 2017, Menéndez *et al.* 2017, Boysen *et al.* 2017, Žulj *et al.* 2018). This layout allows to solve the SPRP for relatively straightforward routing strategies. The picking tour for optimal routing can be determined by a *dynamic programming* (DP) algorithm

proposed by Ratliff and Rosenthal (1983) which exploits the structure of the rectangular single-block warehouse and solves the SPRP in linear time (Heßler and Irnich 2022b). Heuristic routing strategies for the single-block warehouse are, for example, traversal (Goetschalckx and Ratliff 1988), return, midpoint, largest gap (Hall 1993), composite (Petersen 1995), combined (Roodbergen and de Koster 2001a), and mixed (Bahçeci and Öncan 2022).

However, many modern warehouses are equipped with multiple horizontal cross aisles that partition the space into distinct blocks to improve picking efficiency and flexibility (Roodbergen and de Koster 2001a, Schiffer *et al.* 2022). Studies indicate that incorporating an additional cross aisle can significantly reduce the total travel distance (e.g., Vaughan 1999, Valle and Beasley 2020). Nevertheless, a multi-block configuration introduces more complex routing scenarios. In warehouses with two blocks, an SPRP solution for the optimal routing strategy can be determined by the DP method of Roodbergen and de Koster (2001b). Cambazard and Catusse (2018) show that the DP approach can be scaled to any rectilinear Steiner tree problem in the plane in polynomial time, allowing the SPRP to be solved for the optimal routing strategy in any rectangular multi-block warehouse. Although its complexity is exponential in the number of cross aisles, SPRP instances with a reasonable number of blocks can be solved efficiently with the DP (Pansart *et al.* 2018). Several heuristic multi-block routing strategies have been proposed in the literature, including aisle-by-aisle (Vaughan 1999), no-reversal (Valle *et al.* 2017), traversal, combined, and largest gap (Roodbergen and de Koster 2001a). Notably, the latter three are extensions of their respective single-block counterparts.

In a recent work, Wahlen and Gschwind (2023) present a *branch-price-and-cut* (BPC) approach to solve the OBP in single-block warehouses for six routing strategies. The BPC method offers a powerful tool for warehouse optimization as it can be applied to any warehouse layout and routing strategy, provided that the chosen routing strategy is monotone. A *monotone* routing strategy is characterized by a distance function that is monotonously increasing as additional items are picked. However, the introduction of a multi-block warehouse layout presents new challenges in identifying such monotone routing strategies. Building on previous research to solve the SPRP, this work focuses on exploring the monotonicity property of six routing strategies in multi-block layouts. Our approach is to extend the state-of-the-art BPC algorithm, along with two BPC-based heuristics, to address the multi-block OBP for suitable routing strategies.

3.1.1 Contributions

This work contributes to the ongoing development of advanced order batching methods, offering new insights for tackling the challenges of multi-block rectangular warehouse environments. Building on the BPC approach of Wahlen and Gschwind

(2023) which excels at solving the OBP in single-block warehouses, our research extends this method to address the multi-block OBP. The key contributions of this paper can be summarized as follows:

- We thoroughly investigate the monotonicity of six routing strategies in multi-block warehouses, a crucial factor for the effective application of the BPC method.
- We analyze the computational effects of five monotone routing strategies on the BPC's performance, providing insights into their impact on multi-block order batching efficiency.
- The exact BPC approach and two derived BPC-based heuristics serve as a new standard for future research as they outperform the current state-of-the-art methods on publicly available two-block benchmark instances.

3.1.2 Organization of the Paper

The remainder of the paper is structured as follows. In Section 3.2, we review the related literature. Section 3.3 formally defines the OBP, presents a set-partitioning formulation of the problem, and briefly summarizes the BPC algorithm. The considered multi-block warehouse layout and routing strategies are specified in Section 3.4, where special attention is paid to the monotonicity property of the multi-block routing strategies. Section 3.5 presents our computational results. Final conclusions are drawn in Section 3.6.

3.2 Literature Review

In the following, we present the current state of research considering both exact and heuristic solution approaches to solve the OBP in rectangular warehouses consisting of two or more blocks. A summary of the approaches including our methods is provided in Table 3.1.

To the best of our knowledge, there are three exact methods in the literature to date, all of which use *branch-and-cut* (B&C) methods that can be applied to solve the JOBPRP in warehouses with an arbitrary number of blocks.

Valle *et al.* (2016) introduce three *integer linear programming* (IP) formulations, two of which are compact network flow formulations that can be solved with a general-purpose *mixed integer linear programming* (MIP) solver. For the third formulation, which contains exponentially many connectivity constraints, a B&C method is presented. The approaches are evaluated on a benchmark generated by Valle *et al.* (2016), which is based on publicly available data from the supermarket chain Foodmart (Thia 2008), referred to as `Foodmart`. `Foodmart` instances with up to 15 orders can be solved by at least one of the three methods within

Reference	# Blocks	Routing strategy	Main method	Benchmark
Panel A: Exact approaches				
Valle *et al.* (2016)	Arbitrary	Optimal	B&C	`Foodmart`
Valle *et al.* (2017)	Arbitrary	Optimal, no-reversal	B&C	`Foodmart`
Zhang and Gao (2023)	Arbitrary	Optimal, no-reversal	B&C	`Foodmart`
Our approach	Arbitrary	Optimal, no-reversal, aisle-by-aisle, traversal, combined	BPC	`Foodmart`, `Scholz&Wäscher`
Panel B: Heuristic approaches				
Kulak *et al.* (2012)	Arbitrary	Optimal	Cluster-based tabu search	Kulak *et al.* (2012)
Scholz and Wäscher (2017)	Two	Optimal, aisle-by-aisle, traversal, largest gap, combined, combined^{+}, DP-based heuristic	Iterated local search	`Scholz&Wäscher`
Valle and Beasley (2020)	Arbitrary	Optimal, no-reversal	Distance approximation	`Foodmart`
Briant *et al.* (2020)	Arbitrary	Optimal	CG heuristic, hill-climbing	`Foodmart`, Bué *et al.* (2019)
Our approach	Arbitrary	Optimal, no-reversal, aisle-by-aisle, traversal, combined	BPC-based heuristics	`Foodmart`, `Scholz&Wäscher`

Table 3.1: Overview of exact and heuristic solution approaches to the multi-block OBP

six hours of computation time. For small instances, the compact formulations perform better overall, whereas the formulation with an exponential number of constraints dominates with an increasing number of orders because of stronger lower bounds obtained. Valle *et al.* (2017) add highly effective valid inequalities to the non-compact formulation of Valle *et al.* (2016) exploring the warehouse layout. The resulting extended B&C optimally solves `Foodmart` instances comprising up to 20 orders for optimal and no-reversal routing within six hours. The proposed method can also be used to solve the SPRP for the optimal routing strategy with respect to a given set of orders. The authors exploit this by heuristically determining batches for OBP instances with up to 5,000 orders and solving the corresponding SPRP with the above-mentioned approach. Further improvements of the IP formulation are suggested by Zhang and Gao (2023) by reconstructing the connectivity constraints. Their B&C solves modified `Foodmart` instances with up to 23 orders within 40 minutes of computation time for the routing strategies optimal and no-reversal.

In most heuristic approaches, the decisions on batching and routing are made separately. Kulak *et al.* (2012) present a cluster-based tabu search method in order to obtain feasible batches of orders. The corresponding picking tours for optimal routing are determined in a second step by applying a combination of nearest neighbor and or-opt heuristics, or a combination of savings and 2-opt heuristics. The approach demonstrates superior performance compared to a genetic algorithm for the traversal routing strategy in terms of solution quality. An iterated local search method for two-block warehouse layouts is proposed by Scholz and Wäscher (2017). They apply the local search operators swap, shift, and perturbation to solve the OBP for the routing strategies optimal, traversal, largest gap, aisle-by-aisle, combined, combined$^+$, and a heuristic based on the optimal strategy. The approach is assessed by comparing the resulting distances across the different routing strategies, using self-generated instances referred to as `Scholz&Wäscher`. The authors observe that the total distance resulting from their iterated local search approach for the heuristic routing strategies is on average up to 25% longer than that achieved for the optimal routing strategy. Valle and Beasley (2020) suggest an edge-based distance approximation MIP to determine the assignment of orders to batches without directly addressing the SPRP. The use of sub-tour elimination constraints, which are usually very computationally extensive, becomes obsolete with their formulation. Once the batches have been created on the basis of approximated distances, their associated picking tours are determined for the strategies optimal and no-reversal applying the SPRP approach of Valle *et al.* (2017). Computational studies on the `Foodmart` benchmark with up to 75 orders show that the distance approximation approach provides qualitatively similar results to the B&C method of Valle *et al.* (2017) with significantly less computation time. Bri-

ant *et al.* (2020) propose a *column generation* (CG) heuristic which is based on a set-covering formulation of the OBP. The iterative CG process consists of solving a relaxed pricing MIP, which underestimates the actual distance of the picking tour and therefore the reduced cost of the batch. Only for batches with negative estimated reduced costs, the corresponding SPRP for the optimal or no-reversal routing strategy is solved with the approach of Cambazard and Catusse (2018). After the CG procedure, the final OBP solution is determined by solving a set-covering MIP based on the identified batches and their actual distances by a MIP solver. The approach is complemented by a post-optimization process in the form of a hill-climbing procedure initiated from the best feasible solution found. The CG heuristic improves several *best-known solutions* (BKS) from (Valle *et al.* 2017) for the `Foodmart` benchmark including very large-scale instances consisting of up to 5,000 orders. It also provides precise upper bounds for the more general industrial instances introduced by Bué *et al.* (2019).

3.3 Problem Description and Solution Approach

In this section, we formally define the OBP and present a mathematical formulation of it. We also provide an overview of the exact BPC and the BPC-based heuristics.

3.3.1 Problem Definition and Mathematical Formulation

Given a set of customer orders $O = \{1, \ldots, n\}$, each consisting of a set of items to be picked in the warehouse, and a sufficient number of pickers with capacity Q. The capacity consumption of each order $o \in O$ is given by $q_o \geq 0$. A feasible batch represents an order subset $b \subseteq O$ satisfying $\sum_{o \in b} q_o \leq Q$. We assume Q to be sufficiently large to encompass any order, i.e. $q_o \leq Q$ for all $o \in O$, as splitting of the orders is not allowed. According to a predefined routing strategy, an order picker navigates through the warehouse to retrieve the items required to fulfill all orders within a batch. The OBP consists of grouping all orders into capacity-feasible batches such that each order is assigned to exactly one batch, and the total distance traveled to pick the batches is minimized.

Let Ω denote the set of all feasible batches. The distance traveled to pick batch $b \in \Omega$ according to the routing strategy in use is given by a function c_b. Binary parameters r_{ob} indicate if order $o \in O$ is contained in batch $b \in \Omega$ or not. Binary decision variables λ_b are equal to one if batch $b \in \Omega$ is selected in the solution and zero otherwise. The set-partitioning formulation of the OBP is given as

follows:

$$\min \sum_{b \in \Omega} c_b \lambda_b \tag{3.1a}$$

$$\text{s.t.} \sum_{b \in \Omega} r_{ob} \lambda_b = 1 \quad \forall o \in O \tag{3.1b}$$

$$\lambda_b \in \{0, 1\} \quad \forall b \in \Omega \tag{3.1c}$$

The Objective (3.1a) minimizes the total distance traveled, and Constraints (3.1b) ensure that each order is assigned to exactly one batch. Observe that the underlying warehouse layout and the given routing strategy are only taken into account by the function c_b, which generally allows application of Formulation (3.1) to any warehouse scenario. However, c_b is generally not separable in $o \in b$, i.e., the total distance traveled for a batch cannot be separated into the individual orders the batch comprises.

3.3.2 Branch-Price-and-Cut Method

Even with small instance sizes, enumerating all feasible batches leads to a large set Ω, which makes it hardly possible to solve Formulation (3.1) directly. Therefore, BPC-based approaches have been introduced for its solution. These are *branch-and-bound* (B&B) methods where a CG technique is employed in each node of the B&B tree to compute the lower bounds. The CG alternates between solving a *restricted master problem* (RMP), which is the linear relaxation of Formulation (3.1) considering only a subset $\bar{\Omega} \subset \Omega$ of the batches, and solving a pricing problem in order to augment $\bar{\Omega}$ with promising batches. To initialize the BPC process, a start heuristic can be applied to generate an initial set of feasible batches $\bar{\Omega}$. The pricing problem consists of identifying batches with negative reduced costs. Cuts can be added to strengthen the linear relaxations.

For general information on CG and branch-and-price, we refer to (Desrosiers *et al.* 2024). In the following, we briefly describe the main components of the exact BPC and the two BPC-based heuristics employed, and refer to (Wahlen and Gschwind 2023) for further details on the algorithms and the computational setup.

3.3.2.1 Exact BPC

Pricing Problem The reduced cost of a batch $b \in \Omega$ is defined as $\tilde{c}_b = c_b - \sum_{o \in b} \pi_o$, where π_o denotes the dual prices associated with Constraints (3.1b). The CG pricing problem involves either identifying at least one feasible batch $b \in \Omega$ with a negative reduced cost or proving that no such batch exists. If no batch with a negative reduced cost exists, the current solution is optimal for the relaxed

Formulation (3.1), and thus, the CG algorithm terminates. Otherwise, at least one batch with negative reduced cost is added to the RMP.

The pricing problem can be formulated as a *shortest path problem with resource constraints* (SPPRC) on a linear directed multigraph $G = (V, A)$ with $n+1$ vertices $V = \{0, \dots, n\}$ and $2n$ arcs A. Vertex 0 serves as an artificial source, whereas vertices $1, \dots, n$ correspond to the n orders. For each $v \in V \setminus \{0\}$, there is a pair of parallel arcs connecting vertices $v-1$ and v, representing the inclusion or exclusion of order v, respectively. Each arc is associated with the dual price and the capacity consumption that result from the inclusion (exclusion) of order v, specifically π_o and q_o (0 and 0). Solving the pricing problem for the OBP is equivalent to finding a capacity-feasible 0-n-path in G with minimum reduced cost. SPPRCs are typically addressed using DP labeling algorithms (Irnich and Desaulniers 2005). In our BPC, each label represents a partial path containing information such as the last vertex, the set of included orders, the accumulated capacity consumption, and the reduced cost. Those labels are iteratively extended from a given source (0) to a given sink (n) along the network arcs via dedicated resource extension functions.

The non-separability of the distance function presents two main challenges for the labeling algorithm. First, each label propagation requires a computationally extensive evaluation of the distance function to determine the label's reduced cost. Second, the commonly established dominance relation between labels in order to reduce the number of generated labels cannot be applied. To mitigate these issues, the particular labeling algorithm relies on a strong bounding procedure to avoid the enumeration of all feasible paths. It consists of calculating a lower bound for the reduced cost of each capacity-feasible 0-n-path in G that contains the 0-v-path corresponding to a given label at vertex v. The completion bounds can be calculated by solving a single binary knapsack problem. Any labels that cannot be extended to yield a negative reduced cost through any capacity-feasible extension (i.e., those for which the sum of their reduced cost and the corresponding completion bound is greater than or equal to zero) can be discarded. However, the validity of this bounding procedure requires monotonicity of the routing strategy.

Cutting Valid inequalities in the form of *capacity cuts* (CCs) and *subset-row cuts* (SRCs) are added dynamically to strengthen the RMP. CCs ensure that for a subset of orders, the number of batches covering these orders is not smaller than a minimum number of batches needed to accommodate all orders comprised in the subset. Three different methods are used for their separation: a greedy construction procedure, a connected component-based heuristic, and a MIP-based approach. SRCs guarantee that the number of batches containing two or more orders from any subset of three orders is not greater than one. They are separated by simple enumeration. Both types of cuts are non-robust. The CCs additionally

influence the computation of the completion bounds.

Branching Branching is first performed on the number of batches, if fractional, by adding a linear constraint to the RMP. On the second level, a branching procedure based on (Ryan and Foster 1981) is applied. If in the current solution of the RMP, two orders $i, j \in O$ are assigned to the same batch b that has a fractional value λ_b, we can branch on that order pair. One branch ensures that i and j are assigned to separate batches by forcing variables λ_b with $r_{ib} = r_{jb} = 1$ to zero, whereas the other branch ensures that i and j are together in a batch by forcing variables λ_b with $r_{ib} + r_{jb} = 1$ to zero. Both types of decisions can be realized in the RMP by excluding the corresponding batch columns that must also be prevented from being regenerated in the CG. As the node selection strategy, the best-bound-first search is used.

3.3.2.2 BPC-based Heuristics

Building on the powerful CG component of the BPC, two heuristics are derived to solve the OBP. The first one, known as the *set-covering heuristic* (SC) employs a MIP solver to address Formulation (3.1) over all batch columns generated up to the root node, without utilizing branching techniques. The second heuristic, referred to as *depth-first* (BPC-DF), integrates best-bound-first and depth-first search strategies as the node selection method within the BPC algorithm. These approaches are specifically designed to quickly generate strong upper bounds, prioritizing speed over achieving proven optimality.

3.4 Multi-Block Routing Strategies

In this section, we describe the considered multi-block warehouse layout and provide a detailed description of the routing strategies that can be used to solve the SPRP in a multi-block layout. Furthermore, we formally define the monotonicity property of a routing strategy and investigate it for each of the introduced strategies.

3.4.1 Warehouse Layout

We consider a rectangular warehouse layout with parallel vertical aisles of equal length and width. The standard *single-block* warehouse features two perpendicular cross aisles that end the front and back of each aisle. In this paper, we focus on a generalized layout with one or more additional horizontal cross aisles at intermediate positions of the aisles, dividing the warehouse into two or more blocks and

each aisle into just as many sub-aisles. We refer to this kind of warehouse layout as a *multi-block* layout, consisting of $H \geq 2$ blocks $\{1, 2, \ldots, H\}$ and $H + 1$ cross aisles $\{0, 1, \ldots, H\}$, both indexed according to decreasing distance from the depot. Each sub-aisle of a block $h \in \{1, 2, \ldots, H\}$ can be entered from both its back cross aisle $h - 1$ and front cross aisle h. Racks are positioned on both the left and right side of each sub-aisle. Each rack contains several storage locations and each storage location can hold multiple items. We assume, however, that each item is assigned to a single, predetermined storage location. Pickers always travel in the horizontal center of the aisles and cross aisles. The retrieval of items is performed from the vertical center of a storage location, eliminating the need for horizontal movement. The starting and ending point of each picking tour is a common depot located in cross aisle H in front of the leftmost vertical aisle.

Figure 3.1 presents a top-down view of an exemplary instance of the OBP consisting of $n = 5$ orders in a warehouse with $H = 3$ blocks and $S = 6$ aisles. Each sub-aisle exhibits five storage locations on both sides. The individual storage locations of the required items are labeled with the order number $o \in \{1, \ldots, 5\}$ to which they belong. In the following examples, we assume a vertical length of $\ell = 1$ per storage location so that each rack has a length of $L = 5$. Let the horizontal distance between two adjacent aisles be $W = 3$, and the distance to enter or leave a sub-aisle from a cross aisle be $a = 1$.

3.4.2 Detailed Description of Routing Strategies

In general, heuristic routing strategies offer an initial advantage over optimal routing due to their intuitive design, making them easier for pickers to apply in practice. However, as the number of blocks in a warehouse increases, the complexity of heuristic picking routes also potentially grows, diminishing this advantage. On the other hand, as demonstrated by our computational study in Section 3.5, the trade-off between total distance traveled and computation time suggests that addressing the multi-block OBP for heuristic routing strategies may still be a justified choice.

All multi-block routing strategies examined in this paper – optimal, no-reversal, aisle-by-aisle, traversal, combined, and largest gap – are detailed below. We designate a block, aisle or sub-aisle as *required* if it contains at least one storage location with an item that needs to be picked and has not yet been accessed. The resulting picking routes are depicted in Figure 3.2 for batch $b = \{1, 2, 3, 4, 5\}$, illustrating the respective routing strategies employed.

Optimal In an SPRP tour for the optimal routing strategy, all required sub-aisles are visited either by a single traversal, a double traversal, a return trip (U-turn) from the front or back cross aisle, or a double return visit omitting the largest gap

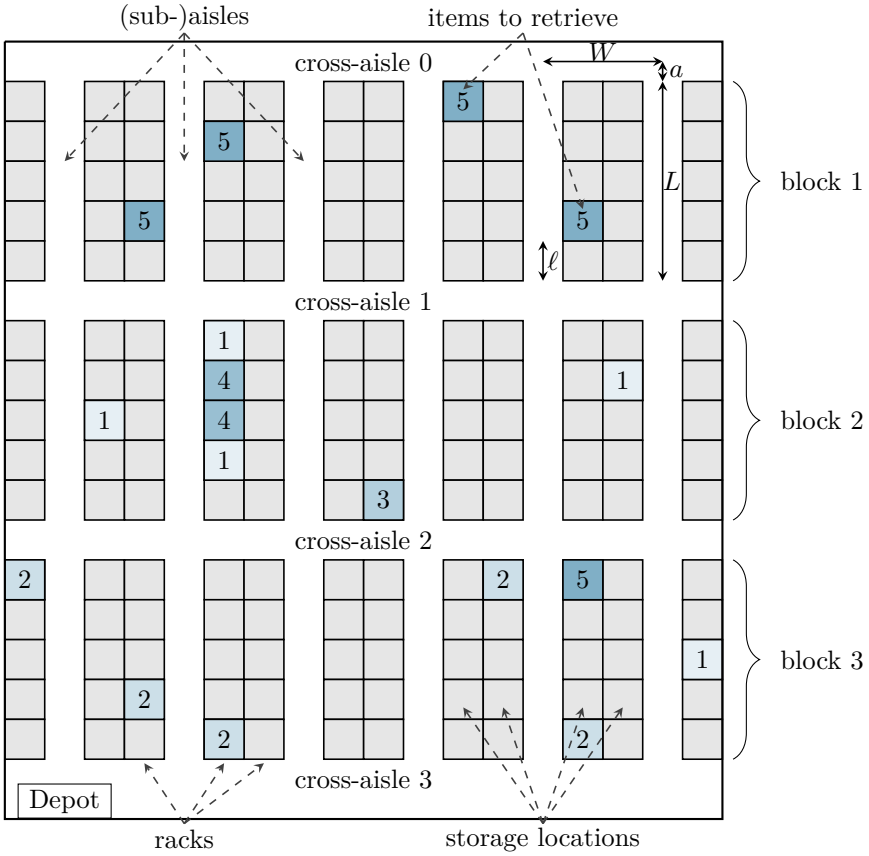


Figure 3.1: Rectangular parallel-aisles three-block warehouse layout

between two required storage locations of the sub-aisle. Each cross aisle section between two adjacent aisles is passed through a maximum of two times. The minimum distance of all feasible picking tours that start and end at the depot can be determined by solving a DP based on the aforementioned sub-aisle and cross aisle transitions. We refer to (Cambazard and Catusse 2018, Ratliff and Rosenthal 1983) for a description of the general DP algorithm and to (Pansart *et al.* 2018) for details on its application to the SPRP in multi-block warehouses.

No-Reversal The no-reversal strategy proposed by Valle *et al.* (2017) seeks a minimum distance tour that completely traverses all required sub-aisles. It does not allow return movements within a sub-aisle but only in the cross aisles, which can help to avoid congestion within the aisles. A no-reversal picking tour can be determined by applying the distance-minimizing DP for the optimal routing strategy, omitting all return moves in the sub-aisles. In other words, it is an "optimal" tour under the constraint that return movements are prohibited.

Aisle-by-Aisle The aisle-by-aisle strategy was specifically designed by Vaughan (1999) for use in multi-block warehouses. The core concept is to enter each aisle no more than once, visiting all required sub-aisles of an aisle consecutively. A DP algorithm is used to determine the cross aisles that minimize the total travel distance when transitioning between consecutive aisles, moving from left to right through the warehouse. Note that the original approach terminates after visiting the rightmost required aisle. We extend this by incorporating the order picker's return to the depot using its closest cross aisle, following the method proposed by Roodbergen and de Koster (2001a).

Traversal The multi-block traversal (also known as the S-shape) strategy presented by Roodbergen and de Koster (2001a) operates sequentially, advancing from block to block based on increasing index. In this approach, all items within a block are picked consecutively before moving on to the next block. Starting from the depot, the picker enters the leftmost required aisle and traverses it entirely up to the front cross aisle of the farthest required block, which is traversed horizontally to access the block's leftmost required sub-aisle. Each required sub-aisle in that block is fully traversed from left to right. If the number of required sub-aisles in the block is odd, the rightmost required sub-aisle is visited by a return move in order to exit into the block's front cross aisle. Roodbergen and de Koster (2001a) define that the picker then moves to the closest of either the leftmost or rightmost required sub-aisles in the next block that is closer to the depot. In this block, the process of traversing all required sub-aisles is repeated, and this continues for each subsequent block until the depot is reached. In case of an even number of required sub-aisles, a return move is performed in the respective last approached sub-aisle. If a block between the farthest required block and the depot is not required, it is just traversed. The picker returns to the depot using the foremost cross aisle.

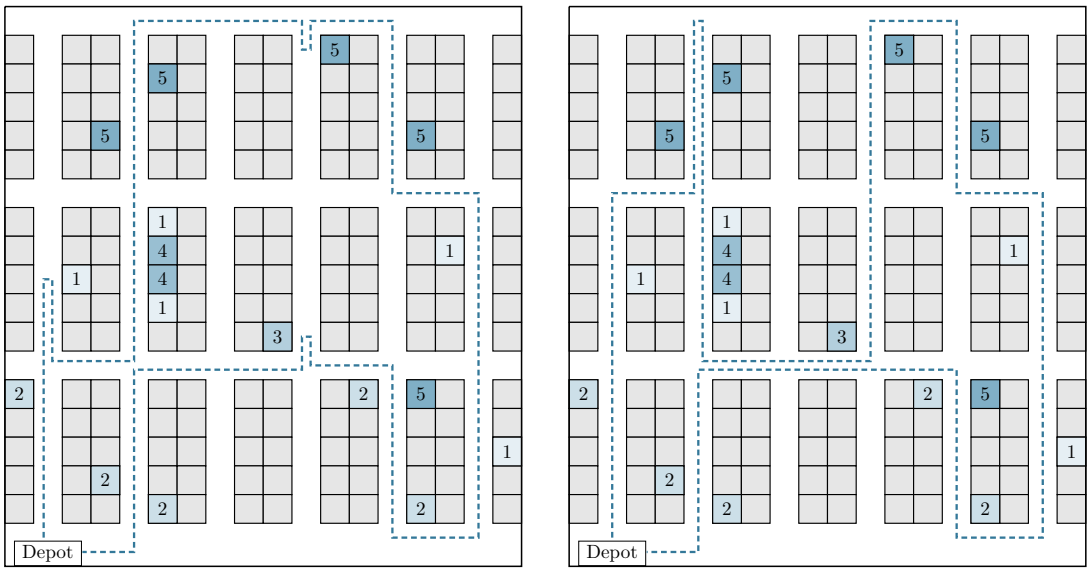
To ensure monotonicity of the traversal strategy – which is not guaranteed by the above definition provided by Roodbergen and de Koster (2001a), as demonstrated in Section 3.4.3 – we establish the horizontal travel direction for each block as follows: All odd-indexed blocks are traversed from left to right, whereas all even-indexed blocks are traversed from right to left. This definition does not alter the fact that the leftmost required aisle is always traversed to reach the farthest required block first. As previously stated, each required sub-aisle is traversed, and the picker exits each block into its corresponding front cross aisle, which may necessitate a return in the last visited sub-aisle. If there is only one required sub-aisle in the farthest required block, it is visited by a return move. Otherwise, the leftmost required sub-aisle is traversed. If the block has an odd index, the picker collects all remaining items in this block moving toward the rightmost required sub-aisle, according to the previously described policy. In contrast, if the block

has an even index, the picker moves horizontally along the block's back cross aisle to reach the rightmost required sub-aisle. Starting from there, the picker visits all required storage locations (that have not yet been accessed) within the block, moving toward the left. Subsequently, when approaching the next required block, if it is even (odd), the picker moves horizontally to the rightmost (leftmost) required sub-aisle of that block and visits all required sub-aisles, proceeding toward the leftmost (rightmost) required sub-aisle of the block.

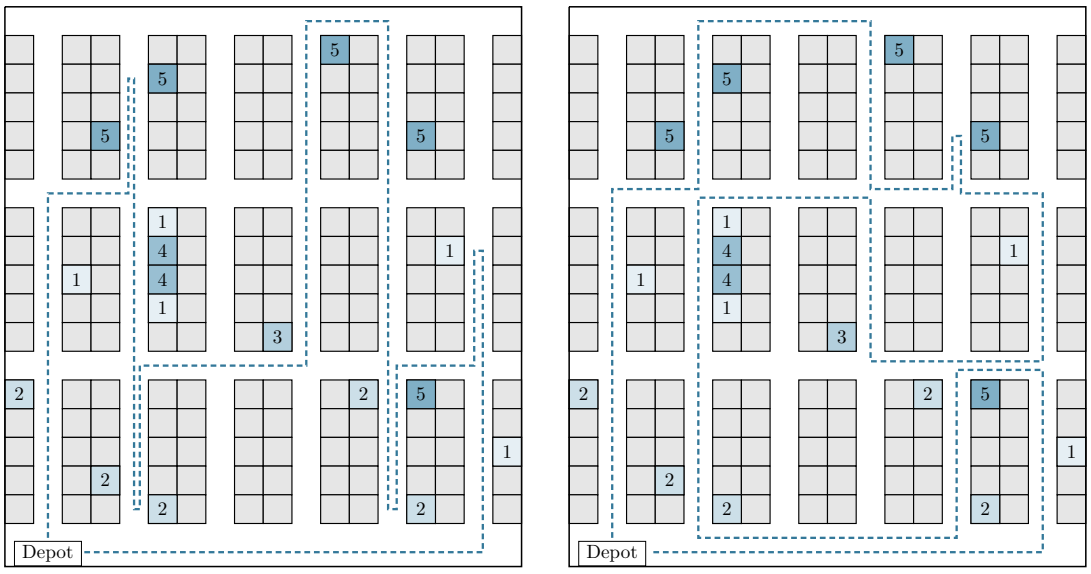
Our revised definition of the traversal strategy, ensures consistency of the picking tours, which not only enhances intuitiveness for the pickers but also preserves monotonicity of the routing strategy within a multi-block warehouse layout (see Section 3.4.3). In a two-block setting, our definition offers significant potential for reducing the total travel distance compared to the conventional definition, as the resulting horizontal distance is consistently either smaller or equal, whereas the number of traversal and return moves remains unchanged. Note that Roodbergen and de Koster (2001a) suggest a similar improvement. However, in warehouses with more than two blocks, this strategy may be less advantageous in terms of total distance traveled.

Combined According to the combined routing strategy given by Roodbergen and de Koster (2001a), either a traversal or a return move is performed in each required sub-aisle. By solving a DP for each individual block, the minimum distance intra-block picking routes are determined which consist only of these two sub-aisle transitions. First, the leftmost required aisle is traversed to the front cross aisle of the farthest required block. After picking all items in that block from left to right according to the DP solution and leaving the block in its front cross aisle, the picker approaches the closest out of the leftmost required and rightmost required sub-aisles of the next block. The procedure is repeated block by block and the picker uses the foremost cross-aisle to return to the depot.

We propose a revised version of the combined approach by specifying horizontal travel directions for each block to guarantee monotonicity, a property not assured by the traditional definition given by Roodbergen and de Koster (2001a). Specifically, blocks with an odd index are visited from left to right, and blocks with an even index from right to left, if required. This approach adopts the sequence of required sub-aisle visits proposed for the traversal strategy. Excluding the leftmost required aisle, which is traversed first towards the farthest required block, the DP process for each block begins at the corresponding back cross aisle and terminates at the front cross aisle. An exception occurs when the farthest required block has an odd index or only one required sub-aisle, where the DP process is both initiated and concluded at the front cross aisle. In contrast, if the farthest required block has an even index, a traversal is performed within the leftmost required sub-aisle,

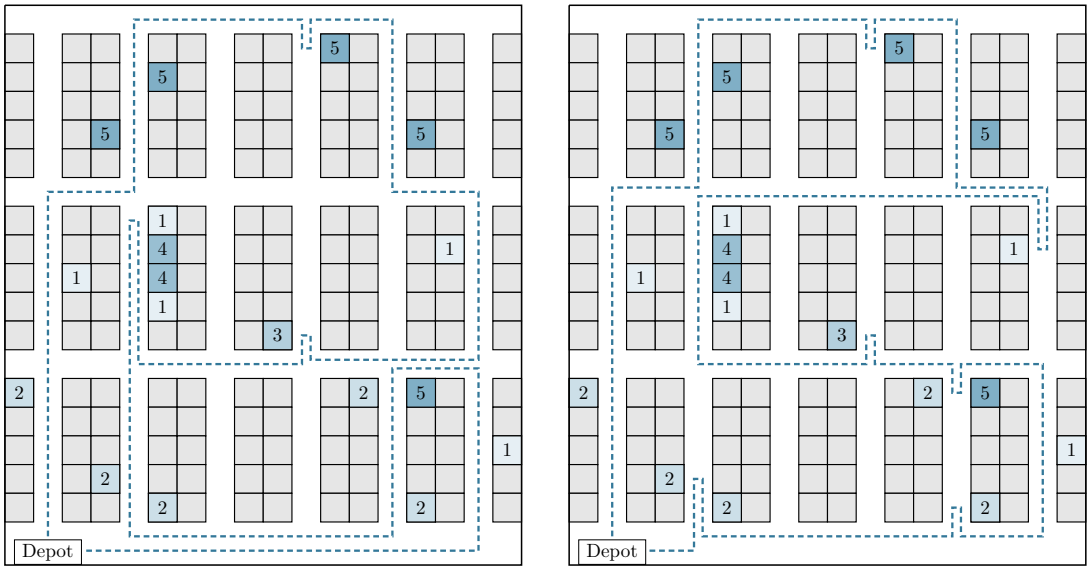


(a) Optimal strategy **(b)** No-reversal strategy



(c) Aisle-by-aisle strategy **(d)** Traversal strategy

Figure 3.2: Picking routes for batch $b = \{1, 2, 3, 4, 5\}$ and different routing strategies



(e) Combined strategy **(f)** Largest gap strategy

Figure 3.2: Picking routes for batch $b = \{1, 2, 3, 4, 5\}$ and different routing strategies (cont.)

allowing the DP process to begin at the back of the rightmost required sub-aisle. It is evident that in a two-block layout, our revised combined strategy results in a total horizontal distance that is either smaller than or equal to that of the traditional definition, similar to the improvement suggested by Roodbergen and de Koster (2001a). However, this advantage does not necessarily extend to the general multi-block scenario.

Largest Gap The multi-block largest gap routing strategy proposed by Roodbergen and de Koster (2001a) proceeds by visiting the blocks in increasing order of their index. The concept is to enter each required sub-aisle from both enclosing cross aisles in a manner that leaves out the maximum distance between two adjacent required storage locations within the sub-aisle, or between an enclosing cross aisle and its nearest required storage locations in the sub-aisle. The set of items in a block located above the gap is picked from the block's back cross aisle, the others from the respective front cross aisle. Starting at the depot, the leftmost required aisle is traversed to the front of the farthest required block. If there is only one sub-aisle required in this block, it is visited by a return move. Otherwise, the leftmost required sub-aisle of the block is traversed and the picker moves towards the rightmost required sub-aisle collecting all items that have to be picked from the back cross aisle. The last required sub-aisle of the block is traversed completely

and the picker collects the remaining items of that block from the front cross aisle in opposite horizontal direction. The picker then navigates to the farthest required sub-aisle among the next block's leftmost and rightmost required sub-aisles, visiting all required storage locations from the block's back cross aisle by following the shortest path. The farthest required sub-aisle is traversed and the picker picks the remaining items of the block from its front cross aisle. This procedure is repeated block by block towards the depot.

3.4.3 Monotonicity Property

The monotonicity of a routing strategy allows both modeling the OBP as a set-covering problem by replacing Constraints (3.1b) with covering restrictions, and using a dedicated bounding procedure to solve the CG pricing problem, constituting a key feature of the BPC approach (see Section 3.3). The monotonicity property can be formally defined as follows.

Definition 3.1. *A routing strategy is* monotone *if its distance function c_b is monotone, i.e., if for any two feasible batches $b_1 \subseteq b_2$ follows $c_{b_1} \leq c_{b_2}$.*

All routing strategies discussed, with the exception of the largest gap strategy, exhibit monotonicity in a warehouse with $H \geq 1$ blocks, as demonstrated by the following proposition.

Proposition 3.1. *The routing strategies optimal, no-reversal, aisle-by-aisle, traversal, and combined are monotone in a multi-block warehouse layout.*

Proof. Let R denote the set of storage locations required by orders in $b_2 \setminus b_1$. Without loss of generality, we assume in the following that R is non-empty (otherwise $c_{b_1} \leq c_{b_2}$ obviously holds).

Optimal The optimal routing strategy follows a distance-minimizing TSP tour over all required storage locations allowing all sub-aisle and cross aisle transitions. Since the distances between all storage locations satisfy the triangle inequality, an additional location in R can never reduce the length of an optimal TSP tour and we immediately have $c_{b_1} \leq c_{b_2}$.

No-Reversal The no-reversal routing strategy selects the minimum distance picking tour over all required storage locations allowing all cross aisle transitions but restricting movement within sub-aisles to traversal only. Because an additional location in R does not impact traversal length and cannot decrease the number of sub-aisles to be traversed, $c_{b_1} \leq c_{b_2}$ obviously holds for the no-reversal strategy.

Aisle-by-Aisle The aisle-by-aisle routing strategy chooses the distance-minimizing picking route, entering each required aisle exactly once. By definition, the total horizontal distance traveled for a batch is exactly twice the distance from the depot to the rightmost required aisle. This distance increases strictly if an additional storage location in R is located in an aisle that is further to the right than the rightmost required aisle of b_1. Otherwise it remains the same. Because the addition of a location in R does not affect the sub-aisle traversal length but may increase the distance of a return move, the total vertical traveled cannot decrease. Therefore, the inequality $c_{b_1} \leq c_{b_2}$ holds for the aisle-by-aisle strategy.

Traversal Due to the predefined sequence of sub-aisles to be visited according to our definition of the traversal strategy, the total horizontal distance traveled cannot be reduced by an additional required storage location in R. Conversely, any location in R that is positioned further to the right than the rightmost required sub-aisle or further to the left than the leftmost required sub-aisle of b_1 in any block may only result in additional horizontal distance. With regard to the vertical distances, any location in R does not affect the distance of a traversal but may only increase the distance for a return in the corresponding sub-aisle. Further, each additional required sub-aisle results in additional vertical distance (i.e., two traversals instead of a single return, or an additional return move) which has been shown by Wahlen and Gschwind (2023). Thus, the traversal strategy satisfies $c_{b_1} \leq c_{b_2}$.

Combined Within each block, our definition of the combined strategy chooses the distance-minimizing picking route using only the sub-aisle moves traversal and return, given the sequence of the sub-aisles to be visited. Any locations in R do not influence the distance of a traversal move, but can only increase the distance for returns in the corresponding sub-aisles. Accordingly, any additional required sub-aisle will result in an increase in the vertical distance traveled. Furthermore, any location in R that is located further to the right than the rightmost required sub-aisle or further to the left than the leftmost required sub-aisle of b_1 in any block can only increase the total horizontal distance traveled. Consequently, $c_{b_1} \leq c_{b_2}$ holds for the combined strategy.

□

The traditional definitions of the multi-block routing strategies traversal and combined, as proposed by Roodbergen and de Koster (2001a), do not adhere to the monotonicity property. Because the horizontal travel directions are not fixed a priori, the vertical distance traveled within a block may decrease if the

picking sequence of sub-aisles in the block is reversed due to an additional required storage location. Figure 3.3 shows the resulting picking tours for the traditional definition of the two strategies in a two-block layout. The picking route of batch $b_1 = \{2\}$ is illustrated with a dashed blue line. The red dotted route is obtained for batch $b_2 = b_1 \cup \{3\}$. The corresponding distances are $c_{b_1} = 49 > c_{b_2} = 46$. In contrast, according to our revised definitions of traversal and combined, the tour for b_1 follows the red dotted path without entering block 1, resulting in a total distance of 43.

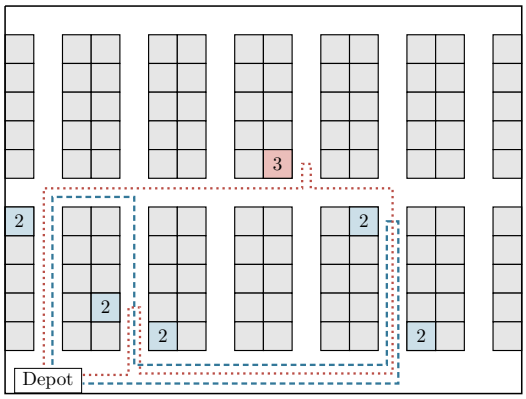


Figure 3.3: Picking routes for the traditional definitions of traversal and combined and batches $b_1 = \{2\}$ (blue) and $b_2 = \{2, 3\}$ (red)

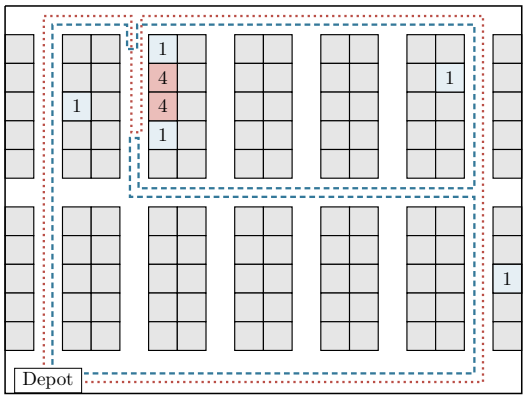


Figure 3.4: Picking routes for the largest gap strategy and batches $b_1 = \{1\}$ (blue) and $b_2 = \{1, 4\}$ (red)

The largest gap strategy exhibits non-monotone behavior in warehouses with $H \geq 2$ blocks, as illustrated by the example in Figure 3.4 within a two-block layout. The picking route corresponding to batch $b_1 = \{1\}$ comprising only order 1 is shown in blue dashed lines. Note that the largest gap in the second sub-aisle

of block 1 is between the two required storage locations in that sub-aisle, hence the sub-aisle needs to be visited twice by return moves from both enclosing cross aisles 0 and 1. Comparing the tour for batch $b_2 = b_1 \cup \{4\}$, shown as a red dotted line, the largest gap in the mentioned sub-aisle is now between the front cross aisle 1 and the nearest item. Accordingly, a single return move from cross aisle 0 is executed in this sub-aisle and there is no re-entering from cross aisle 1, making redundant any horizontal movement in that cross aisle. The resulting distances are $c_{b_1} = 90 > c_{b_2} = 67$, which contradicts the monotonicity property.

3.5 Computational Results

Our BPC-based approaches were implemented in `C++` and compiled into a 64-bit single-thread executable with MS Visual Studio 2022. CPLEX 20.10 with default parameters (except for the time limit and allowing only one thread) was used as a MIP solver. The computations were carried out on the HPC cluster Elwetritsch of the University of Kaiserslautern-Landau consisting of several Intel Xeon Gold 6126 processors running at 2.60 GHz. The computational setup utilized in this study aligns with that described in (Wahlen and Gschwind 2023) for all routing strategies and instances, with one notable exception: the travel distances of batches are stored in a hash table upon their initial computation. This method, previously discussed and tested by Wahlen and Gschwind (2023), was found to be ineffective for the single-block OBP. However, pretests demonstrated significant speedups for the multi-block OBP, thereby justifying its implementation in the current study. We set the time limit for each instance to 3,600 seconds. Unsolved instances are considered with the time limit of 3,600 seconds in our analysis. Instance-by-instance results are provided at `https://wiwi.rptu.de/fgs/logistik/obp-multiblock-detailedresults`.

3.5.1 Benchmark Instances

We focus our computational study on the benchmark prepared by Valle *et al.* (2016, 2017), which is derived from an industrial database (`Foodmart`), and on the instances generated by Scholz and Wäscher (2017) (`Scholz&Wäscher`). Only a small fraction of the considered instances have been solved to proven optimality before.

The `Foodmart` instances are based on anonymized online grocery purchases made at the supermarket chain Foodmart over a period of two years. All purchases made by a customer within a given number of days have been merged into a single order. Each instance consists of the n orders with the highest number of different items requested during the first Δ days, such that order size is positively correlated with

the number of days. The `Foodmart` benchmark is characterized by the number of orders $n \in \{5, 6, \ldots, 49, 50, 75\}$, the number of days $\Delta \in \{5, 10, 20\}$, and the warehouse configuration including a number of aisles $S \in \{8, 16\}$. The benchmark comprises 282 instances. Additionally, six large-scale instances consisting of up to 5,000 orders are considered for the heuristic approaches. The capacity of each picker is assumed to be $Q = 8$ boxes, each of which can accommodate up to 40 items belonging to the same order. The capacity consumption of each individual order $o \in \{1, \ldots, n\}$ consisting of $|o|$ items therefore corresponds to $q_o = \left\lceil \frac{|o|}{40} \right\rceil$ boxes (with $q_o \leq 2$ for the given instances). For more details on the instance generation, we refer to (Valle *et al.* 2017). The considered warehouse layout has $H = 2$ blocks either consisting of $S = 8$ aisles with rack lengths of $L^1 = 16$ in block 1 and $L^2 = 17$ in block 2, or $S = 16$ aisles with $L^1 = 8$ and $L^2 = 9$. The length of a single storage location is $\ell = 1$. Entering a sub-aisle corresponds to a travel distance of $a = 1.5$. The distance between two adjacent aisles is $W = 5$. The depot is located one unit distant from its nearest cross aisle in front of the leftmost aisle. The distance between the depot and the cross aisle is measured Euclidean, i.e, at the level of aisle $s \in \{1, \ldots, S\}$ it is $\sqrt{(1+a)^2 + ((s-1)W)^2}$.

Note that for the routing strategies optimal and no-reversal, the remote depot location in this setup is technically equivalent to introducing an additional block $H + 1$. It results in over 100 feasible states in the DP approach for solving the SPRP, compared to only 25 feasible states when the depot is located directly at cross aisle H of the warehouse (Roodbergen and de Koster 2001b). A detailed explanation of the state generation process, along with a comprehensive list of feasible states for the `Foodmart` instances, is provided in Appendix 3.A.

The `Scholz&Wäscher` instances are each specified by the number of orders $n \in \{20, 40, 60, 80\}$, the capacity $Q \in \{30, 45, 60, 75\}$, and the number of aisles $S \in \{10, 30\}$ in the warehouse. The number of items per order $o \in \{1, \ldots, n\}$ is uniformly distributed over $[5, 25]$ and defines its capacity consumption q_o. In the considered warehouse with $H = 2$ blocks, the length of each rack is $L = 25$, and the length of a single location is $\ell = 1$. The horizontal distance between two adjacent aisles is $W = 5$. To enter a sub-aisle, a distance of $a = 1$ must be traveled. There are 50 instances per (n, Q, S)-combination and thus, the `Scholz&Wäscher` benchmark comprises a total of 1,600 instances.

3.5.2 Comparison with State-of-the-Art

We first compare our BPC with the state-of-the-art B&C approaches of Valle *et al.* (2017) and Zhang and Gao (2023) for the optimal routing strategy. Valle *et al.* (2017) consider all `Foodmart` instances with $n \in \{5, 6, \ldots, 14, 15, 20, 25, 30\}$ orders and $S = 8$ aisles, and set a time limit of 21,600 seconds (six hours), whereas the

study of Zhang and Gao (2023) is based on the subset of `Foodmart` instances with $n \in \{5, 10, 15, 16, 17, \ldots, 24, 25, 30\}$ and $S = 8$ with a time limit of 2,400 seconds (40 minutes). Tables 3.2 and 3.3 show the comparison aggregated on the number of orders n, respectively. The tables contain information on the number of instances (*Inst*), the number of optimal solutions (*Opt*) and the average solution time in seconds ($t[s]$) of each approach. Note that the data presented for (Valle *et al.* 2017) in Table 3.2 is sourced from (Valle and Beasley 2020), who rerun the B&C calculations. Note further that Zhang and Gao (2023) simplistically assume a depot position within cross aisle 2, neglecting the Euclidean distances to the depot. In order to nevertheless allow a fair comparison with their algorithm we make the same assumption, causing our computation times stated in Table 3.3 to differ from those in the other tables. Instance-specific results are provided in Tables 3.13 and 3.14 in Appendix 3.B.

Tables 3.2 and 3.3 demonstrate that the BPC clearly outperforms both B&C methods. Whereas Valle *et al.* (2017) and Zhang and Gao (2023) optimally solve 35 and 27 of 42 considered instances, respectively, we solve all of these instances to proven optimality. Furthermore, our BPC method is on average more than three orders of magnitude faster than each of the B&C approaches for the considered instances. A striking conclusion from comparing the two tables is that the average computation time of the BPC reduces to a fraction when adopting the assumption of Zhang and Gao (2023) with respect to the depot location (e.g., 24.9 seconds vs. 4.6 seconds on average for $n = 30$). This demonstrates the exponential increase in the complexity of the DP algorithm used to solve the SPRP for optimal routing as a function of the number of blocks H in the warehouse (Cambazard and Catusse 2018).

3.5.3 Computational Analysis of BPC Algorithm

We consult both the `Foodmart` and the `Scholz&Wäscher` instances to examine which input parameters influence the performance of the BPC for the optimal routing strategy. The results are presented in Tables 3.4 and 3.5 where the additional columns each indicate the percentage of optimally solved instances (*%Opt*).

Table 3.4 shows the analysis of the `Foodmart` instances aggregated by the number of aisles S, the number of days Δ, and intervals of the number of orders n. The difficulty of the instances most significantly depends on n. All instances with $n \leq 20$ are solved in few seconds, but the computation time steadily increases with each interval of n. Only one out of six instances with $n = 75$ can be solved within one hour. Doubling S results in slightly fewer solvable instances (97.2% vs. 98.6%) and more than doubles the average computation time (410.3 seconds vs. 164.6 seconds), although the effective warehouse size in terms of of available storage locations remains almost identical. Interestingly, the average number of

		VBC (2017)		Our method	
n	Inst	Opt	t[s]	Opt	t[s]
5	3	3	3.1	3	0.0
6	3	3	7.6	3	0.0
7	3	3	7.5	3	0.0
8	3	3	13.5	3	0.0
9	3	3	59.5	3	0.0
10	3	3	83.7	3	0.1
11	3	3	325.7	3	0.2
12	3	3	311.0	3	0.2
13	3	3	455.1	3	0.4
14	3	3	1,480.8	3	0.4
15	3	3	1,003.6	3	0.3
20	3	2	11,852.9	3	3.1
25	3	0	21,600.0	3	7.1
30	3	0	21,600.0	3	24.9
Total	42	35	4,200.3	42	2.6

Table 3.2: Comparison of our BPC algorithm with the B&C approach of Valle *et al.* (2017) for the optimal routing strategy on a subset of the `Foodmart` instances

		Z&G (2023)		Our method	
n	Inst	Opt	t[s]	Opt	t[s]
5	3	3	0.4	3	0.0
10	3	3	10.9	3	0.0
15	3	3	41.6	3	0.1
16	3	3	282.3	3	0.1
17	3	3	737.3	3	0.2
18	3	3	451.0	3	0.3
19	3	2	900.3	3	0.6
20	3	2	898.3	3	0.6
21	3	2	893.0	3	2.1
22	3	2	917.0	3	0.9
23	3	1	1,696.7	3	0.4
24	3	0	2,400.0	3	1.1
25	3	0	2,400.0	3	2.6
30	3	0	2,400.0	3	4.6
Total	42	27	1,002.1	42	1.0

Table 3.3: Comparison of our BPC algorithm with the B&C approach of Zhang and Gao (2023) for the optimal routing strategy on a modified subset of the `Foodmart` instances

		$S = 8$			$S = 16$		
Δ	n	Inst	%Opt	t[s]	Inst	%Opt	t[s]
5	(0, 10]	6	100.0	0.0	6	100.0	0.0
	(10, 20]	10	100.0	1.1	10	100.0	3.5
	(20, 30]	10	100.0	22.7	10	100.0	58.8
	(30, 40]	10	100.0	201.9	10	100.0	424.5
	(40, 50]	10	100.0	391.1	10	90.0	1,829.7
	(50, 75]	1	0.0	3,600.0	1	0.0	3,600.0
Subtotal		47	97.9	207.8	47	95.7	569.5
10	(0, 10]	6	100.0	0.0	6	100.0	0.0
	(10, 20]	10	100.0	1.8	10	100.0	7.7
	(20, 30]	10	100.0	16.0	10	100.0	51.6
	(30, 40]	10	100.0	155.2	10	100.0	288.5
	(40, 50]	10	100.0	373.1	10	100.0	1,084.4
	(50, 75]	1	0.0	3,600.0	1	0.0	3,600.0
Subtotal		47	97.9	192.8	47	97.9	381.3
20	(0, 10]	6	100.0	0.0	6	100.0	0.0
	(10, 20]	10	100.0	0.3	10	100.0	0.7
	(20, 30]	10	100.0	4.3	10	100.0	11.5
	(30, 40]	10	100.0	43.6	10	100.0	124.4
	(40, 50]	10	100.0	221.9	10	100.0	820.5
	(50, 75]	1	100.0	1,679.5	1	0.0	3,600.0
Subtotal		47	100.0	93.2	47	97.9	280.2
Total		141	98.6	164.6	141	97.2	410.3

Table 3.4: Summary results of our BPC algorithm for the `Foodmart` instances and the optimal routing strategy

different items per order seems to have an opposing effect, as instances with larger parameter Δ are more likely to be solved optimally. For example, 96.8% of the instances with $\Delta = 5$ are solved, compared to 99.0% with $\Delta = 20$.

In the `Scholz&Wäscher` benchmark, the number of aisles S, the capacity Q, and the number of orders n are varied. The aggregated results of our BPC for these instances are presented in Table 3.5. Although the `Scholz&Wäscher` instances are significantly larger in terms of both warehouse size and number of orders, our BPC method performs almost equally well in solving them for optimal routing. This result can likely be attributed to the differing depot locations assumed in the two instance classes. In line with the findings from the `Foodmart` instances, we observe that the instance difficulty increases with higher values of n. However, the impact of S on performance appears less consistent compared to the `Foodmart` instances. Although `Scholz&Wäscher` instances with more aisles tend to be more challenging

		$S = 10$			$S = 30$		
Q	n	Inst	%Opt	t[s]	Inst	%Opt	t[s]
30	20	50	100.0	0.0	50	100.0	0.0
	40	50	100.0	0.0	50	100.0	0.0
	60	50	100.0	0.1	50	100.0	0.1
	80	50	100.0	0.1	50	100.0	0.1
Subtotal		200	100.0	0.0	200	100.0	0.1
45	20	50	100.0	0.1	50	100.0	0.1
	40	50	100.0	1.0	50	100.0	0.6
	60	50	100.0	10.1	50	100.0	2.9
	80	50	100.0	22.6	50	100.0	11.8
Subtotal		200	100.0	8.5	200	100.0	3.8
60	20	50	100.0	0.3	50	100.0	0.3
	40	50	100.0	8.1	50	100.0	7.3
	60	50	98.0	207.0	50	96.0	262.7
	80	50	86.0	778.4	50	92.0	666.8
Subtotal		200	96.0	248.5	200	97.0	234.3
75	20	50	100.0	0.8	50	100.0	1.2
	40	50	100.0	25.7	50	100.0	35.1
	60	50	94.0	636.1	50	98.0	640.9
	80	50	66.0	1,918.1	50	42.0	2,584.0
Subtotal		200	90.0	645.2	200	85.0	815.3
Total		800	96.5	225.5	800	95.5	263.4

Table 3.5: Summary results of our BPC algorithm for the `Scholz&Wäscher` instances and the optimal routing strategy

for our BPC on average, we generally observe a performance improvement for instances with $Q \leq 60$ as the number of aisles S increases from 10 to 30. The impact of Q is comparable to that of n. All instances with a small capacity ($Q \leq 45$) are solved optimally within an average of approximately three seconds. In contrast, only 87.5% of the instances with $Q = 75$ are solved, requiring an average computation time of more than 730 seconds.

Overall, the difficulty of the considered instances increases with an increasing number of orders n, an increasing number of aisles S, an increasing capacity Q (`Scholz&Wäscher`), or with a decreasing number of days Δ (`Foodmart`).

3.5.4 Evaluation of Routing Strategies

Table 3.6 summarizes the results for all instances per routing strategy, aggregated on the number of orders n (`Scholz&Wäscher`) or intervals of n (`Foodmart`). The

additional columns show the percentage deviation in the total travel distances for the respective routing strategy compared to the optimal strategy (*dev*). Overall, the choice of the routing strategy significantly influences both the solution quality and the computational performance of the BPC. Recall that the warehouse layout and especially the location of the depot in the `Foodmart` instances significantly increases the complexity of the SPRP solution for the routing strategies optimal and no-reversal (see Appendix 3.A), whereas it barely affects the other heuristic routing strategies. As anticipated, the optimal and no-reversal routing strategies pose greater computational challenges for the BPC method on the `Foodmart` instances compared to aisle-by-aisle, combined, and traversal. With each of the latter three strategies, 98.6% of the `Foodmart` instances are solved in an average runtime of approximately two minutes. In contrast, the average computation time for optimal and no-reversal is significantly higher, often by several multiples, and less optima are provided (97.9% and 94.0%). For the `Scholz&Wäscher` instances, computational performance across all strategies is more consistent, with the proportion of optimally solved instances ranging from 94.2% (no-reversal) to 96.6% (aisle-by-aisle) and average computation times between 212.1 seconds (aisle-by-aisle) and 330.3 seconds (no-reversal).

For all routing strategies considered, the difficulty increases as n increases. Table 3.6 reflects a significantly smaller percentage deviation in results for the `Foodmart` instances compared to the `Scholz&Wäscher` instances. This disparity can be attributed to the considerably larger warehouse dimensions (number and length of aisles) in the `Scholz&Wäscher` instance set. In both classes of instances, the no-reversal strategy yields the largest average travel distances, followed by the traversal strategy. Among the heuristic strategies, the combined strategy consistently achieves the shortest average travel distances. Notably, for the `Foodmart` instances, the BPC method for combined routing yields shorter total travel distances than the optimal routing strategy for $n = 75$, on average. This is because the BPC for optimal routing fails to solve the majority of these instances to optimality within the imposed time limit.

3.5.5 Detailed Analysis

A detailed analysis of our BPC is provided in Table 3.7 for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategies optimal and combined, aggregated on n. The two strategies are representative of a group of routing strategies each in terms of computation time and number of optimally solved instances (see Table 3.6). More detailed results for all routing strategies are provided in Appendix 3.C. The additional columns provide the average time to solve the LP relaxation in seconds (t^{LP}), the average optimality gap of the LP relaxation with respect to the BKS for the respective routing strategy (Gp^{LP}), the average optimality

		Optimal		No-reversal			Aisle-by-aisle			Traversal			Combined		
n	Inst	%Opt	t[s]	%Opt	t[s]	dev	%Opt	t[s]	dev	%Opt	t[s]	dev	%Opt	t[s]	dev
Panel A: `Foodmart` instances															
$(0, 10]$	36	100.0	0.0	100.0	0.0	9.5	100.0	0.0	5.3	100.0	0.0	7.3	100.0	0.0	2.0
$(10, 20]$	60	100.0	2.5	100.0	3.5	8.2	100.0	1.0	5.2	100.0	0.3	6.5	100.0	0.4	2.1
$(20, 30]$	60	100.0	27.5	98.3	183.7	7.9	100.0	7.6	5.3	100.0	16.9	6.5	100.0	7.7	1.8
$(30, 40]$	60	100.0	206.4	96.7	381.1	7.6	100.0	41.8	5.1	100.0	70.2	6.3	100.0	54.3	1.5
$(40, 50]$	60	98.3	786.8	86.7	1,011.7	7.8	100.0	141.5	5.0	100.0	249.5	6.3	100.0	206.7	1.6
$(50, 75]$	6	16.7	3,279.9	0.0	3,600.0	11.9	33.3	3,078.6	3.6	33.3	2,876.9	5.4	33.3	3,034.5	−0.5
Total	282	97.9	287.5	94.0	412.8	8.2	98.6	106.3	5.1	98.6	132.9	6.5	98.6	121.8	1.7
Panel B: `Scholz&Wäscher` instances															
20	400	100.0	0.3	100.0	0.5	20.9	100.0	0.3	10.1	100.0	0.3	18.7	100.0	0.2	5.5
40	400	100.0	9.7	100.0	31.0	20.0	100.0	12.0	9.8	100.0	10.2	18.0	100.0	9.1	5.3
60	400	98.2	220.0	95.0	376.7	19.7	98.5	151.1	9.6	97.0	229.5	17.8	97.5	212.9	5.2
80	400	85.8	747.7	81.8	913.2	19.9	88.0	685.1	9.5	86.2	718.7	17.6	86.8	736.8	5.1
Total	1,600	96.0	244.5	94.2	330.3	20.1	96.6	212.1	9.7	95.8	239.7	18.0	96.1	239.8	5.3

Table 3.6: Summary results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and all routing strategies

gap with respect to the BKS before the first Ryan-and-Foster branching is applied (Gp^{RF}), the average number of B&B nodes solved (Nds), and the average number of CCs (CC) and SRCs (SRC) added. Note that three of the largest `Foodmart` instances ($n = 75$) have been excluded from the gaps reported for optimal routing, as the algorithm provided no root node solution in the given time.

The LP gap and the gap after incorporating cuts are consistently small across all routing strategies and instances, with the exception of cases where $n = 75$. On average, `Scholz&Wäscher` instances exhibit significantly smaller gaps than `Foodmart` instances, and gaps are generally smaller for optimal routing compared to combined routing. For combined routing, solving the LP relaxation is highly effective, with average computation times of 2.7 seconds for `Foodmart` instances and 1.5 seconds for `Scholz&Wäscher` instances. In contrast, the corresponding times for optimal routing are significantly longer, at 104.8 seconds and 13.2 seconds, respectively. As a result, the total computation times for the BPC method are shorter for combined routing than for optimal routing, even though the average number of nodes in the B&B tree is considerably larger for combined routing.

When comparing instance classes, `Scholz&Wäscher` instances have an average B&B tree size more than ten times that of `Foodmart` instances. Nevertheless, for optimal routing, the average computation time is smaller for `Scholz&Wäscher` instances due to much shorter computation times per node, which can be attributed to differences in the underlying warehouse layouts. Conversely, for combined routing, the computation time for `Scholz&Wäscher` instances is approximately double that for `Foodmart` instances on average. This discrepancy arises because, despite

		Optimal								Combined							
n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
Panel A: `Foodmart` instances																	
$(0, 10]$	36	36	0.0	0.0	1.55	1.55	1	4	1	36	0.0	0.0	1.67	1.67	1	4	1
$(10, 20]$	60	60	2.5	1.2	2.40	2.28	2	29	10	60	0.4	0.0	2.36	2.13	2	26	12
$(20, 30]$	60	60	27.5	17.1	1.39	0.75	4	32	36	60	7.7	0.3	1.53	0.78	6	31	42
$(30, 40]$	60	60	206.4	91.8	1.40	0.42	34	32	68	60	54.3	1.8	1.37	0.50	38	30	65
$(40, 50]$	60	59	786.8	344.5	1.40	0.43	102	40	79	60	206.7	5.6	1.53	0.52	146	37	77
$(50, 75]$	6	1	3,279.9	380.1	10.78	10.16	332	29	44	2	3,034.5	49.3	9.78	9.01	1,140	38	86
Total	282	276	287.5	104.8	1.83	1.24	37	29	42	278	121.8	2.7	1.87	1.24	65	28	44
Panel B: `Scholz&Wäscher` instances																	
20	400	400	0.3	0.1	0.76	0.42	6	10	10	400	0.2	0.0	0.81	0.41	5	9	11
40	400	400	9.7	1.7	0.52	0.22	43	14	24	400	9.1	0.3	0.55	0.26	67	13	24
60	400	393	220.0	11.6	0.55	0.34	1,007	14	29	390	212.9	1.4	0.69	0.47	1,068	13	29
80	400	343	747.7	39.4	1.49	1.36	1,699	14	30	347	736.8	4.4	1.60	1.45	2,072	13	31
Total	1,600	1,536	244.5	13.2	0.83	0.58	689	13	23	1,537	239.8	1.5	0.91	0.65	803	12	24

Table 3.7: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategies optimal and combined

similar average computation times for the root node across instance classes, the larger B&B tree size in `Scholz&Wäscher` instances drives up the total computation time. The number of CCs and SRCs generated is comparable across all routing strategies within each instance class. However, `Foodmart` instances require generating approximately twice as many cuts as `Scholz&Wäscher` instances.

3.5.6 Computational Analysis of BPC-based Heuristics

As exposed in Table 3.7, the majority of computation time in the BPC process is spent on closing the optimality gap, with comparatively little time dedicated to solving the root node. This insight is leveraged by the BPC-based heuristics SC and BPC-DF. Table 3.22 in Appendix 3.C provides a comprehensive summary of our heuristic results across all `Foodmart` instances and routing strategies.

Our initial objective is to evaluate the two heuristics for the optimal routing strategy in comparison to the *distance approximation approach* (DAA) proposed by Valle and Beasley (2020) and the *column generation heuristic* (CGH) introduced by Briant *et al.* (2020). Following the methodology outlined by the authors, we focus on a subset of `Foodmart` instances characterized by $S = 8$ and $n \in \{5, 6, \ldots, 14, 15, 20, 25, 30\}$. Recall that all instances considered were solved to proven optimality using our BPC approach for optimal routing, with an average runtime of 2.6 seconds per instance (see Table 3.2). Valle and Beasley (2020) impose a time limit of six hours (21,600 seconds) and Briant *et al.* (2020) allow two

		DAA		CGH		BPC-DF		SC	
n	Inst	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp
5	3	1.2	0.00	0.9	0.00	0.0	0.00	0.0	0.00
6	3	1.0	0.61	1.2	0.00	0.0	0.00	0.0	0.00
7	3	1.6	2.40	3.9	0.00	0.0	0.00	0.0	0.00
8	3	1.7	3.85	5.1	0.00	0.0	0.00	0.0	0.00
9	3	3.0	2.74	17.5	0.08	0.0	0.00	0.0	0.00
10	3	2.7	3.46	21.7	0.14	0.1	0.00	0.1	0.00
11	3	2.5	2.00	60.1	0.29	0.2	0.00	0.1	0.88
12	3	2.7	1.78	90.1	0.00	0.2	0.00	0.2	0.33
13	3	3.1	1.60	94.2	0.32	0.4	0.00	0.3	0.86
14	3	4.8	3.41	123.1	0.18	0.4	0.00	0.4	0.00
15	3	3.9	3.49	147.9	0.07	0.3	0.00	0.3	0.00
20	3	11.7	2.91	1,057.4	1.08	3.0	0.00	2.1	0.42
25	3	90.5	3.40	4,404.2	0.80	6.4	0.00	4.7	0.72
30	3	3,724.4	3.55	7,293.7	2.37	21.0	0.00	18.5	0.05
Total	42	275.3	2.51	951.5	0.38	2.3	0.00	1.9	0.23

Table 3.8: Comparison of our heuristics BPC-DF and SC with the DAA of Valle and Beasley (2020) and the CGH of Briant *et al.* (2020) for a subset of the `Foodmart` instances and the optimal routing strategy

hours plus an additional 12 minutes for post-optimization (7,920 seconds in total). The gap columns (*Gp*) reported in Table 3.8 represent the percentage deviation from the optimal solution aggregated on n. An instance-by-instance comparison is available in Table 3.15 in Appendix 3.B.

Among the two methods from the literature, Valle and Beasley (2020) achieve a shorter average computation time of less than five minutes, with an average optimality gap of 2.51%. Briant *et al.* (2020) reduce the gap to an average of 0.38% at the cost of more than tripling the average computation time. In comparison, BPC-DF solves all considered instances to optimality with an average computation time of 2.3 seconds, whereas SC achieves even faster results, requiring 1.9 seconds of computation time and exhibiting a gap of 0.23% from the optimal solution on average. The results presented in Table 3.8 clearly demonstrate that both BPC-DF and SC outperform the DAA (Valle and Beasley 2020) and the CGH (Briant *et al.* 2020) for the instances under consideration, offering superior performance in terms of both computation time and solution quality.

We extend our analysis to larger `Foodmart` instances with $n \in \{25, 30, 50, 75\}$ orders and $S \in \{8, 16\}$ aisles, as suggested by Valle and Beasley (2020). Given the rapid root node computation times and the relatively small deviations from the BPC solution for optimal routing (see Tables 3.6 and 3.7), exploring the BPC-

				BPC-DF				SC			
		DAA		Optimal		Combined		Optimal		Combined	
n	Inst	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp
25	6	4,685.8	3.13	14.9	0.00	1.0	1.60	12.7	0.61	0.7	1.78
30	6	12,707.5	3.50	41.9	0.00	6.4	1.94	40.4	0.02	2.3	2.13
50	6	21,652.1	6.50	1,766.4	0.00	971.7	1.55	545.4	0.61	44.1	2.85
75	6	21,655.7	7.71	3,517.4	5.43	3,600.0	0.68	2,324.5	5.80	1,035.0	1.78
Total	24	15,175.3	5.21	1,335.2	1.36	1,144.7	1.44	730.7	1.76	270.5	2.13

Table 3.9: Comparison of our heuristics BPC-DF and SC for the routing strategies optimal and combined with the DAA of Valle and Beasley (2020) on a subset of large `Foodmart` instances

based heuristics for combined routing in addition to the optimal routing strategy appears promising. An evaluation of the methods aggregated on n is provided in Table 3.9, with an instance-by-instance comparison detailed in Table 3.16 in Appendix 3.B. Valle and Beasley (2020) set a time limit of 21,600 seconds for their DAA, in addition to the time required for computing the optimal SPRP solution. The overall BKS per instance across all routing strategies constitutes the basis for the reported gaps. All BPC-based heuristic variants demonstrate a significant reduction in the gaps, with computation times less than one-tenth of those reported by Valle and Beasley (2020). The highest-quality solutions, characterized by an average gap of approximately 1.4% relative to the BKS, are obtained using BPC-DF for either optimal or combined routing, with an average solution time of around 20 minutes. Notably, SC exhibits even shorter computation times than BPC-DF, with averages of 270.5 seconds for combined routing and 730.7 seconds for optimal routing, while maintaining relatively small gaps of 2.13% and 1.76%, respectively.

Table 3.10 confirms that the BPC-based heuristics yield favorable results, even with significantly reduced computation times. The table presents aggregated results for combined routing respecting time limits ($TL[s]$) of 100, 600, 1,800, and 3,600 seconds for the previously considered subset of large `Foodmart` instances. For both heuristics, the average gaps consistently decrease as the time limit increases, indicating a clear trade-off between computational time allowance and solution quality. SC utilizes a smaller proportion of the available computation time on average, whereas BPC-DF consistently provides gaps that are approximately one-third smaller across all time limits.

Finally, we evaluate our BPC-based heuristics on the basis of six very large-scale `Foodmart` instances with $S = 8$, $\Delta = 10$, and $n \in \{100, 200, 500, 1000, 2000, 5000\}$. For this analysis, we set a time limit of 7,200 seconds and employ the combined routing strategy. Briant *et al.* (2020) extend this time limit by an additional

	BPC-DF		SC	
TL[s]	t[s]	Gp	t[s]	Gp
100	46.0	2.63	35.1	3.75
600	233.2	1.56	135.4	2.36
1,800	622.3	1.47	251.7	2.18
3,600	1,144.7	1.40	270.5	2.10

Table 3.10: Summary results of our heuristics BPC-DF and SC for a subset of large `Foodmart` instances and combined routing with different time limits

	CGH			BPC-DF		SC	
n	t[s]	UB^{CG}	UB	t[s]	UB	t[s]	UB
100	TL	3,080.9	2,882.5	6,438.7	2,782.6	3,585.6	2,819.3
200	TL	6,716.3	5,608.8	OOM	5,088.4	7,177.4	5,345.6
500	TL	15,972.0	13,237.0	OOM	12,018.5	7,170.4	13,181.1
1,000	TL	30,719.3	25,504.2	OOM	29,218.4	OOM	29,670.5
2,000	TL	57,763.5	48,173.0	TL	56,643.2	OOM	56,643.2
5,000	TL	137,165.4	128,811.7	TL	133,207.0	TL	135,884.0

Table 3.11: Comparison of our heuristics BPC-DF and SC for the combined routing strategy with the CGH of Briant *et al.* (2020) for very large `Foodmart` instances

720 seconds to conduct a post-optimization process following the execution of their CGH. Table 3.11 presents the best *upper bound* (*UB*) achieved by each approach, as well as the UB prior to the post-optimization phase in (Briant *et al.* 2020) (UB^{CG}). Both BPC-DF and SC exhibit superior performance compared to the pure CGH without post-optimization, yielding smaller UBs and shorter computation times. This advantage persists even for instances where the root node could not be solved due to memory limitations (exceeding 18 GB, indicated as *OOM*) or time constraints (indicated as *TL*). In many such cases, our UB aligns with the value obtained from our initialization heuristic for the RMP, highlighting the competitiveness of this approach, even for the largest instances. For instance sizes of up to 500 orders, the two BPC-based heuristics consistently outperform the CGH, even when post-optimization is applied. However, for very large instances involving 1,000 orders or more, the performance of our BPC-based methods declines. In these cases, Briant *et al.* (2020) achieve stronger UBs, primarily due to their dedicated post-optimization process, which offers a notable advantage over our approaches for the largest instances.

3.6 Conclusions

In this study, we exploited a *branch-price-and-cut* (BPC) approach to address the *order batching problem* (OBP) in warehouses with multiple blocks, a complex logistical challenge crucial for enhancing operational efficiency in large-scale facilities. To the best of our knowledge, no BPC method has previously been applied to solve the OBP in multi-block warehouse environments. Moreover, no exact solution approach has yet been proposed to tackle the multi-block OBP for various heuristic routing strategies.

Our approach extends the BPC framework developed by Wahlen and Gschwind (2023) for the single-block OBP, demonstrating its adaptability to accommodate various warehouse layouts and routing strategies, given a monotone distance function. Specifically, we evaluated the monotonicity of six established routing strategies: optimal, no-reversal, aisle-by-aisle, traversal (or s-shape), combined, and largest gap, two of which were modified for this evaluation. We proved that all these strategies, except for largest gap, are monotone in a rectangular warehouse with parallel aisles and an arbitrary number of blocks. Through extensive computational experiments using publicly available datasets, the BPC approach outperformed existing state-of-the-art methods, achieving a higher number of optimally solved instances and significantly reducing computation times. Notably, we successfully solved instances with up to 80 orders to proven optimality for all five monotone routing strategies. Our experimental results indicate that, among the heuristic routing strategies, the combined routing strategy offers an effective balance between computation time and solution quality for the BPC approach. Furthermore, BPC-based heuristics demonstrated superior performance over existing specialized methods for instances involving up to 500 orders, improving many of the best-known solutions while maintaining short computation times. Performance declines were observed only for instances involving 1,000 orders or more, where more refined strategies, such as the post-optimization method by Briant *et al.* (2020), proved advantageous. Overall, these findings highlight the potential of the BPC – both in its exact form and heuristic variants – to significantly enhance warehouse operations by optimizing order batching in multi-block environments.

Future research could benefit from expanding the BPC approach to a wider variety of warehouse configurations. This exploration may include warehouses that diverge from the traditional rectangular grid with parallel aisles and cross aisles (e.g., Çelik and Süral 2014), implement scattered storage policies (e.g., Heßler and Irnich 2024, Lüke *et al.* 2024), feature high-level racks that require picker elevation to access storage locations (e.g., van Gils *et al.* 2019), or facilitate the decoupling of pickers and trolleys (e.g., Goeke and Schneider 2021). Investigating these areas could help generalize our findings and yield deeper insights into the practical application of the BPC method across various warehouse operations. Moreover,

the monotonicity analysis and proofs presented in this paper can be leveraged in other OBP solution techniques that either rely on or benefit from monotone routing strategies. This includes, for example, methods based on (extended) set-partitioning formulations, or cutting techniques that account for the distances required to fulfill subsets of orders.

Bibliography

Bahçeci, U. and Öncan, T. (2022). An evaluation of several combinations of routing and storage location assignment policies for the order batching problem. *International Journal of Production Research*, **60**(19), 5892–5911.

Boysen, N., Briskorn, D., and Emde, S. (2017). Sequencing of picking orders in mobile rack warehouses. *European Journal of Operational Research*, **259**(1), 293–307.

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A.-L., and Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, **285**(2), 497–512.

Bué, M., Cattaruzza, D., Ogier, M., and Semet, F. (2019). A two-phase approach for an integrated order batching and picker routing problem. In M. Dell'Amico, M. Gaudioso, and G. Stecca, editors, *A View of Operations Research Applications in Italy, 2018*, volume 2, pages 3–18. Springer International Publishing, Cham, Switzerland.

Burkard, R. E., Deineko, V. G., van Dal, R., van der Veen, J. A., and Woeginger, G. J. (1998). Well-solvable special cases of the traveling salesman problem: A survey. *SIAM Review*, **40**(3), 496–546.

Cambazard, H. and Catusse, N. (2018). Fixed-parameter algorithms for rectilinear Steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, **270**(2), 419–429.

Çelik, M. and Süral, H. (2014). Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE Transactions*, **46**(3), 283–300.

de Koster, R., Roodbergen, K. J., and van Voorden, R. (1999). Reduction of walking time in the distribution center of De Bijenkorf. In M. Speranza and P. Stähly, editors, *New Trends in Distribution Logistics*, pages 215–234. Springer Berlin.

de Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, **182**(2), 481–501.

Desrosiers, J., Lübbecke, M., Desaulniers, G., and Gauthier, J. B. (2024). *Branch-and-Price*. Les Cahiers du GERAD, Montréal, Canada.

Gademann, N. and van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, **37**(1), 63–75.

Gademann, N., van den Berg, J., and van der Hoff, H. (2001). An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, **33**(5), 385–398.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 436–451.

Goetschalckx, M. and Ratliff, H. (1988). Order picking in an aisle. *IIE Transactions*, **20**(1), 53–62.

Grosse, E. H., Glock, C. H., Ballester-Ripoll, R., *et al.* (2014). A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. *International Journal of Operations and Quantitative Management*, **20**(2), 65–83.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, **25**(4), 76–87.

Henn, S., Koch, S., and Wäscher, G. (2012). Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini, editor, *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London.

Heßler, K. and Irnich, S. (2022). A note on the linearity of Ratliff and Rosenthal's algorithm for optimal picker routing. *Operations Research Letters*, **50**(2), 155–159.

Heßler, K. and Irnich, S. (2024). Exact solution of the single-picker routing problem with scattered storage. *INFORMS Journal on Computing*. Advance online publication.

Hong, S. and Kim, Y. (2017). A route-selecting order batching model with the s-shape routes in a parallel-aisle order picking system. *European Journal of Operational Research*, **257**(1), 185–196.

Hong, S., Johnson, A. L., and Peters, B. A. (2012). Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, **44**(2), 88–106.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer Science & Business Media, Boston.

Kulak, O., Sahin, Y., and Taner, M. E. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, **24**, 52–80.

Lüke, L., Heßler, K., and Irnich, S. (2024). The single picker routing problem with scattered storage: modeling and evaluation of routing and storage policies. *OR Spectrum*, **46**, 909–951.

Menéndez, B., Bustillo, M., Pardo, E. G., and Duarte, A. (2017). General variable neighborhood search for the order batching and sequencing problem. *European Journal of Operational Research*, **263**(1), 82–93.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, **100**, 117–127.

Pardo, E. G., Gil-Borrás, S., Alonso-Ayuso, A., and Duarte, A. (2024). Order batching problems: Taxonomy and literature review. *European Journal of Operational Research*, **313**(1), 1–24.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. *Decision Sciences Institute Proceedings*, **3**, 1614–1616.

Petersen, C. G. and Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, **92**(1), 11–19.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

Richards, G. (2017). *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. Kogan Page, London, 3rd edition.

Roodbergen, K. J. and de Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, **39**(9), 1865–1883.

Roodbergen, K. J. and de Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, **133**(1), 32–43.

Ryan, D. M. and Foster, B. A. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 269–280. North-Holland Publishing Company, Amsterdam.

Schiffer, M., Boysen, N., Klein, P. S., Laporte, G., and Pavone, M. (2022). Optimal picking policies in e-commerce warehouses. *Management Science*, **68**(10), 7497–7517.

Scholz, A. and Wäscher, G. (2017). Order batching and picker routing in manual order picking systems: The benefits of integrated routing. *Central European Journal of Operations Research*, **25**(2), 491–520.

Thia, F. (2008). MySQL Foodmart Database. Pentaho Wiki. `http://pentaho.dlpage.phi-integration.com/mondrian/mysql-foodmart-database`, May 8.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons, Hoboken, NJ, 4th edition.

Valle, C. A. and Beasley, J. E. (2020). Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, **284**(2), 460–484.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2016). Modelling and solving the joint order batching and picker routing problem in inventories. In R. Cerulli, S. Fujishige, and A. R. Mahjoub, editors, *Combinatorial Optimization*, volume 9849 of *Lecture Notes in Computer Science*, pages 81–97, Cham, Switzerland. Springer International Publishing.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, **262**(3), 817–834.

van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, **277**(3), 814–830.

Vaughan, T. (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, **37**(4), 881–897.

Wahlen, J. and Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, **57**(3), 756–777.

Won, J. and Olafsson, S. (2005). Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, **43**(7), 1427–1442.

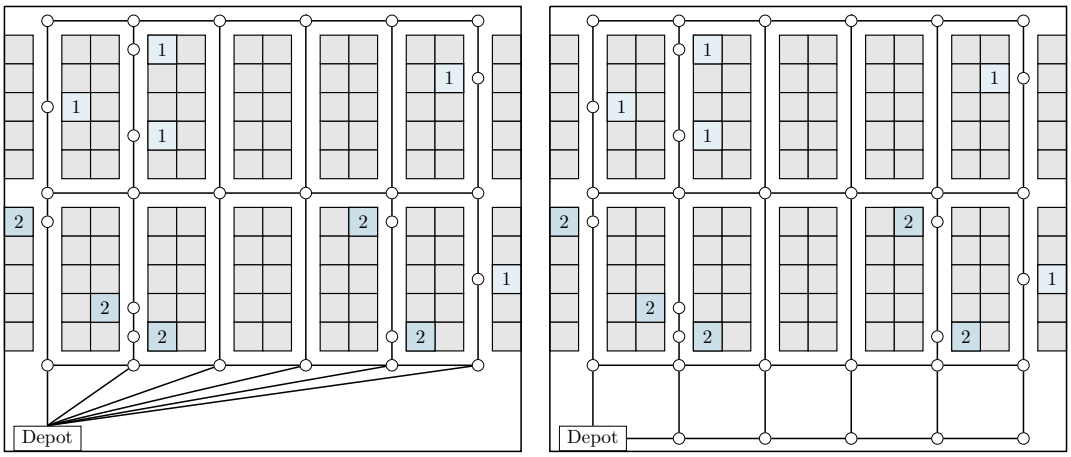
Zhang, K. and Gao, C. (2023). Improved formulations of the joint order batching and picker routing problem. *International Journal of Production Research*, **61**(21), 7386–7409.

Žulj, I., Kramer, S., and Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, **264**(2), 653–664.

Appendix

3.A `Foodmart` State Space

In this paper, the SPRP for optimal routing is solved following the DP method outlined by Pansart *et al.* (2018) for rectangular warehouses. In the case of Euclidean distances between the depot and its nearest cross aisle, as given in the `Foodmart` instances and illustrated in Figure 3.5a for batch $b = \{1, 2\}$ in a warehouse with $H = 2$ blocks, this approach cannot be seamlessly applied. To address this challenge, we adapt the original warehouse structure by incorporating an additional artificial block, referred to as block 3 (i.e., $H + 1$). In block 3, feasible sub-aisle transitions are limited to single traversal, double traversal, and void moves. Each traversal within this block is assigned the initial Euclidean distance from the corresponding aisle to the depot. Notably, the horizontal distances along the artificial cross aisle 3 at the level of the depot are assumed to be zero. However, the introduction of an additional block, as depicted in Figure 3.5b, considerably increases the computational complexity of the SPRP compared to the conventional layout, where the depot is positioned directly in the foremost cross aisle (Cambazard and Catusse 2018).



(a) Original structure **(b)** Transformed graph

Figure 3.5: Warehouse layout of `Foodmart` instances

The equivalent classes or *states* of partial tour subgraphs emerging in the DP can be characterized by features of the four cross aisle vertices in a specific aisle. A comprehensive list of feasible states for the `Foodmart` instances is presented in Table 3.12, with the assumption that at least one storage location must be visited. Each state is defined by the associated degree parities of the four cross aisle vertices, their connectivity and, if crucial, the composition of the connected

0000,1	E0UU,1	EEE0,2,01-2	EUUE,2,012-3
000E,1	UEU0,1	EEE0,2,0-12	EUUE,2,0-123
00E0,1	UU0E,1	EEE0,2,02-1	EUUE,2,03-12
0E00,1	UUE0,1	EEEE,2,012-3	EEUU,2,01-23
E000,1	EU0U,1	EEEE,2,01-23	EEUU,2,0-123
00EE,1	EUU0,1	EEEE,2,0-123	EEUU,2,023-1
0E0E,1	EEUU,1	EEEE,2,013-2	UUEE,2,012-3
E00E,1	UEEU,1	EEEE,2,023-1	UUEE,2,01-23
0EE0,1	UEUE,1	EEEE,2,03-12	UUEE,2,013-2
E0E0,1	UUEE,1	0EUU,2,1-23	UUUU,2,01-23
EE00,1	EUEU,1	0UEU,2,13-2	UUUU,2,03-12
00UU,1	EUUE,1	0UUE,2,12-3	0EEE,3
0U0U,1	UUUU,1	U0EU,2,03-2	E0EE,3
U00U,1	00EE,2	U0UE,2,02-3	EE0E,3
0UU0,1	0E0E,2	UE0U,2,03-1	EEE0,3
U0U0,1	E00E,2	E0UU,2,0-23	EEEE,3,01-2-3
UU00,1	0EE0,2	UEU0,2,02-1	EEEE,3,0-12-3
0EEE,1	E0E0,2	UU0E,2,01-3	EEEE,3,0-1-23
E0EE,1	EE00,2	UUE0,2,01-2	EEEE,3,0-13-2
EE0E,1	0EEE,2,12-3	EU0U,2,0-13	EEEE,3,02-1-3
EEE0,1	0EEE,2,1-23	EUU0,2,0-12	EEEE,3,03-1-2
EEEE,1	0EEE,2,13-2	UEEU,2,013-2	UEEU,3,03-1-2
0UUE,1	E0EE,2,02-3	UEEU,2,023-1	UEUE,3,02-1-3
0UEU,1	E0EE,2,0-23	UEEU,2,03-12	UUEE,3,01-2-3
0EUU,1	E0EE,2,03-2	UEUE,2,012-3	EEUU,3,0-1-23
U0EU,1	EE0E,2,01-3	UEUE,2,023-1	EUEU,3,0-13-2
U0UE,1	EE0E,2,0-13	EUEU,2,0-123	EUUE,3,0-12-3
UE0U,1	EE0E,2,03-1	EUEU,2,013-2	EEEE,4

Table 3.12: List of feasible states of the `Foodmart` instances

components. The degree parities are each classified as either even (E), odd (U), or zero (0) for cross aisle 0 to 3. The connectivity is represented by the total number of components, which can range up to four. If the connectivity is not uniquely defined by the preceding information, the assignment of the cross aisles to the components is specified explicitly. The set of final states consists of all states that feature a single connected component and exhibit only even or zero degree parities.

For instance, the state designation *EEEE,2,012-3* denotes that all four cross aisle vertices have an even number of incident edges (*EEEE*), and there are two connected components (*2*). In this configuration, the upper three cross aisle vertices are connected, whereas the depot's cross aisle vertex forms a separate component and is not connected with the others (*012-3*).

3.B Instance-by-Instance Comparison

In this section, we present the instance-by-instance results of our multi-block BPC approach, which are discussed in aggregate in Section 3.5.

Tables 3.13 and 3.14 provide a detailed comparison of our approach with the B&C methods proposed by Valle *et al.* (2017) and Zhang and Gao (2023), respectively. The tables provide the computation time in seconds ($t[s]$), the best integer solution (UB), and the percentage deviation from the optimal solution (Gp) for each of the approaches.

Table 3.15 depicts the comparison of our BPC-based heuristics for optimal routing with the heuristic approaches DAA (Valle and Beasley 2020) and CGH (Briant *et al.* 2020). It presents the best-known (in this case, optimal) solution value (BKS) for each instance, along with the computation time in seconds ($t[s]$) and the optimality gap (Gp) for each approach.

Larger `Foodmart` instances (more orders and an enlarged warehouse) are considered in Table 3.16 to compare the performance of our heuristics for both the optimal and the combined strategy against the DAA proposed by Valle and Beasley (2020). The provided gaps are based on the overall BKS for each instance.

3.C Detailed Computational Results

In this section, we report detailed computational results of our BPC algorithm and BPC-based heuristics for the five monotone routing strategies optimal, no-reversal, aisle-by-aisle, combined, and traversal.

Tables 3.17–3.21 provide detailed results for the exact BPC and the two benchmark sets `Foodmart` and `Scholz&Wäscher`, aggregated by number of days Δ and number of orders n (`Foodmart`) or capacity Q and n (`Scholz&Wäscher`). The tables report the number of instances solved to optimality within the time limit of one hour (Opt), the average solution time in seconds ($t[s]$), the average time for solving the LP relaxation in seconds (t^{LP}), the average optimality gap with respect to the BKS of the LP relaxation (Gp), the average optimality gap with respect to the BKS before the first node resulting from a Ryan-and-Foster branching is solved (Gp^{RF}), the average number of B&B nodes solved (Nds), and the average number of CCs (CC) and SRCs (SRC) added.

Table 3.22 presents a comparison of our BPC-based heuristics for the `Foodmart` instances, aggregated by the number of orders n. It provides the average computation time in seconds ($t[s]$) and the average gap with respect to the BKS per instance across all routing strategies (Gp).

		VBC (2017)			Our method		
Δ	n	t[s]	UB	Gp	t[s]	UB	Gp
5	5	1.3	348.6	0.00	0.0	348.6	0.00
	6	0.6	364.8	0.00	0.0	364.8	0.00
	7	1.7	374.8	0.00	0.0	374.8	0.00
	8	7.2	503.8	0.00	0.0	503.8	0.00
	9	8.0	539.6	0.00	0.0	539.6	0.00
	10	8.1	581.4	0.00	0.1	581.4	0.00
	11	13.0	613.5	0.00	0.2	613.5	0.00
	12	22.5	621.4	0.00	0.3	621.4	0.00
	13	14.8	623.4	0.00	0.3	623.4	0.00
	14	46.9	639.3	0.00	0.6	639.3	0.00
	15	37.3	653.4	0.00	0.4	653.4	0.00
	20	3,035.2	870.4	0.00	3.3	870.4	0.00
	25	21,600.0	1,123.5	2.52	9.8	1,095.9	0.00
	30	21,600.0	1,263.5	7.63	46.0	1,173.9	0.00
Subtotal		3,314.0	651.5	0.73	4.4	643.2	0.00
10	5	0.4	371.1	0.00	0.0	371.1	0.00
	6	1.5	377.1	0.00	0.0	377.1	0.00
	7	6.7	549.8	0.00	0.0	549.8	0.00
	8	7.1	584.2	0.00	0.0	584.2	0.00
	9	55.0	637.4	0.00	0.0	637.4	0.00
	10	63.9	661.8	0.00	0.1	661.8	0.00
	11	655.0	699.8	0.00	0.3	699.8	0.00
	12	39.5	707.7	0.00	0.3	707.7	0.00
	13	497.9	725.7	0.00	0.8	725.7	0.00
	14	3,889.3	727.8	0.00	0.5	727.8	0.00
	15	664.6	882.6	0.00	0.4	882.6	0.00
	20	10,923.4	992.4	0.00	5.3	992.4	0.00
	25	21,600.0	1,266.1	6.42	3.9	1,189.7	0.00
	30	21,600.0	1,345.6	5.79	21.9	1,272.0	0.00
Subtotal		4,286.0	752.1	0.87	2.4	741.4	0.00
20	5	7.7	573.8	0.00	0.0	573.8	0.00
	6	20.6	656.2	0.00	0.0	656.2	0.00
	7	14.0	689.8	0.00	0.0	689.8	0.00
	8	26.1	697.8	0.00	0.0	697.8	0.00
	9	115.6	727.7	0.00	0.0	727.7	0.00
	10	179.1	920.5	0.00	0.0	920.5	0.00
	11	309.0	980.5	0.00	0.1	980.5	0.00
	12	871.0	1,004.3	0.00	0.1	1,004.3	0.00
	13	852.7	1,009.1	0.00	0.2	1,009.1	0.00
	14	506.1	1,011.1	0.00	0.1	1,011.1	0.00
	15	2,308.9	1,028.7	0.00	0.2	1,028.7	0.00
	20	21,600.0	1,373.5	2.98	0.6	1,333.7	0.00
	25	21,600.0	1,692.3	4.47	7.5	1,619.9	0.00
	30	21,600.0	1,944.7	4.81	6.8	1,855.5	0.00
Subtotal		5,000.8	1,022.1	0.88	1.1	1,007.8	0.00
Total		4,200.3	808.6	0.82	2.6	797.4	0.00

Table 3.13: Comparison of our BPC algorithm with the B&C approach of Valle *et al.* (2017) on a subset of the `Foodmart` instances for the optimal routing strategy

		Z&G (2023)			Our method		
Δ	n	t[s]	UB	Gp	t[s]	UB	Gp
5	5	0.2	346.0	0.00	0.0	346.0	0.00
	10	1.5	578.0	0.00	0.0	578.0	0.00
	15	7.8	650.0	0.00	0.1	650.0	0.00
	16	37.0	766.0	0.00	0.1	766.0	0.00
	17	30.0	802.0	0.00	0.1	802.0	0.00
	18	81.0	840.0	0.00	0.2	840.0	0.00
	19	135.0	856.0	0.00	0.4	856.0	0.00
	20	86.0	864.0	0.00	0.5	864.0	0.00
	21	136.0	892.0	0.00	3.3	892.0	0.00
	22	171.0	892.0	0.00	1.2	892.0	0.00
	23	290.0	908.0	0.00	0.6	908.0	0.00
	24	2,400.0	1,059.0	0.47	2.4	1,054.0	0.00
	25	2,400.0	1,102.0	1.47	2.9	1,086.0	0.00
	30	2,400.0	1,206.0	3.79	8.9	1,162.0	0.00
Subtotal		584.0	840.1	0.41	1.5	835.4	0.00
10	5	0.1	368.0	0.00	0.0	368.0	0.00
	10	6.3	656.0	0.00	0.0	656.0	0.00
	15	59.0	874.0	0.00	0.1	874.0	0.00
	16	65.0	926.0	0.00	0.1	926.0	0.00
	17	123.0	960.0	0.00	0.6	960.0	0.00
	18	106.0	970.0	0.00	0.6	970.0	0.00
	19	166.0	978.0	0.00	1.4	978.0	0.00
	20	209.0	984.0	0.00	1.4	984.0	0.00
	21	143.0	990.0	0.00	1.6	990.0	0.00
	22	180.0	1,000.0	0.00	0.5	1,000.0	0.00
	23	2,400.0	1,140.0	1.06	0.4	1,128.0	0.00
	24	2,400.0	1,162.0	0.00	0.6	1,162.0	0.00
	25	2,400.0	1,220.0	3.57	0.6	1,178.0	0.00
	30	2,400.0	1,320.0	4.76	3.7	1,260.0	0.00
Subtotal		761.2	967.7	0.67	0.8	959.6	0.00
20	5	1.0	570.0	0.00	0.0	570.0	0.00
	10	25.0	912.0	0.00	0.0	912.0	0.00
	15	58.0	1,022.0	0.00	0.0	1,022.0	0.00
	16	745.0	1,200.0	0.00	0.0	1,200.0	0.00
	17	2,059.0	1,250.0	0.00	0.1	1,250.0	0.00
	18	1,166.0	1,288.0	0.00	0.1	1,288.0	0.00
	19	2,400.0	1,326.0	1.69	0.1	1,304.0	0.00
	20	2,400.0	1,340.0	1.36	0.1	1,322.0	0.00
	21	2,400.0	1,542.0	3.07	1.4	1,496.0	0.00
	22	2,400.0	1,578.0	4.37	1.0	1,512.0	0.00
	23	2,400.0	1,624.0	4.37	0.2	1,556.0	0.00
	24	2,400.0	1,640.0	3.93	0.3	1,578.0	0.00
	25	2,400.0	1,648.0	2.62	4.3	1,606.0	0.00
	30	2,400.0	1,900.0	3.37	1.2	1,838.0	0.00
Subtotal		1,661.0	1,345.7	1.77	0.6	1,318.1	0.00
Total		1,002.1	1,051.2	0.95	1.0	1,037.7	0.00

Table 3.14: Comparison of our BPC algorithm with the approach of Zhang and Gao (2023) on a subset of the modified `Foodmart` instances for the optimal routing strategy

			DAA		CGH		BPC-DF		SC	
Δ	n	BKS	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp
5	5	348.59	1.0	0.00	1.2	0.00	0.0	0.00	0.0	0.00
	6	364.81	0.6	0.00	1.1	0.00	0.0	0.00	0.0	0.00
	7	374.81	1.4	0.00	1.8	0.00	0.0	0.00	0.0	0.00
	8	503.75	1.1	0.00	3.2	0.00	0.0	0.00	0.0	0.00
	9	539.61	1.5	6.30	4.3	0.00	0.0	0.00	0.0	0.00
	10	581.42	1.7	2.44	10.9	0.00	0.1	0.00	0.1	0.00
	11	613.51	1.0	0.70	53.7	0.00	0.2	0.00	0.1	0.65
	12	621.35	1.1	0.06	45.3	0.00	0.3	0.00	0.3	0.00
	13	623.39	1.6	2.95	75.9	0.00	0.3	0.00	0.3	0.00
	14	639.31	3.4	1.64	116.2	0.01	0.6	0.00	0.5	0.01
	15	653.42	1.2	3.66	166.5	0.00	0.4	0.00	0.4	0.00
	20	870.37	2.9	3.22	1,596.8	1.79	3.3	0.00	3.4	0.00
	25	1,095.89	32.2	3.26	7,246.5	1.14	9.8	0.00	8.5	0.71
	30	1,173.86	386.4	3.92	7,304.0	5.67	41.2	0.00	34.7	0.00
Subtotal			31.2	2.01	1,187.7	0.62	4.0	0.00	3.5	0.10
10	5	371.09	0.5	0.00	0.9	0.00	0.0	0.00	0.0	0.00
	6	377.09	1.4	0.00	1.2	0.00	0.0	0.00	0.0	0.00
	7	549.80	1.8	2.87	1.8	0.00	0.0	0.00	0.0	0.00
	8	584.18	1.6	5.82	1.4	0.00	0.0	0.00	0.0	0.00
	9	637.37	3.9	0.00	7.2	0.00	0.0	0.00	0.0	0.00
	10	661.80	3.6	5.12	16.2	0.00	0.1	0.00	0.1	0.00
	11	699.80	2.8	2.84	114.8	0.86	0.4	0.00	0.2	1.98
	12	707.71	2.8	2.27	183.0	0.00	0.3	0.00	0.3	0.00
	13	725.71	3.8	0.28	124.1	0.84	0.8	0.00	0.3	0.28
	14	727.80	6.8	2.95	182.6	0.54	0.4	0.00	0.4	0.00
	15	882.61	5.6	2.93	189.2	0.22	0.4	0.00	0.4	0.00
	20	992.40	3.9	1.50	955.8	1.26	5.0	0.00	2.3	1.26
	25	1,189.70	14.2	3.70	2,677.6	1.18	4.4	0.00	4.0	0.00
	30	1,271.98	142.4	3.31	7,305.8	0.16	16.2	0.00	15.4	0.14
Subtotal			13.9	2.40	840.1	0.36	2.0	0.00	1.7	0.26
20	5	573.77	2.1	0.00	0.7	0.00	0.0	0.00	0.0	0.00
	6	656.18	1.1	1.83	1.2	0.00	0.0	0.00	0.0	0.00
	7	689.80	1.5	4.33	8.0	0.00	0.0	0.00	0.0	0.00
	8	697.80	2.3	5.72	10.5	0.00	0.0	0.00	0.0	0.00
	9	727.71	3.5	1.92	41.2	0.23	0.0	0.00	0.0	0.00
	10	920.52	2.9	2.82	37.9	0.42	0.0	0.00	0.0	0.00
	11	980.51	3.7	2.45	11.7	0.00	0.1	0.00	0.1	0.00
	12	1,004.32	4.2	3.00	41.9	0.00	0.1	0.00	0.1	0.99
	13	1,009.08	3.9	1.58	82.7	0.11	0.2	0.00	0.2	2.29
	14	1,011.08	4.1	5.65	70.6	0.00	0.2	0.00	0.1	0.00
	15	1,028.70	5.0	3.87	87.8	0.00	0.2	0.00	0.2	0.00
	20	1,333.70	28.2	4.02	619.6	0.18	0.6	0.00	0.6	0.00
	25	1,619.88	225.2	3.24	3,288.5	0.08	5.0	0.00	1.5	1.44
	30	1,855.50	10,644.5	3.43	7,271.4	1.28	5.4	0.00	5.5	0.00
Subtotal			780.9	3.13	826.7	0.16	0.8	0.00	0.6	0.34
Total			275.3	2.51	951.5	0.38	2.3	0.00	1.9	0.23

Table 3.15: Comparison of our heuristics BPC-DF and SC for the optimal routing strategy to the heuristic approaches of Valle and Beasley (2020) and Briant *et al.* (2020) on a subset of the `Foodmart` instances

						BPC-DF				SC			
				V&B (2020)		Optimal		Combined		Optimal		Combined	
Δ	n	S	BKS	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp
5	25	8	1,095.89	32.2	3.26	9.8	0.00	0.3	0.89	8.5	0.71	0.4	0.89
		16	1,325.51	1,664.2	2.25	33.0	0.00	2.5	2.59	32.8	0.00	1.7	2.59
	30	8	1,173.86	386.4	3.92	41.2	0.00	16.2	2.06	34.7	0.00	1.3	2.40
		16	1,413.70	21,637.5	5.35	100.2	0.00	17.1	2.42	101.1	0.00	8.2	2.42
	50	8	1,878.19	21,602.4	6.38	684.2	0.00	89.0	1.77	307.9	0.56	8.9	2.62
		16	2,254.19	21,647.4	6.65	3,600.0	0.00	3,512.7	1.62	1,177.6	0.86	142.9	3.75
	75	8	2,590.67	21,603.9	9.94	3,600.0	0.00	3,600.0	1.44	1,226.5	0.77	399.4	2.75
		16	3,127.36	21,665.7	9.14	3,600.0	11.70	3,600.0	0.00	3,600.0	11.70	1,489.1	1.15
Subtotal				13,780.0	5.86	1,458.6	1.46	1,354.7	1.60	811.1	1.83	256.5	2.32
10	25	8	1,189.70	14.2	3.70	4.4	0.00	0.2	1.04	4.0	0.00	0.3	1.04
		16	1,441.98	4,505.4	2.50	27.4	0.00	1.0	1.55	22.5	0.83	0.9	1.69
	30	8	1,271.98	142.4	3.31	16.2	0.00	0.5	0.97	15.4	0.14	0.4	0.97
		16	1,524.36	21,698.1	1.55	60.8	0.00	2.7	2.36	62.5	0.00	2.6	2.87
	50	8	2,008.71	21,607.5	5.61	2,261.9	0.00	1,508.7	1.33	187.5	1.02	25.2	2.82
		16	2,414.87	21,731.8	6.21	2,616.9	0.00	634.4	2.01	959.4	0.48	70.0	4.41
	75	8	2,789.75	21,606.9	8.33	3,600.0	0.00	3,600.0	1.09	1,452.4	1.08	707.6	2.74
		16	3,415.57	21,679.9	5.21	3,600.0	11.02	3,600.0	0.00	3,600.0	11.02	1,212.3	1.57
Subtotal				14,123.3	4.55	1,523.5	1.38	1,168.4	1.29	788.0	1.82	252.4	2.26
20	25	8	1,619.88	225.2	3.24	5.0	0.00	1.0	1.56	1.5	1.44	0.2	2.32
		16	1,894.13	21,673.6	3.81	10.0	0.00	0.8	1.93	6.9	0.71	0.5	2.15
	30	8	1,855.50	10,644.5	3.43	5.4	0.00	0.4	2.43	5.5	0.00	0.4	2.43
		16	2,199.59	21,736.3	3.46	27.5	0.00	1.2	1.38	23.1	0.00	1.1	1.65
	50	8	2,539.59	21,608.3	7.47	163.1	0.00	5.5	1.05	106.4	0.55	3.3	1.52
		16	3,027.78	21,715.2	6.68	1,272.5	0.00	79.6	1.53	533.4	0.20	14.2	1.99
	75	8	3,520.49	21,610.5	7.55	3,104.4	0.00	3,600.0	1.57	468.2	0.35	234.1	2.15
		16	4,258.04	21,767.5	6.09	3,600.0	9.88	3,600.0	0.00	3,600.0	9.88	2,167.8	0.32
Subtotal				17,622.6	5.21	1,023.5	1.24	911.1	1.43	593.1	1.64	302.7	1.82
Total				15,175.3	5.21	1,335.2	1.36	1,144.7	1.44	730.7	1.76	270.5	2.13

Table 3.16: Comparison of our heuristics BPC-DF and SC for the routing strategies optimal and combined to the heuristic approaches of Valle and Beasley (2020) on a subset of large `Foodmart` instances

Panel A: `Foodmart` instances										
Δ	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
5	$(0,10]$	12	12	0.0	0.0	1.34	1.34	1	2	0
	$(10,20]$	20	20	2.3	1.9	2.73	2.73	2	27	4
	$(20,30]$	20	20	40.8	28.2	1.62	0.93	3	34	38
	$(30,40]$	20	20	313.2	135.7	1.59	0.60	45	28	71
	$(40,50]$	20	19	1,110.4	517.9	1.94	0.50	77	41	87
	$(50,75]$	2	0	3,600.0	535.5	2.59	1.18	274	48	48
Subtotal		94	91	388.7	156.9	1.90	1.21	33	29	44
10	$(0,10]$	12	12	0.0	0.0	1.93	1.93	1	5	0
	$(10,20]$	20	20	4.8	1.5	3.27	2.92	3	48	21
	$(20,30]$	20	20	33.8	17.5	1.64	0.81	4	42	42
	$(30,40]$	20	20	221.9	103.6	1.50	0.41	28	30	76
	$(40,50]$	20	20	728.8	355.9	1.32	0.43	93	39	86
	$(50,75]$	2	0	3,600.0	399.5	1.42	0.86	399	35	53
Subtotal		94	92	287.1	110.3	1.92	1.24	36	35	49
20	$(0,10]$	12	12	0.0	0.0	1.37	1.37	1	6	2
	$(10,20]$	20	20	0.5	0.3	1.19	1.19	1	11	6
	$(20,30]$	20	20	7.9	5.7	0.91	0.50	5	19	29
	$(30,40]$	20	20	84.0	36.1	1.11	0.25	27	38	57
	$(40,50]$	20	20	521.2	159.6	0.93	0.36	134	39	65
	$(50,75]$	2	1	2,639.8	205.3	0.59	0.43	324	5	33
Subtotal		94	93	186.7	47.3	1.07	0.67	43	24	34
Total		282	276	287.5	104.8	1.63	1.04	37.1	29.3	42.3
Panel B: `Scholz&Wäscher` instances										
Q	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
30	20	100	100	0.0	0.0	0.26	0.23	2	0	0
	40	100	100	0.0	0.0	0.25	0.22	2	1	0
	60	100	100	0.1	0.0	0.15	0.12	2	1	0
	80	100	100	0.1	0.1	0.10	0.09	2	1	0
Subtotal		400	400	0.0	0.0	0.19	0.17	2	1	0
45	20	100	100	0.1	0.0	0.51	0.29	7	4	5
	40	100	100	0.8	0.1	0.35	0.16	17	6	14
	60	100	100	6.5	0.6	0.27	0.12	123	6	20
	80	100	100	17.2	1.5	0.20	0.11	148	8	20
Subtotal		400	400	6.2	0.6	0.33	0.17	74	6	15
60	20	100	100	0.3	0.1	0.81	0.44	6	11	13
	40	100	100	7.7	1.1	0.59	0.22	86	14	31
	60	100	97	234.9	5.8	0.62	0.40	2,377	15	36
	80	100	89	722.6	17.0	1.24	1.07	4,088	16	40
Subtotal		400	386	241.4	6.0	0.82	0.53	1,639	14	30
75	20	100	100	1.0	0.2	1.46	0.71	8	23	22
	40	100	100	30.4	5.5	0.89	0.27	66	35	51
	60	100	96	638.5	39.9	1.16	0.72	1,527	35	59
	80	100	54	2,251.1	139.2	4.42	4.16	2,557	31	61
Subtotal		400	350	730.2	46.2	1.98	1.47	1,040	31	48
Total		1,600	1,536	244.5	13.2	0.83	0.59	689	13	23

Table 3.17: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy optimal

Panel A: `Foodmart` instances										
Δ	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
5	$(0, 10]$	12	12	0.0	0.0	1.65	1.65	1	2	0
	$(10, 20]$	20	20	4.5	1.2	3.31	2.73	6	17	24
	$(20, 30]$	20	20	79.5	11.0	2.36	1.08	41	37	59
	$(30, 40]$	20	20	244.5	49.9	2.01	0.84	192	24	78
	$(40, 50]$	20	17	1,351.0	143.7	3.37	1.94	1,131	38	77
	$(50, 75]$	2	0	3,600.0	816.4	20.74	19.38	1,656	52	98
Subtotal		94	89	433.9	61.2	3.00	2.03	327	26	53
10	$(0, 10]$	12	12	0.0	0.0	2.44	2.44	1	7	5
	$(10, 20]$	20	20	4.1	0.8	3.69	2.93	5	38	22
	$(20, 30]$	20	19	466.4	8.1	2.67	0.98	460	41	80
	$(30, 40]$	20	20	384.8	35.1	2.34	0.85	539	30	84
	$(40, 50]$	20	20	467.8	106.1	1.80	0.66	248	38	97
	$(50, 75]$	2	0	3,600.0	857.8	19.28	18.62	1,368	51	86
Subtotal		94	91	358.1	50.2	2.96	1.86	296	33	63
20	$(0, 10]$	12	12	0.0	0.0	2.17	2.17	1	4	0
	$(10, 20]$	20	20	1.9	0.2	1.70	1.06	43	13	22
	$(20, 30]$	20	20	5.3	2.2	1.31	0.44	15	13	40
	$(30, 40]$	20	18	513.8	11.9	1.84	0.62	3,481	45	66
	$(40, 50]$	20	15	1,216.4	37.1	1.41	0.67	4,877	34	71
	$(50, 75]$	2	0	3,600.0	330.3	7.33	7.09	4,006	9	76
Subtotal		94	85	446.3	18.0	1.76	1.02	1,876	23	44
Total		282	265	412.8	43.1	2.57	1.64	833	27	53
Panel B: `Scholz&Wäscher` instances										
Q	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
30	20	100	100	0.0	0.0	0.30	0.28	2	0	0
	40	100	100	0.0	0.0	0.19	0.16	2	2	0
	60	100	100	0.1	0.0	0.12	0.09	3	1	1
	80	100	100	0.1	0.0	0.10	0.08	2	1	1
Subtotal		400	800	0.0	0.0	0.18	0.15	2	1	0
45	20	100	100	0.1	0.0	0.66	0.38	6	4	7
	40	100	100	1.6	0.1	0.51	0.24	80	5	16
	60	100	100	16.6	0.3	0.42	0.21	581	5	20
	80	100	99	52.7	0.7	0.31	0.17	859	5	24
Subtotal		400	798	17.7	0.3	0.48	0.25	381	5	17
60	20	100	100	0.4	0.0	1.22	0.57	11	10	19
	40	100	100	15.6	0.5	0.81	0.42	377	12	33
	60	100	96	312.5	2.4	0.82	0.54	5,926	12	43
	80	100	86	931.1	6.2	1.57	1.37	9,513	14	47
Subtotal		400	764	314.9	2.3	1.11	0.73	3,957	12	35
75	20	100	100	1.3	0.1	1.90	0.67	13	18	30
	40	100	100	107.0	2.3	1.28	0.59	1,165	30	54
	60	100	84	1,177.8	12.8	2.02	1.52	7,290	30	67
	80	100	42	2,668.7	37.8	6.56	6.24	9,265	32	71
Subtotal		400	652	988.7	13.2	2.94	2.26	4,433	27	56
Total		1600	3014	330.3	4.0	1.18	0.85	2,193	11	27

Table 3.18: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy no-reversal

Panel A: `Foodmart` instances										
Δ	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
5	$(0,10]$	12	12	0.0	0.0	0.96	0.96	1	2	0
	$(10,20]$	20	20	0.7	0.2	2.48	2.33	2	27	10
	$(20,30]$	20	20	7.2	2.0	1.55	1.02	3	40	40
	$(30,40]$	20	20	45.1	11.1	1.19	0.51	19	26	62
	$(40,50]$	20	20	213.3	34.9	1.92	0.63	100	31	84
	$(50,75]$	2	1	2,902.2	274.2	10.99	9.53	1,109	52	94
Subtotal		94	93	118.4	16.1	1.88	1.28	50	28	44
10	$(0,10]$	12	12	0.0	0.0	2.95	2.95	1	8	1
	$(10,20]$	20	20	2.0	0.1	3.69	3.29	3	47	14
	$(20,30]$	20	20	13.7	1.6	1.77	0.84	5	37	46
	$(30,40]$	20	20	53.0	7.1	1.49	0.58	33	27	80
	$(40,50]$	20	20	110.2	22.3	1.24	0.42	52	31	90
	$(50,75]$	2	1	2,733.6	235.5	7.96	7.38	875	58	94
Subtotal		94	93	96.2	11.6	2.29	1.63	39	32	51
20	$(0,10]$	12	12	0.0	0.0	1.16	1.16	1	3	1
	$(10,20]$	20	20	0.2	0.0	1.42	1.32	2	17	14
	$(20,30]$	20	20	2.0	0.4	1.04	0.49	8	18	35
	$(30,40]$	20	20	27.4	2.4	1.36	0.27	25	49	65
	$(40,50]$	20	20	101.1	9.9	0.95	0.29	68	39	62
	$(50,75]$	2	0	3,600.0	126.5	11.74	11.60	3,288	14	50
Subtotal		94	92	104.4	5.4	1.41	0.90	92	27	39
Total		282	278	106.3	11.0	1.86	1.27	60	29	44
Panel B: `Scholz&Wäscher` instances										
Q	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
30	20	100	100	0.0	0.0	0.26	0.22	2	1	0
	40	100	100	0.0	0.0	0.18	0.16	2	1	0
	60	100	100	0.1	0.0	0.13	0.10	2	1	0
	80	100	100	0.1	0.0	0.09	0.08	2	1	0
Subtotal		400	800	0.0	0.0	0.16	0.14	2	1	0
45	20	100	100	0.2	0.0	0.62	0.26	16	4	7
	40	100	100	0.8	0.1	0.43	0.18	16	5	14
	60	100	100	6.8	0.6	0.38	0.17	106	6	21
	80	100	100	15.4	1.5	0.26	0.13	116	3	21
Subtotal		400	800	5.8	0.6	0.42	0.19	63	4	16
60	20	100	100	0.3	0.1	0.86	0.36	5	10	16
	40	100	100	9.5	1.0	0.66	0.29	149	12	32
	60	100	99	124.9	5.1	0.64	0.39	1,450	12	40
	80	100	92	602.3	14.6	1.15	0.95	4,232	13	46
Subtotal		400	782	184.2	5.2	0.83	0.50	1,459	12	33
75	20	100	100	0.8	0.2	1.49	0.72	6	18	25
	40	100	100	37.8	4.7	1.00	0.37	163	26	51
	60	100	95	472.7	29.7	1.18	0.73	1,485	29	61
	80	100	60	2,122.4	97.4	5.05	4.76	3,783	28	64
Subtotal		400	710	658.4	33.0	2.18	1.65	1,359	25	50
Total		1,600	3,092	212.1	9.7	0.90	0.62	721	11	25

Table 3.19: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy aisle-by-aisle

Panel A: `Foodmart` instances										
Δ	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
5	$(0, 10]$	12	12	0.0	0.0	1.30	1.30	1	2	0
	$(10, 20]$	20	20	0.2	0.0	2.53	2.33	2	20	5
	$(20, 30]$	20	20	8.6	0.5	1.82	0.75	6	36	48
	$(30, 40]$	20	20	76.9	2.9	1.47	0.70	44	26	65
	$(40, 50]$	20	20	280.4	8.2	2.07	0.66	159	37	81
	$(50, 75]$	2	0	3,600.0	63.2	16.21	14.88	1,090	55	85
Subtotal		94	92	154.5	3.8	2.19	1.43	68	27	44
10	$(0, 10]$	12	12	0.0	0.0	2.65	2.65	1	7	1
	$(10, 20]$	20	20	0.9	0.0	3.35	3.12	2	45	20
	$(20, 30]$	20	20	13.1	0.3	1.72	1.32	4	40	42
	$(30, 40]$	20	20	73.1	1.9	1.55	0.51	50	33	74
	$(40, 50]$	20	20	233.8	6.1	1.45	0.50	146	40	88
	$(50, 75]$	2	1	2,818.8	56.4	6.59	5.82	916	52	108
Subtotal		94	93	128.3	3.0	2.20	1.62	63	36	50
20	$(0, 10]$	12	12	0.0	0.0	1.04	1.04	1	3	2
	$(10, 20]$	20	20	0.1	0.0	1.21	0.93	2	14	11
	$(20, 30]$	20	20	1.4	0.1	1.06	0.26	9	15	36
	$(30, 40]$	20	20	12.9	0.6	1.09	0.29	21	31	55
	$(40, 50]$	20	20	106.0	2.5	1.06	0.38	133	33	61
	$(50, 75]$	2	1	2,684.8	28.4	6.55	6.33	1,413	9	65
Subtotal		94	93	82.7	1.3	1.21	0.66	65	20	36
Total		282	278	121.8	2.7	1.87	1.24	65	28	44
Panel B: `Scholz&Wäscher` instances										
Q	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
30	20	100	100	0.0	0.0	0.29	0.26	2	0	0
	40	100	100	0.0	0.0	0.24	0.23	2	0	0
	60	100	100	0.0	0.0	0.13	0.11	2	1	0
	80	100	100	0.1	0.0	0.12	0.09	3	1	0
Subtotal		400	800	0.0	0.0	0.19	0.17	2	1	0
45	20	100	100	0.1	0.0	0.54	0.28	8	5	5
	40	100	100	1.0	0.0	0.37	0.19	38	6	13
	60	100	100	3.6	0.1	0.31	0.13	50	7	19
	80	100	100	9.8	0.3	0.22	0.11	73	5	21
Subtotal		400	800	3.6	0.1	0.36	0.18	42	6	15
60	20	100	100	0.2	0.0	0.92	0.44	6	11	14
	40	100	100	6.1	0.2	0.60	0.27	82	12	30
	60	100	100	141.6	0.8	0.51	0.29	1,650	13	37
	80	100	93	594.6	2.2	1.10	0.92	4,324	14	41
Subtotal		400	786	185.6	0.8	0.78	0.48	1,516	12	31
75	20	100	100	0.6	0.1	1.49	0.65	6	21	24
	40	100	100	29.3	0.9	0.99	0.36	145	32	51
	60	100	90	706.4	4.8	1.79	1.34	2,568	34	60
	80	100	54	2342.8	15.1	4.96	4.68	3,887	30	62
Subtotal		400	688	769.8	5.2	2.31	1.76	1,651	29	49
Total		1,600	3,074	239.8	1.5	0.91	0.65	803	12	24

Table 3.20: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy combined

Panel A: `Foodmart` instances										
Δ	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
5	$(0, 10]$	12	12	0.0	0.0	2.01	2.01	1	3	1
	$(10, 20]$	20	20	0.4	0.0	2.90	2.71	2	24	14
	$(20, 30]$	20	20	16.0	0.3	2.28	0.66	10	42	66
	$(30, 40]$	20	20	84.4	1.2	1.93	0.74	99	28	73
	$(40, 50]$	20	20	306.2	3.4	2.31	0.91	232	39	81
	$(50, 75]$	2	0	3,600.0	23.3	17.87	16.54	2,480	54	86
Subtotal		94	92	163.2	1.5	2.64	1.68	126	30	52
10	$(0, 10]$	12	12	0.0	0.0	2.44	2.44	1	6	1
	$(10, 20]$	20	20	0.5	0.0	3.91	3.91	2	47	22
	$(20, 30]$	20	20	33.3	0.2	2.26	0.68	14	41	75
	$(30, 40]$	20	20	96.6	0.8	1.74	0.55	114	31	84
	$(40, 50]$	20	20	257.1	2.5	1.62	0.62	210	41	91
	$(50, 75]$	2	1	2,215.9	21.0	9.00	8.35	745	74	81
Subtotal		94	93	129.6	1.2	2.53	1.71	88	36	60
20	$(0, 10]$	12	12	0.0	0.0	1.65	1.65	1	4	2
	$(10, 20]$	20	20	0.2	0.0	1.74	1.31	3	14	13
	$(20, 30]$	20	20	1.3	0.1	1.13	0.25	7	19	38
	$(30, 40]$	20	20	29.7	0.3	1.51	0.39	52	43	61
	$(40, 50]$	20	20	185.3	1.0	1.25	0.53	332	34	65
	$(50, 75]$	2	1	2,814.9	10.6	6.53	6.37	3,568	8	64
Subtotal		94	93	106.0	0.5	1.55	0.87	160	24	39
Total		282	278	132.9	1.1	2.24	1.42	125	30	50
Panel B: `Scholz&Wäscher` instances										
Q	n	Inst	Opt	t[s]	t^{LP}	Gp^{LP}	Gp^{RF}	Nds	CC	SRC
30	20	100	100	0.0	0.0	0.27	0.23	2	0	0
	40	100	100	0.0	0.0	0.20	0.18	2	1	0
	60	100	100	0.0	0.0	0.11	0.10	2	1	1
	80	100	100	0.0	0.0	0.09	0.08	2	1	0
Subtotal		400	800	0.0	0.0	0.17	0.15	2	1	0
45	20	100	100	0.1	0.0	0.63	0.35	7	5	7
	40	100	100	0.6	0.0	0.46	0.19	15	7	15
	60	100	100	6.3	0.1	0.38	0.17	141	6	21
	80	100	100	20.9	0.2	0.29	0.15	374	5	23
Subtotal		400	800	7.0	0.1	0.44	0.22	134	6	16
60	20	100	100	0.4	0.0	1.15	0.56	14	12	16
	40	100	100	9.1	0.1	0.72	0.33	195	12	33
	60	100	97	195.9	0.5	0.73	0.46	3,189	18	41
	80	100	90	666.7	1.3	1.33	1.12	5,731	19	45
Subtotal		400	774	218.0	0.5	0.98	0.62	2,282	15	34
75	20	100	100	0.6	0.0	1.80	0.95	6	21	24
	40	100	100	31.1	0.5	1.12	0.44	148	30	54
	60	100	91	716.0	2.4	1.73	1.20	2,967	37	67
	80	100	55	2,187	6.9	5.00	4.66	5,323	36	71
Subtotal		400	692	733.7	2.4	2.41	1.82	2,111	31	54
Total		1,600	3,066	239.7	0.8	1.00	0.70	1,132	13	26

Table 3.21: Detailed results of our BPC algorithm for the `Foodmart` and the `Scholz&Wäscher` instances and the routing strategy traversal

			Optimal		No-reversal		Aisle-by-aisle		Combined		Traversal	
Δ	n	Inst	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp	t[s]	Gp
Panel A: BPC-DF heuristic												
5	$(0,10]$	12	0.0	0.00	0.0	10.70	0.0	6.80	0.0	2.50	0.0	9.40
	$(10,20]$	20	2.2	0.00	2.9	8.90	0.5	6.00	0.1	2.90	0.2	8.20
	$(20,30]$	20	41.5	0.00	47.4	8.20	5.4	5.60	5.1	1.90	6.0	7.30
	$(30,40]$	20	338.2	0.00	274.8	7.80	37.9	4.90	56.7	1.50	65.8	6.80
	$(40,50]$	20	1,408.3	0.00	1,708.4	8.10	335.6	4.90	502.0	1.70	340.3	6.80
	$(50,75]$	2	3,600.0	5.90	3,600.0	7.40	3,600.0	3.30	3,600.0	0.70	3,600.0	5.70
Subtotal		94	457.5	0.13	509.3	8.54	157.3	5.49	196.6	2.04	164.3	7.51
10	$(0,10]$	12	0.0	0.00	0.0	8.60	0.0	3.90	0.0	1.70	0.0	6.20
	$(10,20]$	20	4.0	0.00	2.6	6.50	0.9	3.80	0.7	1.40	0.5	4.80
	$(20,30]$	20	30.4	0.00	443.7	7.60	5.9	4.90	4.3	1.50	6.4	5.90
	$(30,40]$	20	245.4	0.00	417.4	7.20	40.9	5.00	39.2	1.50	93.7	5.60
	$(40,50]$	20	980.2	0.00	684.2	6.90	105.2	5.10	260.5	1.50	356.7	5.80
	$(50,75]$	2	3,600.0	5.50	3,600.0	5.60	3,600.0	4.10	3,600.0	0.50	3,407.4	4.10
Subtotal		94	344.7	0.12	405.9	7.22	109.1	4.59	141.4	1.48	169.8	5.58
20	$(0,10]$	12	0.0	0.00	0.0	9.30	0.0	5.10	0.0	1.80	0.0	6.40
	$(10,20]$	20	0.5	0.00	0.9	9.30	0.1	5.80	0.0	1.90	0.1	6.50
	$(20,30]$	20	7.3	0.00	4.6	8.00	1.0	5.30	0.5	1.80	0.4	6.20
	$(30,40]$	20	82.9	0.00	613.1	7.80	14.2	5.30	7.0	1.50	24.9	6.50
	$(40,50]$	20	588.5	0.00	1,466.8	7.40	132.7	4.90	88.9	1.50	289.4	6.30
	$(50,75]$	2	3,352.2	4.90	3,600.0	6.10	3,600.0	4.70	3,600.0	0.80	3,600.0	5.30
Subtotal		94	215.8	0.10	520.3	8.23	108.1	5.28	97.1	1.67	143.6	6.36
Total		282	339.3	0.12	478.5	8.00	124.8	5.12	145.0	1.73	159.2	6.48
Panel B: SC heuristic												
5	$(0,10]$	12	0.0	0.00	0.0	10.70	0.0	6.90	0.0	2.50	0.0	9.40
	$(10,20]$	20	2.2	0.20	1.5	9.20	0.3	6.20	0.1	3.00	0.1	8.60
	$(20,30]$	20	32.1	0.40	12.8	9.00	3.1	6.20	1.9	2.40	0.7	8.10
	$(30,40]$	20	159.5	0.70	56.2	9.20	14.9	5.60	8.3	2.30	5.9	8.40
	$(40,50]$	20	550.4	1.00	180.6	10.10	58.1	6.10	41.3	3.10	28.1	8.90
	$(50,75]$	2	2,413.2	6.20	1,619.0	8.60	156.1	5.50	944.2	1.90	704.7	7.30
Subtotal		94	209.7	0.62	87.9	9.53	19.6	6.13	31.1	2.66	22.4	8.59
10	$(0,10]$	12	0.0	0.00	0.0	8.60	0.0	3.90	0.0	1.90	0.0	6.20
	$(10,20]$	20	1.8	0.80	1.1	7.30	0.2	4.80	0.1	2.20	0.1	6.20
	$(20,30]$	20	18.6	0.60	9.1	7.90	2.0	5.40	0.9	2.00	0.8	7.00
	$(30,40]$	20	115.4	1.00	41.0	8.20	12.1	6.00	6.7	2.60	5.6	6.60
	$(40,50]$	20	386.4	0.80	127.1	8.30	41.3	6.20	22.0	2.60	15.0	6.80
	$(50,75]$	2	2,526.2	6.00	1,229.3	6.70	272.8	6.30	959.9	2.20	1,117.1	5.70
Subtotal		94	164.9	0.81	64.1	7.99	17.6	5.40	26.7	2.29	28.3	6.57
20	$(0,10]$	12	0.0	0.00	0.0	9.30	0.0	5.10	0.0	1.80	0.0	6.40
	$(10,20]$	20	0.5	0.20	0.3	9.40	0.1	6.00	0.0	2.00	0.0	6.60
	$(20,30]$	20	6.1	0.20	2.6	8.20	0.6	5.40	0.3	2.00	0.2	6.40
	$(30,40]$	20	38.8	0.40	14.1	8.50	3.0	5.90	2.1	1.80	1.1	7.30
	$(40,50]$	20	171.0	0.60	50.0	8.00	15.7	5.50	8.4	2.10	7.2	7.10
	$(50,75]$	2	2,034.1	5.10	386.5	7.10	1,080.2	6.10	1,200.9	1.20	236.6	5.80
Subtotal		94	89.3	0.41	22.5	8.59	27.1	5.63	27.8	1.94	6.8	6.77
Total		282	154.6	0.61	58.1	8.70	21.4	5.72	28.6	2.29	19.2	7.31

Table 3.22: Summary results of our BPC-based heuristics for the `Foodmart` instances and all routing strategies

Bibliography

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A.-L., and Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, **285**(2), 497–512.

Cambazard, H. and Catusse, N. (2018). Fixed-parameter algorithms for rectilinear Steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, **270**(2), 419–429.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, **100**, 117–127.

Valle, C. A. and Beasley, J. E. (2020). Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, **284**(2), 460–484.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, **262**(3), 817–834.

Zhang, K. and Gao, C. (2023). Improved formulations of the joint order batching and picker routing problem. *International Journal of Production Research*, **61**(21), 7386–7409.

Chapter 4

Branch-and-Price for the Set-Union Bin Packing Problem

Julia Wahlen and Timo Gschwind

Abstract

Given a set of items, each requiring a set of elements, the *set-union bin packing problem* (SUBP) consists of grouping all items into a minimum number of bins such that each item is assigned to exactly one bin and the total weight of all distinct elements required in a bin does not exceed its capacity. The SUBP is a generalization of the well-known bin packing problem, where items can share one or more elements in a non-additive fashion. In the literature, it has been addressed by various names such as pagination problem, job grouping problem, tool switching problem, or bin packing problem with overlapping items. We propose a *branch-and-price* (B&P) algorithm for solving the SUBP. For the column generation pricing problem, which is a *set-union knapsack problem* (SUKP), we present and explore alternative solution methods, namely the direct solution of an integer program with a general-purpose MIP solver and two labeling algorithms on ad hoc defined graphs. The overall best B&P variant combines an upfront greedy pricing heuristic and an item-based labeling approach without the application of any dominance. The latter is based on the representation of the pricing problem as a shortest path problem with resource constraints and relies on strong completion bounds as acceleration technique. Ryan-and-Foster branching is applied to ensure integer solutions. Extensive computational results demonstrate the effectiveness of the proposed method. Our B&P significantly outperforms the state-of-the-art IP formulations. It solves to optimality more than 10,000 instances from the literature that have only been solved heuristically before, improving the best-known solutions for more than half of the benchmark.

4.1 Introduction

The *set-union bin packing problem* (SUBP) is an extension of the well-known *bin packing problem* (BP). Given a set of items, each requiring a set of weighted elements, the SUBP consists of grouping all items into a minimum number of bins such that the total weight of all distinct elements required by the items in a bin does not exceed the bin capacity. Unlike in the classical BP, packing together two or more items into the same bin may occupy less capacity than the sum of their individual capacity consumptions. As a generalization of the BP, the SUBP is $\mathcal{NP}$-hard (Tang and Denardo 1988). It appears in numerous industries (see, e.g., Shirazi and Frizelle 2001, Crama *et al.* 2007) and related problems have been considered in different fields. We briefly discuss selected areas of application.

The *tool switching instants problem* or *machine stop minimization problem* is a classical and extensively studied problem in flexible manufacturing systems (Konak and Kulturel-Konak 2007, Konak *et al.* 2008, Marvizadeh and Choobineh 2013, Adjiashvili *et al.* 2015, Burger *et al.* 2015, Gokgur and Özpeynirci 2022). Within automated manufacturing systems, every operation or task requires a specific set of tools to be loaded into a machine and each machine is equipped with its individual tool magazine. In general, the capacity of these magazines is insufficient to accommodate the complete range of tool slots needed for all operations so that machine stops are necessary to switch tools. The objective is to minimize the number of such machine stops.

In a *job grouping* or *part grouping problem* scheduling application, several jobs have to be assigned to machines, with the aim of minimizing the number of identical machines used. Each job requires a set of specific tools, which have to be installed in the machine on which the job is to be processed. Each machine can only hold a limited number of different tools (Hirabayashi *et al.* 1984, Tang and Denardo 1988, Crama and Oerlemans 1994, Denizel 2003, Jans and Desrosiers 2013, Desrosiers *et al.* 2013).

In the field of *virtual machines* (VMs), virtualization technology enables multiple VMs to run simultaneously on a single physical server. VMs residing on the same server can share identical content storage pages, resulting in a reduction of cumulative storage requirements on server resources. The aim here is to minimize the number of servers required in order to minimize costs. This problem has been coined the *virtual machine packing problem* (Sindelar *et al.* 2011).

The equivalent *pagination problem* arises from the field of linguistics (Grange *et al.* 2018, 2023). It asks for the distribution of a given collection of tiles into the fewest number of pages. A tile is defined as a finite set of symbols from a given alphabet and each pair of sets can share zero, one or more symbols. Once some data from two tiles are packed into the same page, they do not need to be repeated twice.

In graph theory, the *k-clique covering problem* is defined on a hypergraph whose vertices can be interpreted as elements connected by hyperedges that visualize the affiliation to the items. The aim is to use the least number of cliques of size k (or subsets of maximum k vertices) such that each edge is contained in at least one such clique (Goldschmidt *et al.* 1996).

The SUBP describes the underlying optimization problem for all these applications. We study the exact solution of the SUBP using *branch-and-price* (B&P). For the related BP, *column generation* (CG) and B&P based approaches have been successfully applied (e.g., Gschwind and Irnich 2016, Wei *et al.* 2020, Baldacci *et al.* 2024). There, the CG pricing problem is a *binary knapsack problem* (KP). Analogously, solving the SUBP with B&P results in a *set-union knapsack problem* (SUKP) as pricing problem. On the theoretical side, both SUBP and SUKP are $\mathcal{NP}$-hard (Tang and Denardo 1988, Crama and Oerlemans 1994). On the practical side, a key difficulty for solution approaches to both problems is the fact that the capacity consumption of a bin is given by a function that is not separable in the comprised items. A similar challenge occurs, e.g., for *order batching problems* (OBP) in warehousing. Given a set of customer orders each comprising individual items to be picked, the OBP consists of designing a set of picking batches such that each customer order is assigned to exactly one batch, all batches satisfy the capacity restriction of the pickers, and the total distance traveled by the pickers is minimal. The travel distances of the picking batches are given by a function that is not separable in the combined orders. The recent work of Wahlen and Gschwind (2023) successfully applies B&P to solve the OBP. For the solution of the pricing problem, they propose a labeling algorithm which does not apply any dominance rules between labels (due to the non-separability of the cost function) but relies on strong completion bounds to limit the number of generated labels.

4.1.1 Contributions

The main contributions of this paper are as follows:

- We propose, to the best of our knowledge, the first B&P algorithm for the SUBP. The B&P is based on the *set-partitioning formulation* (SPF) of the SUBP and is applicable for solving instances with general element weights.
- We propose and explore different exact solution approaches to the SUKP pricing problem that are based on three alternative formulations: an *integer programming* (IP) formulation, an item-based *shortest path problem with resource constraints* (SPPRC), and an element-based SPPRC. Furthermore, we derive a greedy pricing heuristic following ideas of Arulselvan (2014) that is able to quickly generate a large number of negative reduced cost-columns

and typically leaves only few iterations to the exact pricer. The overall best performing variant combines the upfront greedy pricing heuristic with an item-based labeling algorithm that does not apply any dominance but relies on strong completions bounds. While the general idea of the labeling algorithm is generic and has also been applied by Wahlen and Gschwind (2023) for the OBP, the bounding procedure, which is crucial for its effectiveness, is strongly problem specific. We derive completion bounds that can be effectively computed for each label by solving a binary KP.

- We show the competitiveness of the proposed algorithm on the large unit-weight benchmark of Grange *et al.* (2018) and on new large-scale general-weight instances based on He *et al.* (2018). Our B&P by far outperforms the state-of-the-art IP formulations of Jans and Desrosiers (2013) and is able to optimally solve 92% of the 10,986 instances by Grange *et al.* (2018). We improve more than 5,800 *best-known solutions* (BKS) reported by Grange *et al.* (2018) and confirm all remaining BKS except for 131.
- Thanks to the large number of proven optima, we can perform a first analysis of the *(modified) integer round up property* ((M)IRUP) for the SUBP. We observe that for all instances the optimal objective value equals the rounded-up value of the corresponding optimal LP relaxation plus one, i.e., the MIRUP is satisfied.

4.1.2 Organization of the Paper

The remainder of the paper is structured as follows. In Section 4.2, we review the related literature. Section 4.3 formally defines the SUBP and presents three different IP formulations of the problem. The details of our exact B&P algorithm are given in Section 4.4. Section 4.5 presents our computational results. Final conclusions are drawn in Section 4.6.

4.2 Literature Review

The SUBP was first introduced as the *parts-grouping problem* in a manufacturing context by Hirabayashi *et al.* (1984), who propose a set-covering formulation. The term *set-union bin packing* was established by Goldschmidt *et al.* (1994) to describe the problem in analogy to the related SUKP. The literature contains several studies dealing with the SUBP addressed by various names such as *job grouping problem* (Tang and Denardo 1988, Crama and Oerlemans 1994, Jans and Desrosiers 2013, Desrosiers *et al.* 2013), *tool switching instants problem* (Konak *et al.* 2008, Konak and Kulturel-Konak 2007, Marvizadeh and Choobineh 2013, Gokgur and

Özpeynirci 2022), *virtual machine packing problem* (Sindelar *et al.* 2011), *machine stop minimization problem* (Adjiashvili *et al.* 2015), *pagination problem* (Grange *et al.* 2018, 2023), *bin packing problem with color constraint* (Kochetov and Kondakov 2017), *subset bin packing problem* (Dror and Haouari 2000, Izumi *et al.* 1998), or *bin packing problem with overlapping items* (Grange *et al.* 2018).

In the following, we focus on the current state of research on exact solution approaches to the SUBP. For simplicity, we uniformly use the terminology *item*, *element* and *bin* to describe the approaches, even if the authors use other terms depending on the application. A discussion of SUBP variants and extensions can be found, e.g., in (Calmels 2019, Locatelli 2023). We refer to (Wei 2021) for a broad overview on approaches to the SUKP, which constitutes the CG pricing problem of the SUBP.

The literature on exact solution approaches is still very limited, despite the high practical relevance of the SUBP. Tang and Denardo (1988) present a SPF of the SUBP and show that the SUBP is a generalization of the well-known BP. Furthermore, the authors propose an exact *branch-and-bound* (B&B) approach to solve the SUBP. For bounding, a lower bound and an upper bound are determined in each B&B node by a *sweeping procedure* and a *maximum intersection minimum union* (MIMU) heuristic, respectively. In the B&B tree, each node corresponds to a maximum class which is defined as a set of items whose union of required elements respects the bin capacity and adding another item to this set violates the bin capacity. Branching is realized by sequential maximum partition where at each node of the tree, all maximum classes that contain an arbitrarily selected item that was not part of a previously formed class are generated. To reduce the computational cost and the number of B&B nodes, the item with the smallest number of elements that still needs to be grouped is chosen. The B&B approach is considered 'efficient' for unit-weight instances with up to 30 items where a capacity of up to ten is assumed and the maximum number of elements is 20. The study is of theoretical nature and does not show computational results in terms of computation times.

Denizel (2003) utilizes an IP formulation of the SUBP presented by Crama and Oerlemans (1994) to propose a Langrangean decomposition-based lower bounding procedure. This lower bound, strengthened by adding valid inequalities to the decomposed model, is then used in an exact B&B algorithm. In each node, an upper bound is derived from the lower bound solution. For branching, the sequential maximum partition idea of Tang and Denardo (1988) is refined. Instead of choosing the least compatible item, the author selects the item that has not yet been grouped and that requires the largest weight of additional elements. The algorithm is able to optimally solve instances limited to a maximum of 30 unit-weight elements, 20 items and a capacity of 19 within a time limit of 5,400 seconds. The

procedure is also valid for the general-weight case.

Jans and Desrosiers (2013) analyze several IP formulations of the problem. They consider variants of the *symmetric IP formulation* (SF) of Crama and Oerlemans (1994) by adding symmetry breaking constraints such as, e.g., variable reduction and lexicographic ordering restrictions. They also present a new formulation of the SUBP based on the *asymmetric representatives formulation* (ARF) idea that was first proposed for vertex coloring (Junglas 2007, Campêlo *et al.* 2008) and was generalized for binary clustering problems by Jans and Desrosiers (2010). All IPs are solved with CPLEX and outperform the existing solution approaches from the literature. On the small instances of Denizel (2003), the new ARF is 40 times faster compared to the pure SF and outperforms the B&B algorithm of Denizel (2003) by factor seven. The other IPs with symmetry breaking also outperform the B&B algorithm, providing a speedup of factor six (SF with variable reduction) to eleven (SF with lexicographic ordering) compared to pure SF. On newly generated, larger instances with up to 60 items, 30 elements and a maximum capacity of 27, the SF with limited lexicographic ordering constraints performs best with a speedup of factor five compared to the pure SF model.

4.3 Problem Description and Mathematical Formulations

In this section, we formally define the SUBP and present three mathematical formulations of it.

4.3.1 Problem Definition

We are given a set of items $I = \{1, \ldots, n\}$, a set of weighted elements $E = \{1, \ldots, m\}$ with weights $w_e \geq 0$ for all $e \in E$, and an unlimited number of bins with capacity Q. Each item $i \in I$ requires a specific subset of elements $E_i \subset E$. The bin capacity Q specifies the maximum total weight of distinct elements per bin and we assume Q to be sufficiently large to encompass any item. The SUBP consists of grouping the items I into the minimum number of bins such that each item is assigned to exactly one bin and each bin satisfies the capacity restriction. In contrast to the standard BP, items can share one or more elements in a non-additive fashion, i.e., each element is considered only once in determining the total weight of a given set of items, regardless of whether the element is required by multiple of these items.

Example 4.1. *Consider an instance of the SUBP with four items* $I = \{i_1, \ldots, i_4\}$ *and five unit-weight elements* $E = \{e_1, \ldots, e_5\}$*. The sets of required elements are*

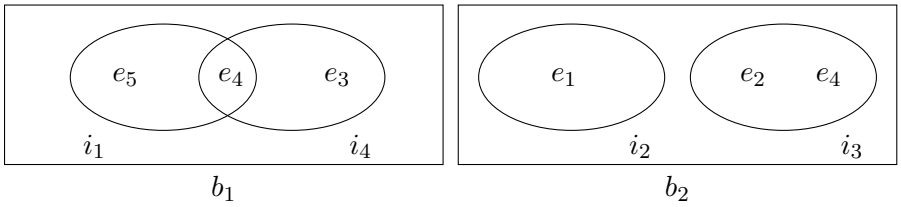


Figure 4.1: Example solution of a unit-weight SUBP instance with four items $I = \{i_1, \ldots, i_4\}$ requiring elements $E_{i_1} = \{e_4, e_5\}$, $E_{i_2} = \{e_1\}$, $E_{i_3} = \{e_2, e_4\}$, $E_{i_4} = \{e_3, e_4\}$, and $Q = 3$

$E_{i_1} = \{e_4, e_5\}$, $E_{i_2} = \{e_1\}$, $E_{i_3} = \{e_2, e_4\}$, and $E_{i_4} = \{e_3, e_4\}$. The bin capacity is $Q = 3$. Two bins are necessary to hold all items. Figure 4.1 depicts an optimal solution where items i_1 and i_4 requiring elements $\{e_3, e_4, e_5\}$ are assigned to bin b_1 while items i_2 and i_3 requiring elements $\{e_1, e_2, e_4\}$ are assigned to bin b_2. Note that element e_4 is shared by items i_1 and i_4 in b_1 so that the total weight in each bin is three.

It is convenient to introduce some additional notation. For a subset $E' \subseteq E$ of elements, denote by $w(E') = \sum_{e \in E'} w_e$ the total weight of the elements E'. For a subset $I' \subseteq I$ of items, denote by $E(I') = \bigcup_{i \in I'} E_i$ the set of distinct elements required by the items I' and by $w(I') = w(E(I')) = \sum_{e \in E(I')} w_e$ the total weight of these elements. Furthermore, we define the *frequency* $f_e(I') = |\{i \in I' \mid e \in E_i\}|$ of an element $e \in E$ as the number of items in set $I' \subseteq I$ that require element e. Note that if there is no common element between any pair of items, i.e., $f_e(I) \leq 1$ for all $e \in E$, the SUBP reduces to the BP (Tang and Denardo 1988).

4.3.2 Symmetric Formulation

To formulate the SUBP as a generalization of the BP, let B be a sufficiently large set of available bins, e.g., $B = \{1, \ldots, n\}$. A more evolved estimate on the size of B is described in Section 4.B of the appendix. The SF uses three types of binary indicator variables. Variables x_{ib} equal to one if item $i \in I$ is assigned to bin $b \in B$, and zero otherwise. Variables y_{eb} equal to one if element $e \in E$ is required in bin $b \in B$, and zero otherwise. Variables z_b equal to one if bin $b \in B$ is used, and zero otherwise.

The SF of Crama and Oerlemans (1994) reads as follows:

$$\min \sum_{b \in B} z_b \tag{4.1a}$$

$$\text{s.t.} \sum_{b \in B} x_{ib} = 1 \quad \forall i \in I \tag{4.1b}$$

$$\sum_{e \in E} w_e y_{eb} \leq Q z_b \quad \forall b \in B \tag{4.1c}$$

$$x_{ib} \leq y_{eb} \quad \forall b \in B, i \in I, e \in E_i \tag{4.1d}$$

$$x_{ib} \in \{0,1\} \quad \forall b \in B, i \in I \tag{4.1e}$$

$$y_{eb} \in \{0,1\} \quad \forall b \in B, e \in E \tag{4.1f}$$

$$z_b \in \{0,1\} \quad \forall b \in B \tag{4.1g}$$

The Objective (4.1a) minimizes the total number of bins used. Constraints (4.1b) ensure that all items are packed exactly once. Constraints (4.1c) guarantee compliance with the capacity of each bin regarding its allocated elements. The coupling of items and their required set of elements is enforced by Constraints (4.1d).

In order to avoid symmetric solutions, Jans and Desrosiers (2013) propose (among others) the following symmetry breaking constraints, which can be added to Formulation (4.1):

$$\sum_{i \in I} 2^{n-i} x_{ib} \geq \sum_{i \in I} 2^{n-i} x_{i,b+1} \quad \forall b \in B \setminus \{|B|\}. \tag{4.2}$$

Constraints (4.2) establish a lexicographic ordering of the bins according to the lowest indexed item assigned to each bin. Formulation (4.1) together with Constraints (4.2) was among the best formulations tested by Jans and Desrosiers (2013), in particular for larger instances. It is referred to as SF-LEX-I in the following.

4.3.3 Asymmetric Representatives Formulation

The ARF proposed by Jans and Desrosiers (2013) identifies bins by the lowest-indexed items packed into them. This makes redundant the use of bin setup variables. More specifically, binary variables v_{ih} are equal to one if and only if item $i \in I$ is assigned to the bin identified by item $h \in I, h \leq i$. Binary variables y_{eh} are now defined to equal one if and only if element $e \in E$ is assigned to the bin identified by item $h \in I$.

The ARF reads as follows:

$$\min \sum_{h \in I} v_{hh} \tag{4.3a}$$

$$\text{s.t.} \sum_{h \in I, h \leq i} v_{ih} = 1 \quad \forall i \in I \tag{4.3b}$$

$$\sum_{e \in E} w_e y_{eh} \leq Q v_{hh} \quad \forall h \in I \tag{4.3c}$$

$$v_{ih} \leq v_{hh} \quad \forall i, h \in I, i \geq h \tag{4.3d}$$

$$v_{ih} \leq y_{eh} \quad \forall i, h \in I, i \geq h, e \in E_i \tag{4.3e}$$

$$v_{ih} \in \{0, 1\} \quad \forall i, h \in I, i \geq h \tag{4.3f}$$

$$y_{eh} \in \{0, 1\} \quad \forall h \in I, e \in E \tag{4.3g}$$

The Objective (4.3a) minimizes the total number of bins used. Constraints (4.3b) ensure that all items are assigned to exactly one bin. Constraints (4.3c) guarantee compliance with the capacity of each bin with respect to its allocated elements. The coupling of items and their bin identifier is enforced by Constraints (4.3d), while Constraints (4.3e) ensure coupling of each item and its required set of elements.

4.3.4 Set-Partitioning Formulation

The SPF of the SUBP has been first proposed by Tang and Denardo (1988). An item subset $I' \subseteq I$ is said to be feasible if it satisfies $w(I') \leq Q$. Let Ω be the set of all feasible item subsets I'. We refer to such (feasible) item subsets as (feasible) bins in the following. Binary parameters r_{ib} indicate if item $i \in I$ is contained in bin $b \in \Omega$ ($r_{ib} = 1$) or not ($r_{ib} = 0$). Binary decision variables λ_b equal to one if bin $b \in \Omega$ is selected and zero otherwise. Then, the SUBP can be formulated as follows:

$$\min \sum_{b \in \Omega} \lambda_b \tag{4.4a}$$

$$\text{s.t.} \sum_{b \in \Omega} r_{ib} \lambda_b = 1 \quad \forall i \in I \tag{4.4b}$$

$$\lambda_b \in \{0, 1\} \quad \forall b \in \Omega \tag{4.4c}$$

The Objective (4.4a) minimizes the total number of bins used, while Constraints (4.4b) ensure that all items are packed exactly once.

4.4 Branch-and-Price Algorithm

Formulation (4.4) contains an exponential number of variables, i.e., feasible bins, so that it typically cannot be solved directly. We, therefore, employ a B&P algorithm

for its solution. A B&P algorithm is a B&B algorithm that uses CG to compute the lower bounds. CG alternates between solving a *restricted master problem* (RMP), in our case the linear relaxation of (4.4) comprising only a subset of the variables, and solving a pricing problem that generates variables with negative reduced cost. For details on CG and B&P, we refer to (Barnhart *et al.* 1998, Lübbecke and Desrosiers 2005).

In Sections 4.4.1 and 4.4.2, we present the details of our B&P with respect to pricing problem solution and branching, respectively. An overview of additional design choices and implementation details can be found in Appendix 4.B.

4.4.1 Pricing Problem

Let π_i be the dual prices associated with Constraints (4.4b). The reduced cost of a bin b is given by $\tilde{c}_b = 1 - \sum_{i \in b} \pi_i$. The pricing problem consists of identifying at least one feasible bin $b \in \Omega$ with negative reduced cost or to guarantee that no such bin exists. Note that minimizing $\tilde{c}_b$ over $b \in \Omega$ is equivalent to finding a feasible bin b that maximizes $\sum_{i \in b} \pi_i$. The latter corresponds to solving a SUKP over items $i \in I$ with required elements E_i and profits π_i, and bin capacity Q. Whenever the resulting objective function value is greater than one, a feasible bin with negative reduced cost is found. As shown by Goldschmidt *et al.* (1994), the SUKP is $\mathcal{NP}$-hard even for very restrictive cases.

In the following, we present different exact solution approaches to the pricing problem that are based on three alternative formulations of it: an SUKP IP formulation, an item-based SPPRC, and an element-based SPPRC. The former is solved directly with a general-purpose MIP solver. The latter two are solved by ad hoc defined labeling algorithms, both of them with and without dominance. Details on a greedy pricing heuristic and further acceleration strategies to speed-up the solution of the pricing problem are provided in Appendix 4.A.

4.4.1.1 IP Formulation

The SUKP pricing problem can be modeled using the IP formulation of Hirabayashi *et al.* (1984). Let x_i (y_e) be binary variables indicating the inclusion or not of item $i \in I$ (element $e \in E$) into the bin.

The formulation reads as follows:

$$\max \sum_{i \in I} x_i \pi_i \tag{4.5a}$$

$$\text{s.t.} \sum_{e \in E} y_e w_e \leq Q \tag{4.5b}$$

$$x_i \leq y_e \quad \forall i \in I, e \in E_i \tag{4.5c}$$

$$x_i \in \{0,1\} \quad \forall i \in I \tag{4.5d}$$

$$y_e \in \{0,1\} \quad \forall e \in E \tag{4.5e}$$

The Objective (4.5a) maximizes the total profit of included items. Constraints (4.5b) ensure that the total weight of all required elements does not exceed the bin capacity, while Constraints (4.5c) guarantee the coupling of each item with its required elements. The solution of Formulation (4.5) with a general-purpose MIP solver represents a first option for solving the pricing problem.

4.4.1.2 Item-based SPPRC

The pricing problem can also be modeled as an SPPRC on an ad hoc defined graph. SPPRCs are typically solved with *dynamic programming* (DP) labeling algorithms (Irnich and Desaulniers 2005). In a labeling algorithm, labels representing partial paths are extended from a given source to a given sink along the network arcs using *resource extension functions* (REFs). To avoid enumerating all feasible paths, dominance relations between labels to eliminate provably non-optimal paths and bounding procedures to discard unpromising paths that cannot reach a given objective value threshold can be applied.

We describe two different types of SPPRC representations of the pricing problem denoted item-based and element-based SPPRC and their solution by DP labeling algorithms. All presented labeling algorithms heavily rely on a strong bounding procedure to discard unpromising labels. SPPRC representations similar to the item-based SPPRC have also been used for other pricing problems with a knapsack-type substructure (e.g., Heßler *et al.* 2018, Gschwind *et al.* 2019).

Item-based Representation Let $G = (V, A)$ be a linear directed multigraph with $n + 1$ vertices $V = \{0, \ldots, n\}$ and $2n$ arcs A. Vertex 0 is an artificial source. Vertices $1, \ldots, n$ correspond with the n items in a given sorting. For ease of notation, we assume throughout this section that in the SPPRC graph the items are sorted by their index, meaning that vertex $v \in V \setminus \{0\}$ corresponds with item $i = v$. For each $v \in V \setminus \{0\}$, there are two parallel arcs a_v^1 and a_v^0 connecting vertices $v - 1$ and v indicating the inclusion or not, respectively, of item v. Each arc $a_v^k \in A, v \in V \setminus \{0\}, k \in \{0,1\}$ is associated with a set of items I_v^k, a set of

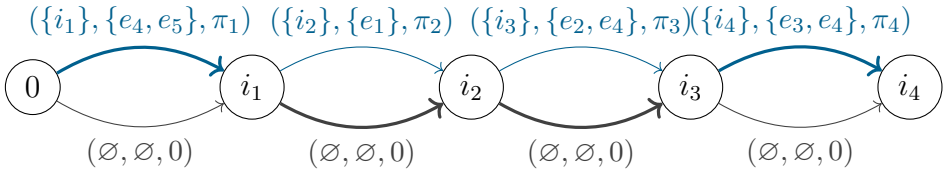


Figure 4.2: Linear directed multigraph G of the item-based SPPRC representation of the pricing problem

elements E_v^k, and a dual price π_v^k. Accordingly, for arc a_v^1 we have $I_v^1 = I_v$, $E_v^1 = E_v$, and $\pi_v^1 = \pi_v$, while for arc a_v^0 we have $I_v^0 = E_v^0 = \varnothing$ and $\pi_v^0 = 0$. Associating sets of items I_v^k with the arcs allows the simultaneous consideration of multiple items which is needed for the incorporation of branching decisions (see Section 4.4.2). Note that the information E_v^k on the elements is redundant (it can be determined from the items I_v^k), but simplifies the presentation and the labeling algorithm.

Any 0-n-path $(v_0, a_{v_1}^{k_1}, v_1, \ldots, a_{v_n}^{k_n}, v_n)$ in G defines a bin b with items $I(b) = \bigcup_{i=1}^n I_{v_i}^{k_i}$ requiring elements $E(b) = \bigcup_{i=1}^n E_{v_i}^{k_i}$. It is feasible if $w(E(b)) \leq Q$. Recall that the capacity consumption $w(E(b))$, or $w(b)$ for short, is a function that is not separable in the items $I(b)$ of a bin b, but always depends on the union of elements required by the items in b. The reduced cost of bin b is $\tilde{c}_b = 1 - \sum_{i=1}^n \pi_{v_i}^{k_i}$. The solution of the pricing problem is equivalent to finding a capacity-feasible 0-n-path in G with minimum reduced cost.

Example 4.2 (continued)**.** *Figure 4.2 illustrates graph G for the example SUBP instance and dual prices π_i. There are two arcs between each pair of consecutive vertices, indicating the inclusion (blue arc) or not (gray arc) of the item associated with the respective head vertex. Consider vertex $v = i_1$. The ingoing blue arc $(\{i_1\},\{e_4,e_5\},\pi_{i_1}) = a_1^1$ represents the inclusion of the singleton item i_1, i.e., its required set of elements $E_{i_1} = \{e_4, e_5\}$, into a bin and is associated with the dual price π_{i_1}. The ingoing gray arc $(\varnothing,\varnothing,0) = a_1^0$ corresponds with not including any item ($I_1^0 = E_1^0 = \varnothing$) and therefore $\pi_1^0 = 0$. The path corresponding to bin b_1 comprising items i_1 and i_4 (as shown in Figure 4.1) is visualized by bold arcs.*

Item-based Labeling Algorithm A partial path $P_L = (0, a_1^{k_1}, 1, \ldots, a_v^{k_v}, v)$ from the source 0 to some vertex v is represented by a label

$$L = \Big(v(L), E(L), I(L), \tilde{c}(L), w(L), I^C(L)\Big)$$

storing its last vertex $v(L)$, the set of required elements $E(L)$, the set of items $I(L)$, the reduced cost $\tilde{c}(L)$, the capacity consumption $w(L)$, and the set of compatible items $I^C(L)$. An item is considered compatible if it can be added with any

capacity-feasible extension of label L to the sink vertex n. The initial label at the artificial source 0 is given by $(0, \varnothing, \varnothing, 1, 0, I)$. In the linear graph, labels are processed vertex-by-vertex in our labeling algorithm. This means that starting with the initial label at the artificial source, we always propagate all labels at a given vertex $v-1$ along the arcs a_v^0 and a_v^1 to vertex v, before all resulting labels at vertex v are in turn propagated to vertex $v+1$, etc., until finally the sink vertex is reached.

The extension of a label L at vertex $v-1$ to vertex v along arc a_v^k is feasible, if $w(E(L) \cup E_v^k) \leq Q$. If the extension is feasible, a new label L' is created according to the following REFs:

$$v(L') = v \tag{4.6a}$$
$$E(L') = E(L) \cup E_v^k \tag{4.6b}$$
$$I(L') = I(L) \cup I_v^k \tag{4.6c}$$
$$\tilde{c}(L') = \tilde{c}(L) - \pi_v^k \tag{4.6d}$$
$$w(L') = \sum_{e \in E(L) \cup E_v^k} w_e \tag{4.6e}$$
$$I^C(L') = I^C(L) \setminus \{i \in I^C(L) \mid q(E(L) \cup E_v^k \cup E_i) > Q \vee i \in I_v^k\} \tag{4.6f}$$

REFs (4.6a)–(4.6d) update the current vertex, the set of elements, the set of items, and the reduced cost in a straightforward manner according to the respective component of arc a_v^k. The total weight of all distinct elements is determined by REF (4.6e). REF (4.6f) is used to identify the new set of compatible items by reducing the former set of compatible items by the items whose inclusion would cause the capacity to be exceeded and by the item(s) associated with arc a_v^k.

The non-separability of the capacity consumption $w(L')$ in the items imposes two major drawbacks on the algorithm when labeling on items. First, in every label propagation, a more costly evaluation of the capacity consumption $w(L')$ is necessary in REF (4.6e). Second, it renders infeasible the standard *less-or-equal* dominance relation of the capacity resource as applied in many labeling algorithms for SPPRC variants. Instead, the specific sets of packed elements have to be taken into account:

Definition 4.1. *Let L_1 and L_2 be two different labels associated with the same last vertex $v(L_1) = v(L_2)$. Label L_2 is said to be dominated by label L_1 if*

$$\tilde{c}(L_1) \leq \tilde{c}(L_2) \ \wedge \ E(L_1) \subseteq E(L_2). \tag{4.7}$$

If L_1 dominates L_2, then L_2 can be discarded. In case of mutual dominance, one label has to be kept. Due to the rather strict condition $E(L_1) \subseteq E(L_2)$, we can expect this dominance rule to be weak in general. It is not clear whether or not

the additional effort to test a set of labels for dominance relations pays off. We, therefore, compare two variants of the proposed labeling algorithm in Section 4.5.2: one that applies dominance and one that does not.

Bounding Procedure Let $LB(L)$ be a lower bound on the reduced cost of any capacity-feasible 0-n-path in G that contains the 0-$v(L)$-path P_L corresponding to label L. Obviously, any label L with $LB(L) \geq 0$ can be discarded. In the following, we describe a method for computing values $LB(L)$ that can also be adapted to cope with the branching decisions of our B&P algorithm (see Section 4.4.2).

For a label L, let $R(L)$ be the set of $v(L)$-n-paths that can be appended to the 0-$v(L)$-path P_L to form capacity-feasible 0-n-paths. A path $r \in R(L)$ is called a completion of L. Denote by L^r the label corresponding to path (P_L, r) and let $I(r) = I(L^r) \setminus I(L) \subseteq I^C(L)$. It holds that $\tilde{c}(L^r) = \tilde{c}(L) - \sum_{i \in I(r)} \pi_i$. Thus, a valid lower bound $LB(L)$ on the reduced cost of any capacity-feasible 0-n-path containing path P_L is given by

$$LB(L) = \tilde{c}(L) - \max_{r \in R(L)} \sum_{i \in I(r)} \pi_i. \tag{4.8}$$

Intuitively speaking, the value $\max_{r \in R(L)} \sum_{i \in I(r)} \pi_i$ represents the maximum dual prices that can be collected when extending label L to a capacity-feasible 0-n-path. Because set $R(L)$ comprises all completions r such that $w(L^r) \leq Q$, this value is equivalent to the optimal solution value of a SUKP regarding the compatible items $i \in I^C(L)$ each requiring the subset of elements $E_i' = E_i \setminus E(L)$, with profits π_i, and capacity $Q' = Q - w(L)$. Note that the sets $R(L)$ depend not only on the capacity consumption $w(L)$ and last vertex $v(L)$, but on the specific sets of compatible items $I^C(L)$ and elements $E(L)$. Therefore, individual completion bounds $B(L) = \max_{r \in R(L)} \sum_{i \in I(r)} \pi_i$ have to be defined for each label L.

The exact solution of an SUKP for each label L is not practicable. Instead, we solve a relaxation in order to obtain valid completion bounds. More specifically, we solve a standard binary KP over items $i \in I^C(L)$ with ad hoc defined weights $\tilde{w}_i(L)$, profits π_i, and capacity Q' using a DP algorithm that runs in pseudo-polynomial time $\mathcal{O}(nQ)$ (Kellerer *et al.* 2004). The resulting optimal KP value constitutes completion bound $B_1(L) \geq B(L)$. For each item $i \in I^C(L)$, weight $\tilde{w}_i(L)$ is constructed based on the relative (to their frequency in $I^C(L)$) weights $\tilde{w}_e(L) = \frac{w_e}{f_e(I^C(L))}$ of its required relevant elements $e \in E_i'$. For example, an element that occurs in five compatible items (but not in any packed item) contributes one fifth of its original weight to the weight of each of its comprising items. Elements that are required by items already in the bin can be omitted. Reducing the weight of each single required element to this fraction allows elements to be included multiple times in the KP without exceeding their original weight that would accumulate in

the SUKP. To foster an efficient table-based implementation of the DP algorithm, integer item weights can be obtained by flooring. The resulting item weights yield a valid but potentially weaker bound. Instead, we propose multiplying the sum of relative element weights by a factor $d \in \mathbb{N}^+$ before rounding down. Formally, item weights are computed as

$$\tilde{w}_i(L) = \left\lfloor d \cdot \sum_{e \in E_i{}'} \tilde{w}_e(L) \right\rfloor. \tag{4.9}$$

Accordingly, however, the residual capacity and thus the dimension of the DP table also increase by factor d. In pretests, we found that a value of $d = 10$ provides a good tradeoff between strength of the bound and computational effort for its determination.

Example 4.3 (continued)**.** *Consider the example in Figure 4.3. Let label* $L = (i_2, \{e_1\}, \{i_2\}, 0.5, 1, \{i_3, i_4\})$ *be a label at vertex* i_2 *and assume dual prices* $\pi_{i_3} = 0.4$ *and* $\pi_{i_4} = 0.2$*. We have* $f_{e_2}(\{i_3, i_4\}) = f_{e_3}(\{i_3, i_4\}) = 1$ *and* $f_{e_4}(\{i_3, i_4\}) = 2$*, thus* $\tilde{w}_{e_2} = \tilde{w}_{e_3} = 1$ *and* $\tilde{w}_{e_4} = 0.5$*. With* $d = 1$*, the resulting KP item weights are* $\tilde{w}_{i_3}(L) = \tilde{w}_{i_4}(L) = \lfloor 1 \cdot 1.5 \rfloor = 1$ *and the KP capacity is* $Q' = 2$*. In the corresponding optimal KP solution, both items* i_3 *and* i_4 *are added to the knapsack and the completion bound value is* $B_1^{d=1}(L) = 0.6$*. Because* $B_1^{d=1}(L) > \tilde{c}(L)$*, label* L *cannot be discarded but needs to be further extended. Figure 4.3 reveals that with* $d = 2$*, we have* $\tilde{w}_{i_3}(L) = \tilde{w}_{i_4}(L) = 3$ *and* $Q' = 4$*. With these values, we obtain* $B_1^{d=2}(L) = 0.4 \leq \tilde{c}(L)$ *and label* L *can be discarded.*

As an additional, computationally cheaper completion bound, we consider

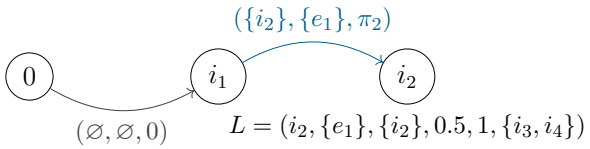
$$B_2(L) = \sum_{i \in I^C(L)} \pi_i. \tag{4.10}$$

It allows to quickly eliminate unpromising labels L with $\tilde{c}(L) \geq B_2(L)$. Only for labels L satisfying $\tilde{c}(L) < B_2(L)$, we compute the bound $B_1(L)$ by solving the corresponding binary KP and discard L if $\tilde{c}(L) \geq B_1(L)$.

We also experimented with other bounds, such as the sum of the dual prices π_i of all items $i \in I, i > v$ denoted $B_3(v)$, which applies to all labels L with $v(L) = v$ and needs to be set up only once prior to the actual labeling process, or the solution of the linear relaxation of Formulation (4.5) with respect to all $i \in I^C(L)$ denoted $B_4(L)$. However, pretests showed that the combination of B_1 and B_2 was the most promising (see Appendix 4.E).

4.4.1.3 Element-based SPPRC

We now present an alternative element-based SPPRC representation and solution approach of the pricing problem. For conciseness, we focus on the differences compared to our item-based SPPRC. Note that the element-based SPPRC approach is



Data for $B_1(L)$ with $d = 1$

Q'	$\tilde{w}_{e_2}$	$\tilde{w}_{e_3}$	$\tilde{w}_{e_4}$	$\tilde{w}_{i_3}$	$\tilde{w}_{i_4}$	π_{i_3}	π_{i_4}
2	1	1	0.5	1	1	0.4	0.2

Data for $B_1(L)$ with $d = 2$

Q'	$\tilde{w}_{e_2}$	$\tilde{w}_{e_3}$	$\tilde{w}_{e_4}$	$\tilde{w}_{i_3}$	$\tilde{w}_{i_4}$	π_{i_3}	π_{i_4}
4	1	1	0.5	3	3	0.4	0.2

DP table for $B_1(L)$ with $d = 1$

Q'	i_3	i_4
2	0.4	<u>0.6</u>
1	0.4	0.4
0	0.0	0.0

DP table for $B_1(L)$ with $d = 2$

Q'	i_3	i_4
4	0.4	<u>0.4</u>
3	0.4	0.4
2	0.0	0.0
1	0.0	0.0
0	0.0	0.0

Figure 4.3: Example representation for the determination of bound $B_1(L)$ for a label L at vertex i_2

similar to the element-based DP algorithm described by Goldschmidt *et al.* (1994) to solve SUKPs.

Element-based Representation Let $\hat{G} = (\hat{V}, \hat{A})$ be a linear directed multi-graph with $m+1$ vertices $\hat{V} = \{0, \dots, m\}$ and $2m$ arcs $\hat{A}$. Vertex 0 is an artificial source and vertices $1, \dots, m$ are associated with the m elements. Again, we assume that the elements are sorted by their index so that vertex $\hat{v} \in \hat{V} \setminus \{0\}$ corresponds with element $e = \hat{v}$. Each arc $\hat{a}^k_{\hat{v}} \in \hat{A}, \hat{v} \in \hat{V} \setminus \{0\}, k \in \{0, 1\}$, then indicates the inclusion or not, respectively, of element $\hat{v}$, and is associated with a set of elements $E^k_{\hat{v}}$ and a weight $w^k_{\hat{v}}$. In contrast to the item-based SPPRC, the capacity consumption of each inclusion is now evident from the corresponding arc. However, the set of feasible items and the dual prices are not directly provided.

Any 0-m-path $(\hat{v}_0, \hat{a}^{k_1}_{\hat{v}_1}, \hat{v}_1, \dots, \hat{a}^{k_m}_{\hat{v}_m}, \hat{v}_m)$ in $\hat{G}$ defines a bin b which is feasible if $w(b) = \sum_{e=1}^{m} w^{k_e}_{\hat{v}_e} \leq Q$. The set of feasible items in b can be derived from the set of comprised elements $E(b) = \bigcup_{e=1}^{m} E^{k_e}_{\hat{v}_e}$ and is defined as $I(b) = \{i \in I \mid E_i \subseteq E(b)\}$. Thus, the reduced cost of b is $\tilde{c}_b = 1 - \sum_{i \in I(b)} \pi_i$. The solution of the pricing problem is equivalent to finding a capacity-feasible 0-m-path in $\hat{G}$ with minimum reduced cost.

Example 4.4 (continued)**.** *Figure 4.4 illustrates graph $\hat{G}$ for the example SUBP*

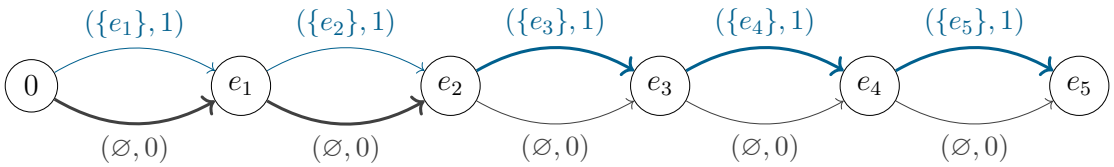


Figure 4.4: Linear directed multigraph $\hat{G}$ of the element-based SPPRC representation of the pricing problem

instance with $m = 5$ elements $e \in \{e_1, \dots, e_5\}$ and their unit weights w_e. The two arcs between each pair of consecutive vertices indicate the inclusion (blue arc) or not (gray arc) of the element associated with the respective head vertex. Consider vertex $\hat{v} = e_1$. The ingoing blue arc $(\{e_1\}, 1) = \hat{a}_1^1$ represents the inclusion of the singleton element e_1 into a bin and is associated with its weight $w_1 = 1$. The ingoing gray arc $(\varnothing, 0) = \hat{a}_1^0$ corresponds with not including any element ($E_1^0 = \varnothing$) and therefore $w_1 = 0$. Bin b_1 comprising items i_1 and i_4 from Figure 4.1 is shown by the path consisting of the bold arcs.

Element-based Labeling Algorithm Each partial path $\hat{P}_L = (0, \hat{a}_1^{k_1}, 1, \dots, \hat{a}_{\hat{v}}^{k_{\hat{v}}}, \hat{v})$ from the source 0 to a vertex $\hat{v}$ is represented by a label L with the same label components as defined in Section 4.4.1.2 for the item-based case. The extension of a label L at vertex $\hat{v} - 1$ to vertex $\hat{v}$ along arc $\hat{a}_{\hat{v}}^k$ is feasible, if $w(L) + w_{\hat{v}}^k \leq Q$. The REFs of the element-based labeling approach are:

$$v(L') = \hat{v} \tag{4.11a}$$
$$E(L') = E(L) \cup E_{\hat{v}}^k \tag{4.11b}$$
$$I(L') = I(L) \cup \{i \in I^C(L) \mid E_i \subseteq E(L) \cup E_{\hat{v}}^k\} \tag{4.11c}$$
$$\tilde{c}(L') = \tilde{c}(L) - \sum_{i \in I^C(L): E_i \subseteq E(L) \cup E_{\hat{v}}^k} \pi_i \tag{4.11d}$$
$$w(L') = w(L) + w_{\hat{v}}^k \tag{4.11e}$$
$$I^C(L') = I^C(L) \setminus \{i \in I^C(L) \mid q(E(L) \cup E_{\hat{v}}^k \cup E_i) > Q \vee E_i \subseteq E(L) \cup E_{\hat{v}}^k \vee \cup_{\kappa \in \{0,1\}} E_{\hat{v}}^\kappa \cap E_i \nsubseteq E_{\hat{v}}^k\} \tag{4.11f}$$

The REFs (4.11a), (4.11b), and (4.11e) update the current vertex, the set of elements, and the capacity consumption in a straightforward manner according to the respective components of arc $\hat{a}_{\hat{v}}^k$. The set of feasible items is augmented by those compatible items that can be completely realized by the current set of elements in REF (4.11c). REF (4.11d) updates the reduced cost by subtracting the dual prices of all newly realized items. In REF (4.11f), the new set of compatible

items is identified by reducing the former set by the items of three categories: (i) items whose inclusion would cause the capacity to be exceeded, (ii) items that can be realized by the current extension along arc $\hat{a}^k_{\hat{v}}$, and (iii) items requiring elements that are excluded by the current extension along arc $\hat{a}^k_{\hat{v}}$ (with the union $\cup_{\kappa\in\{0,1\}} E^\kappa_{\hat{v}}$ representing all elements associated with vertex $\hat{v}$).

Because the additional capacity consumption is directly associated with the arcs, the determination of $w(L')$ in REF (4.11e) is less expensive compared to (4.6e). However, the set of feasible items $I(L')$ must be identified for each inclusion of an element in order to be able to determine the reduced cost $\tilde{c}(L')$ of L'. Consequently, REF (4.11c) is more costly than (4.6c). Again, the standard *less-or-equal* relations of the reduced costs and the capacity consumptions are not sufficient to guarantee dominance between labels. Instead, it has to be incorporated, which additional items may potentially be included, i.e., which dual prices may be collected, by the addition of elements that have not yet been considered:

Definition 4.2. *Let L_1 and L_2 be two different labels associated with the same last vertex $v(L_1) = v(L_2)$. Label L_2 is said to be dominated by label L_1 if*

$$\tilde{c}(L_1) \leq \tilde{c}(L_2) \ \wedge \ w(L_1) \leq w(L_2) \ \wedge \ I^C(L_1) \supseteq I^C(L_2). \tag{4.12}$$

Similar to the item-labeling case, the dominance relation is expected to be weak due to the rather strict condition $I^C(L_1) \supseteq I^C(L_2)$ and we examine two variants (with and without dominance) of the labeling algorithm in Section 4.5.2. Note further, that the element-based labeling with dominance (but without bounding) is essentially equivalent to the DP algorithm by Goldschmidt *et al.* (1994). In their DP, the stages correspond with the elements (in a given sorting) and at each stage the DP decides for each state whether or not to include the current element. The states are characterized by their stage, the residual capacity, and the set of those packed elements that are needed to include any item that can be realized when adding elements of subsequent stages. The latter is equivalent to directly considering the items that can be realized regarding packed elements and subsequent elements, i.e., the compatible items $I^C(L)$.

Bounding Procedure The technique described in Section 4.4.1.2 for item-based labeling can be transferred directly to the element-based approach. The label-specific bounds B_1 and B_2 and their determination are identical.

4.4.2 Branching

We apply the well-known Ryan-and-Foster branching scheme (Ryan and Foster 1981) to ensure integrality. Denote by $(\bar{\lambda}_b)_{b\in\Omega'}$ the current fractional solution of

the RMP. Let $f_{ij} = \sum_{b \in \Omega'} r_{ib} r_{jb} \bar{\lambda}_b$ indicate if the two items $i, j \in I$ are assigned to the same bin. If f_{ij} is fractional, we can branch on pair (i, j) by creating two child nodes. The *separate* branch ensures $f_{ij} = 0$ by forcing variables λ_b with $r_{ib} = r_{jb} = 1$ to zero. The *together* branch ensures $f_{ij} = 1$ by forcing variables λ_b with $r_{ib} + r_{jb} = 1$ to zero. Both types of decisions can be easily realized in the RMP by forbidding the corresponding bin columns.

Bins that are incompatible with the Ryan-and-Foster branching decisions must also be prevented from being (re-)generated. This imposes structural changes on the pricing problem requiring the adaptation of the corresponding solution approaches. A generally valid approach is to embed the original pricing algorithm into a B&B algorithm that enforces consistency with the additional *separate*/*together* constraints (see Gschwind *et al.* (2021) for details). For the element-based pricing, this (or a similar technique) seems to be the only viable way, as it is not able to explicitly decide on items. For the MIP-solver and item-based methods, more effective approaches are possible. In the former, the IP Formulation (4.5) must simply respect an additional linear constraint for each branching decision on an item pair (i, j). In the *separate* branch, we add $x_i + x_j \leq 1$ to Formulation (4.5) while in the *together* branch, we add $x_i = x_j$. With item-based labeling, we alter the graph G of the SPPRC representation of the pricing problem. The overall idea is to group together the items affected by mutual branching decisions, represent them by a single vertex in G, and decide on the inclusion of all items in a group simultaneously. On the modified graph, the same labeling algorithm presented in Section 4.4.1.2 can be applied to solve the pricing problem in the presence of branching decisions. The details of the graph modification are described in Appendix 4.C.

4.5 Computational Results

Our B&P algorithm was implemented in `C++` and compiled into 64-bit single-thread code with MS Visual Studio 2019. CPLEX 20.10 with default parameters (except for the time limit and allowing only a single thread) is used to reoptimize the RMPs and as MIP-solver. The computations were carried out on the HPC cluster Elwetritsch of RPTU Kaiserslautern-Landau consisting of several Intel Xeon Gold 6126 processors running at 2.60 GHz. Memory was limited to 6 GB per thread. Notice that the performance of a single thread of the cluster is comparable to that of a standard desktop processor. The same computational setup was used for all instances and the time limit for each instance was set to 1,800 seconds. Unsolved instances are considered with the time limit of 1,800 seconds in our analysis. All used benchmark instances together with instance-by-instance results of our best performing B&P variant are provided at `https://wiwi.rptu.de/fgs/logistik/`

`subp-detailedresults`.

4.5.1 Benchmark Instances

We focus our computational study on the extensive unit-weight benchmark of Grange *et al.* (2018) (denoted `pagination`) and on large-scale, general-weight instances that are derived from the SUKP benchmark of He *et al.* (2018) (denoted `general`). A description of both benchmark sets is provided in Appendix 4.D. All considered instances have never been solved to proven optimality before (apart from 43 of the smallest instances from (Grange *et al.* 2018) that have been solved with CPLEX by the authors).

4.5.2 Analysis of Pricing Problem Solution Methods

We first investigate different pricing variants of our B&P algorithm on its performance. To this end, we compare the different exact solution approaches proposed in Section 4.4.1, namely solving Formulation (4.5) with CPLEX, item-based labeling with and without dominance, and element-based labeling with and without dominance. Furthermore, we test all solvers with and without the upstream greedy pricing heuristic (see Section 4.A of the appendix). Table 4.1 summarizes the results for solving the root node of all `pagination` instances. It reports the percentage number of instances with optimally solved LP relaxation ($\%Sol^{LP}$) and the average time for solving the LP relaxation in seconds (t^{LP}).

Consider first the results without using the greedy heuristic. The performance of all approaches decreases as n increases. Overall, IP and item-based labeling clearly outperform the element-based labeling in terms of both the number of solved instances and solution time. Although more instances can be solved using the IP, item-based labeling (without dominance) shows on average shorter computation times, in particular for large instances with $n \geq 75$. For both labeling methods, the application of dominance is not advantageous. A slight reduction in computation time can be achieved only for instances with the smallest number of elements ($m = 20$). A main reason for the poor performance of element-based labeling is the extremely large number of generated labels. Apparently, the bounding procedure is much less effective for element-based labeling compared to item-based labeling. As a result, more than two thirds of instances exceed the memory limit if dominance is not applied. If dominance is applied, memory issues are much less frequent but the large number of labels requires a huge number of dominance tests (quadratic in the number of labels).

All variants greatly benefit from integrating the greedy heuristic. Overall, the IP and item-based labeling are still able to solve twice as many instances in a fraction of the time compared to element-based labeling. Unlike without the heuristic,

			Item-based labeling				Element-based labeling			
	IP		No dominance		With dominance		No dominance		With dominance	
n	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}
Panel A: No greedy heuristic										
20	**100.0**	0.8	**100.0**	**0.0**	**100.0**	**0.0**	49.2	931.6	49.8	918.9
25	**100.0**	1.8	**100.0**	**0.2**	**100.0**	0.3	46.7	1,001.8	44.6	1,047.8
30	**100.0**	3.7	**100.0**	**0.8**	**100.0**	1.8	41.5	1,156.0	36.1	1,219.4
35	**100.0**	7.1	**100.0**	**4.6**	99.7	22.1	34.4	1,309.4	30.3	1,374.5
40	**100.0**	**15.4**	99.9	21.8	97.1	98.2	26.7	1,428.9	21.8	1,527.5
45	**100.0**	**32.3**	99.2	61.6	90.6	251.2	23.5	1,461.7	17.1	1,643.1
50	**100.0**	**67.8**	96.9	118.8	85.7	346.4	19.1	1,508.0	11.1	1,683.4
55	**100.0**	**125.3**	93.5	199.8	75.6	529.3	17.6	1,540.8	7.3	1,709.4
60	**100.0**	**199.8**	91.8	263.8	70.8	610.6	17.3	1,540.0	6.5	1,716.7
65	**99.4**	**287.6**	89.0	334.6	67.0	652.8	16.1	1,548.8	6.3	1,712.2
70	**96.3**	**413.8**	84.3	427.1	64.0	704.1	15.3	1,561.9	5.3	1,718.6
75	**91.5**	531.8	80.9	**497.8**	61.0	758.6	14.7	1,575.4	5.3	1,722.0
80	**86.9**	662.7	75.9	**591.4**	57.3	799.8	14.2	1,584.9	5.0	1,723.2
85	**78.1**	798.3	72.4	**636.2**	58.2	803.2	13.7	1,590.9	5.2	1,722.3
90	**72.4**	896.9	67.9	**702.5**	55.1	828.5	13.4	1,592.3	5.4	1,723.4
95	**67.6**	1,018.0	65.4	**743.8**	53.9	846.6	12.8	1,596.9	5.5	1,715.8
100	60.5	1,126.3	**61.1**	**788.8**	53.4	853.9	13.0	1,604.7	5.1	1,721.6
Total	**91.3**	365.1	86.9	**318.1**	75.8	478.2	22.8	1,444.5	15.8	1,563.7
Panel B: With greedy heuristic										
20	**100.0**	0.2	**100.0**	**0.0**	**100.0**	**0.0**	96.5	137.8	50.0	900.2
25	**100.0**	0.4	**100.0**	**0.0**	**100.0**	**0.0**	90.7	295.8	50.0	907.7
30	**100.0**	0.9	**100.0**	**0.0**	**100.0**	**0.0**	82.7	464.2	48.7	1,003.4
35	**100.0**	1.8	**100.0**	**0.1**	**100.0**	**0.1**	77.2	626.9	43.0	1,149.1
40	**100.0**	4.5	**100.0**	**0.3**	**100.0**	**0.3**	66.7	817.3	32.8	1,336.0
45	**100.0**	10.0	**100.0**	**0.8**	**100.0**	1.0	61.1	934.7	26.3	1,443.5
50	**100.0**	22.2	**100.0**	**2.1**	**100.0**	3.0	53.9	1,061.5	20.8	1,545.2
55	**100.0**	41.0	**100.0**	**4.5**	**100.0**	8.6	48.6	1,148.5	17.1	1,610.3
60	**100.0**	66.6	**100.0**	**8.2**	99.9	25.6	47.2	1,195.6	13.9	1,641.2
65	**100.0**	96.8	**100.0**	**13.7**	99.9	51.1	43.5	1,235.0	11.0	1,655.4
70	99.7	149.1	**100.0**	**23.4**	97.7	119.6	42.0	1,268.8	10.0	1,667.9
75	99.1	208.3	**100.0**	**34.4**	96.5	175.1	38.3	1,294.8	9.1	1,673.2
80	95.8	304.0	**100.0**	**59.8**	91.7	294.9	36.7	1,307.4	8.7	1,678.1
85	93.8	392.6	**100.0**	**80.5**	88.0	371.5	34.1	1,336.0	8.4	1,678.3
90	89.7	469.9	**99.1**	**133.3**	83.2	453.2	33.3	1,350.8	8.6	1,677.1
95	84.1	569.8	**97.8**	**185.8**	77.5	532.7	31.8	1,371.3	8.3	1,676.1
100	79.8	660.4	**96.9**	**232.8**	73.6	611.2	32.1	1,380.1	8.7	1,672.6
Total	96.6	176.9	**99.6**	**46.0**	94.6	156.2	53.8	1,015.6	22.2	1,463.9

Table 4.1: Summary results for pricing variants of our B&P for `pagination` instances

item-based labeling without dominance now dominates the IP-based pricing for all instance classes: it can solve significantly more instances in a fraction of the average computation time.

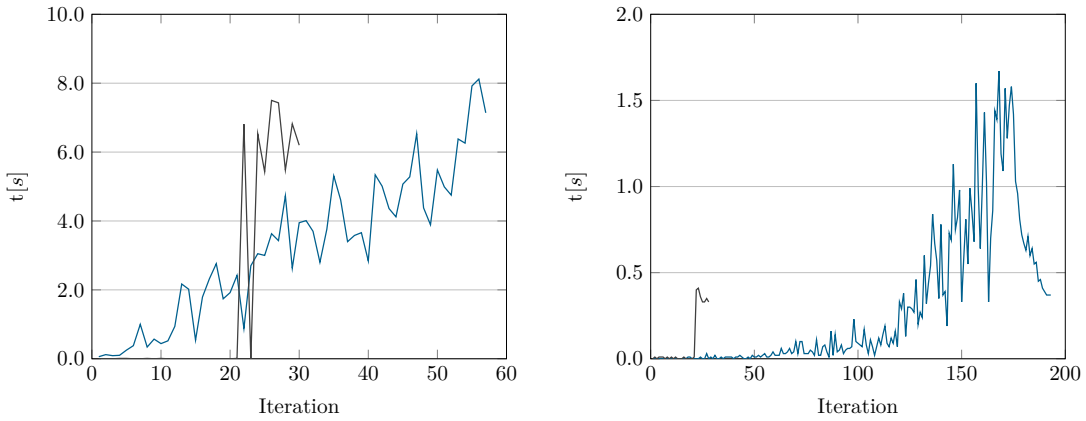
In order to understand the different behavior of the B&P regarding the IP and item-based labeling without dominance when using or not the greedy pricing heuristic, we analyzed the CG process in more detail for those variants. Figure 4.5 depicts, for the fifth `pagination` instance with $(Q, m, n) = (45, 85, 50)$, the computation time in seconds for each individual pricing instance for solving the LP relaxation when using the IP (4.5a) and the item-based labeling (4.5b) with (gray plots) and without (blue plots) the upfront heuristic. Note that the behavior shown in Figure 4.5 is representative for the complete benchmark. Additional examples are shown in Appendix 4.F.

Recall that both the IP and item-based labeling are used in a partial-pricing fashion and apply multiple-column pricing (see Appendix 4.A) so that they typically generate different columns. Consequently, we can expect the corresponding B&P variants to follow different trajectories beyond the first iteration. Figure 4.5 reveals that without the heuristic, much fewer iterations are required when using IP compared to item-based labeling, while the latter shows significantly shorter computation times per pricing instance. Moreover, for IP, the computation times per pricing instance show an increasing trend in the iteration number with the longest computation times arising for the final pricing iterations. In contrast, for item-based labeling, we observe kind of a bell curve with the final iterations being computationally rather inexpensive. With the heuristic, two main observations emerge. First, the number of iterations can be reduced for both approaches, however, substantially more iterations are saved with item-based labeling than with IP. Second, for both approaches the computation times of the (relatively few) iterations of the exact pricer are of similar magnitude as those of the final iterations without the heuristic. Consequently, while for item-based labeling the most time-consuming iterations are replaced by the heuristic, the most time-consuming final iterations still have to be performed for IP.

Summing up, the best performing pricing strategy for our B&P is to select the item-based labeling without dominance but with upfront greedy heuristic. All further calculations are carried out on the basis of this setting.

4.5.3 Comparison with State-of-the-Art

We now compare our B&P to the best-performing IPs ARF and SF-LEX-I of Jans and Desrosiers (2013). Both formulations have been shown to significantly outperform SF, amongst others, which is in line with small pretests that we conducted. Table 4.2 summarizes the comparison on both benchmark sets `pagination` and `general`. It provides for each approach the percentage of instances solved to



(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.5: Computation time per pricing iteration with (gray) and without (blue) greedy heuristic for an exemplary instance

integer optimality (*%Opt*) and the average computation time in seconds ($t[s]$). Additionally, we report for our B&P the number of new (*New BKS*), confirmed (*Same BKS*), and not found (*Missed BKS*) BKS. For `pagination`, these columns refer to the (previous) BKS reported by Grange *et al.* (2018) and quantify the number of instances for which we have found a better, the same, or a worse solution, respectively. Because the `general` instances have not been considered before and the best solutions of our experiments are trivially new BKS, the corresponding columns are left blank for them. We note that for `general`, 11 BKS are found by both ARF and B&P, three only by ARF, and 136 only by B&P. Note further that in the following the gaps of `pagination` instances are determined using the new BKS, i.e., the minimum value of our best solution and the best solution reported by Grange *et al.* (2018).

Table 4.2 indicates that our B&P clearly outperforms the IP formulations: It solves more than five times more instances in a fraction of the average computation times compared to the IPs. All methods tend to solve fewer instances as the number of items n increases in the `pagination` benchmark. The effect, however, is much more drastic with the IPs. ARF and SF-LEX-I show similar overall performance in terms of the number of solved instances and computation times. While SF-LEX-I performs better than ARF for larger unit-weight instances ($n \geq 45$), a few of the largest `general` instances can be solved with ARF but not with SF-LEX-I. However, for the majority of groups, the average computation times for the IPs are dominated by the time limit. Instances with more than 300 items or elements all run out of memory with SF-LEX-I.

With the computations carried out, we are able to improve on thousands of BKS reported by Grange *et al.* (2018) for the `pagination` benchmark. Overall,

		ARF		SF-LEX-I		Our method				
n	Inst	%Opt	t[s]	%Opt	t[s]	%Opt	t[s]	New BKS	Same BKS	Missed BKS
Panel A: `pagination` instances										
20	624	**100.0**	18.5	97.0	137.8	**100.0**	**0.0**	7	617	0
25	642	87.5	430.0	61.4	907.0	**100.0**	**0.0**	29	613	0
30	648	39.8	1,190.8	26.4	1,414.5	**100.0**	**0.1**	59	589	0
35	648	22.4	1,443.8	13.4	1,581.1	**100.0**	**0.2**	108	540	0
40	648	11.1	1,617.2	9.7	1,641.8	**100.0**	**0.6**	140	508	0
45	648	6.8	1,680.1	7.7	1,668.6	**100.0**	**2.1**	193	455	0
50	648	5.6	1,697.3	7.1	1,683.0	**100.0**	**10.4**	275	373	0
55	648	4.2	1,734.8	6.6	1,693.6	**99.2**	**43.0**	338	309	1
60	648	2.6	1,756.9	9.1	1,663.0	**98.2**	**65.1**	415	230	3
65	648	2.2	1,760.0	11.4	1,645.2	**96.0**	**126.6**	471	172	5
70	648	1.1	1,774.0	15.0	1,620.5	**92.9**	**208.9**	495	146	7
75	648	0.5	1,779.4	10.6	1,674.4	**89.8**	**279.0**	536	104	8
80	648	0.5	1,779.6	6.0	1,729.7	**87.2**	**375.9**	559	79	10
85	648	0.3	1,781.8	4.9	1,750.3	**82.4**	**451.4**	567	73	8
90	648	0.6	1,777.2	4.2	1,755.5	**78.7**	**553.7**	556	69	23
95	648	0.3	1,783.6	2.9	1,765.5	**73.8**	**632.1**	550	72	26
100	648	0.3	1,782.2	3.1	1,769.2	**68.7**	**723.2**	541	67	40
Total	10,986	16.6	1,520.8	17.2	1,538.7	**92.1**	**204.8**	5,839	5,016	131
Panel B: `general` instances										
$\{85, 100\}$	30	0.0	1,800.0	0.0	1,800.0	**70.0**	**614.9**			
$\{185, 200\}$	30	0.0	1,800.0	0.0	1,800.0	**60.0**	**817.3**			
$\{285, 300\}$	30	6.7	1,681.4	0.0	1,800.0	**76.7**	**538.7**			
$\{385, 400\}$	30	3.3	1,740.8	0.0	1,800.0	**80.0**	**540.3**			
$\{485, 500\}$	30	3.3	1,743.6	0.0	1,800.0	**90.0**	**395.0**			
Total	150	2.7	1,749.9	0.0	1,800.0	**75.3**	**581.2**			

Table 4.2: Comparison of our B&P with the IPs ARF and SF-LEX-I of Jans and Desrosiers (2013)

we confirm 5,016 BKS and provide 5,839 new BKS. Only for 131 instances, we are not able to reach the previously reported BKS, which is to some extent caused by memory limits (54 instances, see also Table 4.3). We can also observe that the heuristics used by Grange *et al.* (2018) often find an optimal solution for small instances, but commonly fail to do so for larger n.

4.5.4 Computational Analysis of B&P Algorithm

A more detailed analysis of our B&P is provided in Table 4.3. The additional columns are the number of instances that could not be solved due to memory limitations (OOM), the number of instances without memory issues whose root node could not be solved ($No\ LP$), the average percentage optimality gap of the LP relaxation (Gp^{LP}), the average percentage optimality gap of the LP relaxation excluding optimally solved instances (Gp_u^{LP}), the average percentage optimality gap when reaching the time limit excluding optimally solved instances (Gp_u^{tree}),

and the average number of B&B nodes solved (*Nds*). All unsolved instances are considered with the time limit of 1,800 seconds in the average computation times. We exclude instances with missing LP value in columns Gp^{LP}, Gp_u^{LP}, and Gp_u^{tree}.

Table 4.3 reveals that the times needed for solving the LP relaxations are short for most `pagination` instances with an average of 46.0 seconds and only 10% of instances for which this time is longer than 60 seconds. Only for very few instances, our B&P fails to solve the LP relaxation. For the `general` benchmark, the computation times for solving the LP relaxation are much longer (375.4 seconds on average) and our B&P cannot solve the LP relaxation in 14 out of 150 instances. Overall, the `general` instances appear to be more difficult than the `pagination` instances for our B&P, with less optimal integer solutions obtained (75.3% vs. 92.1%) and longer average computations times (581.2 seconds vs. 204.8 seconds). Compared to the `pagination` benchmark, the proportion of total computational time allocated to solving the LP relaxation is substantially higher in the `general` benchmark. Memory issues seem to consistently occur within the `general` benchmark and for a few of the larger `pagination` instances ($n \geq 55$).

Table 4.3 further reveals that the percentage LP gaps for the `pagination` benchmark are rather large (6.1% on average) and much larger than those of the `general` benchmark (average of 1.4%). However, this may be a bit misleading due to small objective function values. Absolute gaps are generally small for both benchmark sets (see following section). Table 4.3 also shows that the percentage LP gaps of the unsolved instances are huge. Again, this may partly be attributed to the small objective function values. In fact, of the 769 unsolved `pagination` instances (nine instances for `general`) for which an LP value is available, 11 (zero) instances have an absolute gap of strictly less than one, i.e., the BKS is proven to be optimal by the lower bound of our B&P, 668 (six) an absolute gap within $[1, 2)$, 90 (one) an absolute gap within $[2, 3)$, and zero (two) an absolute gap of three or more. We believe that at least for the latter two groups, the BKS are not optimal. We can further observe that gaps improve only marginally in the search tree. This can partly be attributed to the depth-first node selection and partly to the nature of the SUBP itself, which often allows for many symmetric solutions with identical objective function values.

From Table 4.3, we have seen that memory limitations can be an issue for our B&P with 54 `pagination` and 14 `general` instances running out of memory. We, therefore, perform a sensitivity analysis regarding the allowed memory in Table 4.4. Specifically, it reports the change in the number of instances for which the B&P runs out of memory (ΔOOM), the number of instances with solved root nodes (ΔLP), and the number of optimally solved instances (ΔOpt) for different memory limits relative to the baseline of six GB. As expected, the number of instances that exhaust available memory decreases as more memory is allowed. For

n	Inst	Opt	%Opt	OOM	No LP	t[s]	t^{LP}	Gp^{LP}	Gp_u^{LP}	Gp_u^{tree}	Nds
Panel A: `pagination` instances											
20	624	624	100.0	0	0	0.0	0.0	5.5	–	–	1.0
25	642	642	100.0	0	0	0.0	0.0	5.4	–	–	1.1
30	648	648	100.0	0	0	0.1	0.0	5.9	–	–	1.2
35	648	648	100.0	0	0	0.2	0.1	5.5	–	–	1.4
40	648	648	100.0	0	0	0.6	0.3	5.8	–	–	1.7
45	648	648	100.0	0	0	2.1	0.8	6.1	–	–	4.1
50	648	648	100.0	0	0	10.4	2.1	6.1	–	–	9.7
55	648	643	99.2	1	0	43.0	4.5	6.5	21.3	20.5	22.2
60	648	636	98.1	3	0	65.1	8.2	6.0	22.2	21.3	26.5
65	648	622	96.0	4	0	126.6	13.7	6.0	20.0	19.4	44.6
70	648	602	92.9	7	0	208.9	23.4	6.2	17.9	17.4	50.1
75	648	582	89.8	7	0	279.0	34.4	6.2	19.0	18.4	114.3
80	648	565	87.2	9	0	375.9	59.8	6.3	17.4	16.8	97.4
85	648	534	82.4	4	0	451.4	80.5	6.5	17.1	16.7	98.4
90	648	510	78.7	6	6	553.7	133.3	6.6	17.0	16.7	167.8
95	648	478	73.8	6	13	632.1	185.8	6.8	16.8	16.4	119.7
100	648	445	68.7	7	21	723.2	232.8	7.1	16.8	16.5	182.3
Total	10,986	10,123	92.1	54	40	204.8	46.0	6.1	17.3	16.9	55.4
Panel B: `general` instances											
$\{85, 100\}$	30	21	70.0	5	1	614.9	186.5	4.7	13.5	13.2	73.4
$\{185, 200\}$	30	18	60.0	2	5	817.3	531.4	1.5	4.7	4.7	588.8
$\{285, 300\}$	30	23	76.7	1	5	538.7	462.3	0.5	5.5	5.5	8.8
$\{385, 400\}$	30	24	80.0	3	3	540.3	388.9	0.2	–	–	1.6
$\{485, 500\}$	30	27	90.0	3	0	395.0	307.7	0.1	–	–	0.9
Total	150	113	75.3	14	14	581.2	375.4	1.4	7.9	7.8	134.7

Table 4.3: Detailed results of our B&P algorithm

Memory	ΔOOM	ΔLP	ΔOpt
Panel A: `pagination` instances			
2 GB	93	−1	−16
4 GB	29	0	−5
6 GB	0	0	0
8 GB	−1	1	0
10 GB	−2	2	0
12 GB	−27	27	2
Panel B: `general` instances			
2 GB	6	0	0
4 GB	1	1	0
6 GB	0	0	0
8 GB	−2	2	2
10 GB	−4	4	3
12 GB	−4	4	3

Table 4.4: B&P results for different amounts of memory allowed

the `pagination` instances, the number of optimally solved instances, however, increases only marginally. For the `general` instances, three out of the 14 instances can be solved to integer optimality when allowing at least 10 GB of memory.

In Appendix 4.G, we perform detailed analyses to investigate which characteristics influence the complexity of an instance for our B&P.

4.5.5 Analysis of Lower Bounds

For the BP, it is known that the LP relaxation of the SPF provides very tight lower bounds. In fact, almost all instances satisfy the IRUP, i.e., rounding up the optimal LP relaxation value to the closest integer results in the optimal integer solution value. Whether or not the MIRUP, i.e., the optimal integer solution value is not greater than the corresponding optimal LP relaxation value rounded up plus one, holds for any BP instance is – as far as we know – still an open question. We refer to (Delorme *et al.* 2016) and references therein for details on the IRUP and MIRUP in the context of BP.

To the best of our knowledge, no evaluation of the IRUP or MIRUP has yet been performed for the SUBP. We numerically investigate both properties on all optimally solved `pagination` and `general` instances in Table 4.5. We can observe that the MIRUP holds for all optimally solved SUBP instances. In over 92.5% of cases, the IRUP applies. This implies that for the vast majority of instances, the key task of the B&P beyond the solution of the root node is to find an optimal primal solution. To analyze the potential of improved strategies like, e.g., better

Class	Inst	Opt	MIRUP	IRUP
`pagination`	10,986	10,123	10,123	9,359
`general`	150	113	113	111
All	11,136	10,236	10,236	9,470

Table 4.5: Analysis of IRUP and MIRUP

primal heuristics or diving heuristics, we examine in Appendix 4.H the effect of using different initial upper bounds on our B&P.

Compared to the BP, SUBP instances appear to violate the IRUP much more frequently. In Appendix 4.I, we present details on two of the smallest instances that we could find that do not satisfy the IRUP.

4.6 Conclusions

In this paper, we study the *set-union bin packing problem* (SUBP) which generalizes the well-known *bin packing problem* (BP) and has important applications in various fields. We propose an exact *branch-and-price* (B&P) algorithm that is applicable for solving instances with general element weights. The corresponding column generation pricing problem is a *set-union knapsack problem* (SUKP). We present and explore different exact solution approaches to the pricing problem that are based on three alternative formulations of the SUKP: an *integer programming* (IP) formulation, an item-based *shortest path problem with resource constraints* (SPPRC), and an element-based SPPRC. The overall best B&P variant combines an upfront greedy pricing heuristic with a labeling algorithm for the item-based SPPRC that does not apply any dominance but relies on strong completion bounds.

We highlight the competitiveness of our proposed B&P in an extensive computational campaign on the large unit-weight SUBP benchmark of Grange *et al.* (2018) comprising 10,986 instances and on new large-scale general-weight instances based on the SUKP benchmark of He *et al.* (2018). Our B&P by far outperforms the state-of-the-art IP approaches of Jans and Desrosiers (2013) on both instance classes. Overall, we are able to solve to optimality 92% of the considered instances, with an average computation time of around 200 seconds. Only a small fraction of the considered instances has been solved to proven optimality before. We provide 5,839 new *best-known solutions* (BKS) for the benchmark by Grange *et al.* (2018) and confirm all remaining BKS except for 131 instances. Furthermore, we analyze which characteristics influence the difficulty of an instance for our B&P. Finally, we investigate the lower bounds provided by the set-partitioning formulation of the SUBP and observe that 92.5% of the optimally solved instances satisfy the *integer*

round up property (IRUP) while all of them satisfy the *modified* IRUP (MIRUP).

We conclude by mentioning three possible avenues for future research. First, there exist numerous practically relevant extensions of the SUBP including global constraints on element availabilities, cardinality constraints on the number of items per bin, heterogeneous bin sizes and costs, alternative objective functions, or multi-level integrated optimization problems in which the SUBP appears as a subproblem. While modified variants of our B&P seem promising for solving the former types of extensions, integrated solution approaches to the latter multi-level problems may also benefit from the proposed solution techniques. Second, to exploit the usually very tight lower bounds, new strategies to obtain high quality primal solutions fast should be explored. Third, theoretical and/or additional computational analyses regarding the MIRUP of the SUBP could provide valuable insights and might even be helpful to further explore the MIRUP for the BP.

Bibliography

Adjiashvili, D., Bosio, S., and Zemmer, K. (2015). Minimizing the number of switch instances on a flexible machine in polynomial time. *Operations Research Letters*, **43**(3), 317–322.

Arulselvan, A. (2014). A note on the set union knapsack problem. *Discrete Applied Mathematics*, **169**, 214–218.

Baldacci, R., Coniglio, S., Cordeau, J.-F., and Furini, F. (2024). A numerically exact algorithm for the bin-packing problem. *INFORMS Journal on Computing*, **36**(1), 141–162.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Burger, A. P., Jacobs, C., van Vuuren, J. H., and Visagie, S. E. (2015). Scheduling multi-colour print jobs with sequence-dependent setup times. *Journal of Scheduling*, **18**, 131–145.

Calmels, D. (2019). The job sequencing and tool switching problem: State-of-the-art literature review, classification, and trends. *International Journal of Production Research*, **57**(15-16), 5005–5025.

Campêlo, M., Campos, V. A., and Corrêa, R. C. (2008). On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, **156**(7), 1097–1111.

Crama, Y. and Oerlemans, A. G. (1994). A column generation approach to job grouping for flexible manufacturing systems. *European Journal of Operational Research*, **78**(1), 58–80.

Crama, Y., Moonen, L. S., Spieksma, F. C., and Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, **182**(2), 952–957.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Denizel, M. (2003). Minimization of the number of tool magazine setups on automated machines: A Lagrangean decomposition approach. *Operations Research*, **51**(2), 309–320.

Desrosiers, J., Jans, R., and Adulyasak, Y. (2013). Improved column generation algorithms for the job grouping problem. *Les Cahiers du GERAD*. G-2013-26.

Dror, M. and Haouari, M. (2000). Generalized Steiner problems and other variants. *Journal of Combinatorial Optimization*, **4**, 415–436.

Gokgur, B. and Özpeynirci, S. (2022). Minimization of number of tool switching instants in automated manufacturing systems. *Journal of Science*, **35**(1), 113–130.

Goldschmidt, O., Nehme, D., and Yu, G. (1994). Note: On the set-union knapsack problem. *Naval Research Logistics*, **41**(6), 833–842.

Goldschmidt, O., Hochbaum, D. S., Hurkens, C., and Yu, G. (1996). Approximation algorithms for the k-clique covering problem. *SIAM Journal on Discrete Mathematics*, **9**(3), 492–509.

Grange, A., Kacem, I., and Martin, S. (2018). Algorithms for the bin packing problem with overlapping items. *Computers & Industrial Engineering*, **115**, 331–341.

Grange, A., Kacem, I., Martin, S., and Minich, S. (2023). Fully polynomial time approximation scheme for the pagination problem with hierarchical structure of tiles. *RAIRO – Operations Research*, **57**(1), 1–16.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Gschwind, T., Irnich, S., Furini, F., and Calvo, R. W. (2021). A branch-and-price framework for decomposing graphs into relaxed cliques. *INFORMS Journal on Computing*, **33**(3), 1070–1090.

He, Y., Xie, H., Wong, T.-L., and Wang, X. (2018). A novel binary artificial bee colony algorithm for the set-union knapsack problem. *Future Generation Computer Systems*, **78**, 77–86.

Heßler, K., Gschwind, T., and Irnich, S. (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, **271**(2), 401–419.

Hirabayashi, R., Suzuki, H., and Tsuchiya, N. (1984). Optimal tool module design problem for NC machine tools. *Journal of the Operations Research Society of Japan*, **27**(3), 205–229.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer Science & Business Media, Boston.

Izumi, T., Yokomaru, T., Takahashi, A., and Kajitani, Y. (1998). Computational complexity analysis of set-bin-packing problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **81**(5), 842–849.

Jans, R. and Desrosiers, J. (2010). Binary clustering problems: Symmetric, asymmetric and decomposition formulations. *Les Cahiers du GERAD*. G-2010-44.

Jans, R. and Desrosiers, J. (2013). Efficient symmetry breaking formulations for the job grouping problem. *Computers & Operations Research*, **40**(4), 1132–1142.

Junglas, D. (2007). *Optimised grid-partitioning for block structured grids in parallel computing*. Ph.D. thesis, Fachbereich Mathematik, Technische Universität Darmstadt, Darmstadt, Germany.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin, Heidelberg.

Kochetov, Y. and Kondakov, A. (2017). VNS matheuristic for a bin packing problem with a color constraint. *Electronic Notes in Discrete Mathematics*, **58**, 39–46.

Konak, A. and Kulturel-Konak, S. (2007). An ant colony optimization approach to the minimum tool switching instant problem in flexible manufacturing system. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 43–48, Honolulu, HI. IEEE.

Konak, A., Kulturel-Konak, S., and Azizoğlu, M. (2008). Minimizing the number of tool switching instants in flexible manufacturing systems. *International Journal of Production Economics*, **116**(2), 298–307.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Locatelli, A. (2023). Optimization methods for knapsack and tool switching problems. *4OR*, **21**(4), 715–716.

Marvizadeh, S. Z. and Choobineh, F. (2013). Reducing the number of setups for CNC punch presses. *Omega*, **41**(2), 226–235.

Ryan, D. M. and Foster, B. A. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 269–280. North-Holland Publishing Company, Amsterdam.

Shirazi, R. and Frizelle, G. (2001). Minimizing the number of tool switches on a flexible machine: An empirical study. *International Journal of Production Research*, **39**(15), 3547–3560.

Sindelar, M., Sitaraman, R. K., and Shenoy, P. (2011). Sharing-aware algorithms for virtual machine colocation. In *Proceedings of the twenty-third annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 367–378, San Jose, CA. Association for Computing Machinery.

Tang, C. S. and Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part II: Minimization of the number of switching instants. *Operations Research*, **36**(5), 778–784.

Wahlen, J. and Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, **57**(3), 756–777.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2020). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*, **32**(2), 428–443.

Wei, Z. (2021). *Optimization algorithms for two knapsack problems*. Ph.D. thesis, Optimization and Control [math.OC], Université d'Angers, Angers, France.

Appendix

4.A Acceleration Strategies for Pricing Problem Solution

In this section, we discuss acceleration techniques to speed-up the solution of the pricing problem.

Greedy SUKP Heuristic

To quickly generate bins with negative reduced costs, we apply a greedy heuristic following (Arulselvan 2014) to solve the SUKP pricing problem. It can be called before performing any of the described exact solution approaches.

The heuristic proceeds as follows. Starting with a bin comprising a single item only, additional items are added iteratively. Arulselvan (2014) suggests to choose in each iteration an item $i \in I$ with the largest profit in relation to the sum of its required elements weights divided by their frequency, i.e., $\frac{\pi_i}{\sum_{e \in E_i} \frac{w_e}{f_e(I)}}$. In pretests, we found that taking into account only the weights of those elements that have not yet been included is more expedient and makes the integration of frequencies dispensable. Therefore, we augment each bin b with an item $i^* \in I \setminus b$ that holds the maximum relative profit taking into consideration only the weights of the required elements that are not yet in the bin:

$$i^* = \arg\max_{i \in I \setminus b} \quad \frac{\pi_i}{\sum_{e \in E_i \setminus E(b)} w_e}. \tag{4.13}$$

After each inclusion, this ratio is recalculated for all remaining items and the procedure is repeated until no more capacity-feasible addition to b is possible.

In each pricing iteration, the greedy heuristic is run several times, once for each item constituting the initial bin. Note that we store all considered bins to prevent construction of identical bins and to speed up the procedure. All identified bins with negative reduced costs are added to the RMP.

Premature Termination of Exact Pricers

We can easily modify all exact pricing algorithms to identify and return bins with negative reduced costs in a heuristic fashion. There is no need to run the full algorithm in each execution. To balance the computational effort and the number of negative reduced-cost bins created in each pricing iteration, we proceed as follows. For the MIP-solver based approach, we prematurely terminate the solution process as soon as a predefined number of 15 bins with negative reduced costs is found. For the labeling approaches, note that any feasible 0-v-path P_L in G (or 0-$\hat{v}$-path in $\hat{G}$) represented by label L defines a feasible bin, which can directly be

returned to the RMP without completion to a 0-n-path (0-m-path). For each vertex, we count the number of created labels with negative reduced costs (recall that labels are created and extended vertex-by-vertex). As soon as this number reaches a threshold $\mathcal{K}$, the labeling algorithm terminates prematurely and all bins with negative reduced costs are added to the RMP. We set $\mathcal{K} = 0.35n$ in item-based labeling and $\mathcal{K} = 0.35m$ in element-based labeling.

Storing Elements-of-Items Relations

The frequent evaluation of the capacity consumption and the examination of realizable items, e.g., within the REFs (4.6) or (4.11), is a computationally expensive part of our algorithms. During the B&P procedure, it can be expected that for many bins $b \in \Omega$ this needs to be evaluated multiple times. We use a hash table implementation to allow a fast retrieval in amortized constant time of the relation between a given set of items and the corresponding required set of elements. This requires storing in the hash table the key-value pairs $(I(b), E(b))$ whenever $E(b)$ is evaluated for the first time for a bin b defined by $I(b)$.

Furthermore, we exploit that for unit weight instances the capacity consumption is equal to the number $|E(b)|$ of elements in a bin b, which can be retrieved more effectively than computing the summed weights w_e of the elements $e \in E(b)$.

Sorting of Vertices in Labeling

When constructing the linear graphs for the SPPRC representations of the pricing problem, any sorting of the items and elements can be employed. However, the sorting has a significant impact on the solution time of the pricing problem. In the item-based graph G, we sort the items $i \in I$ non-increasingly by their profit π_i. In the element-based graph $\hat{G}$, we sort the elements $e \in E$ non-increasingly by the sum of normalized profits of their comprising items relative to their frequency, i.e.,

$$\frac{\sum_{i \in I : e \in E_i} \frac{\pi_i}{\sum_{e \in E_i} w_e}}{f_e(I)}. \tag{4.14}$$

With these sortings, negative reduced-cost bins can often be identified early in the labeling process, allowing for early termination. In addition, labels with positive reduced costs tend to be discarded early due to the stronger completion bounds resulting from these sortings.

4.B Algorithm Design Choices

In this section, we give some details on additional design choices made in our B&P algorithm.

RMP Initialization and Upper Bounds

Before the actual CG procedure starts, we run a SUBP heuristic following the MIMU approach by Tang and Denardo (1988) to hand over an initial set of feasible bins $\Omega' \subset \Omega$ and a first *upper bound* (UB) to the RMP. The idea of this heuristic is to fill bins one at a time by iteratively selecting items $i \in R$ that maximize the intersection of the item's required elements and elements that are already required in the bin (*maximum intersection* part). In case of a tie, the authors choose the item that minimizes the number of additionally required elements (*minimum union* part). The selected item is added to the current bin and discarded from R. If no more remaining item $i \in R$ can be feasibly included in the current bin, another bin is opened and the procedure is repeated until R is empty. Bins are initialized with an item requiring the most elements among all remaining items $i \in R$, where initially $R = I$.

We apply a randomized version of the MIMU heuristic. The intersection values are multiplied by a factor randomly drawn from the interval $[0.85, 1.15]$, which makes practically redundant the minimum-union part. Note that we cover the general-weight case by considering the element-specific weights w_e instead of simply counting the elements as proposed by Tang and Denardo (1988). The heuristic is run four times and all bins contained in any of the heuristic solutions are added to Ω'. The minimum number of bins needed in any of the solutions is set as initial UB.

To further improve the UB during the B&P process, we employ at each node a standard restricted master heuristic solving Formulation (4.4) over the current set of bins $\Omega' \subset \Omega$ to integer optimality with a MIP solver. A time limit of ten seconds is given to the solver for each run.

Initial Lower Bounds

We initialize the RMP with a *lower bound* (LB) on the number of bins by adding the corresponding inequality $\sum_{b \in \Omega} \lambda_b \geq LB$. The value LB is obtained by a combination of relaxations. First, a straightforward bound arises from ceiling the total element weights divided by the bin capacity, that is $LB_1 = \left\lceil \frac{\sum_{e \in E} w_e}{Q} \right\rceil$. The second bound results from the *sweeping procedure* of Tang and Denardo (1988). A pair of items is said to be *compatible* if the combined weight does not exceed the capacity. In each sweeping iteration, a seed item is defined as the item $i \in R$ that

is compatible with fewest other items in R, where initially $R = I$. The seed item and all its compatible items define a (not necessarily feasible) bin and are removed from R. This procedure is repeated until R is empty. The number of bins obtained is denoted LB_2. Third, the *modified sweeping procedure* (Crama and Oerlemans 1994) is executed. After each sweeping iteration, the trivial size bound considering the elements $e \in E_i$ of all remaining items $i \in R$ is added to the number of already created bins from the sweeping procedure. We obtain multiple lower bounds (one for each sweeping iteration) with this procedure and set the maximum value as LB_3. The overall LB is given by $LB = \max\{LB_1, LB_2, LB_3\}$.

Strengthening of Lower Bounds

To strengthen the lower bounds, we experimented with adding valid inequalities in the form of *subset-row cuts* (SRCs, Jepsen *et al.* 2008) and *capacity cuts* (CCs, Baldacci *et al.* 2008), which have proven beneficial in improving the performance of B&P-based approaches to related problems (e.g., Wei *et al.* 2020, Wahlen and Gschwind 2023). In pretests, we found that violated SRCs could be separated regularly and their addition did increase the lower bound. On the other hand, only very few violated CCs could be separated at all, despite a considerable computational effort. Overall, neither the integration of SRCs nor of CCs showed a positive effect on our approach so that we do not include any valid inequalities in our B&P.

Branching Strategy

The computational analysis of our B&P revealed that improving the primal bound seems to be the primary task within the B&B tree (see Sections 4.5.4 and 4.5.5); strengthening the dual bound seems of secondary importance for most instances. Therefore, we use a depth-first node-selection strategy. Still, in pretests we found that the application of strong branching as detailed in the following proved to be beneficial.

In each B&B node, a candidate set of item pairs (i, j) whose f_{ij}-values are closest to 0.6 is identified. For each pair, a rough evaluation of the two child nodes is performed by solving the RMP with the corresponding branching constraint but without any CG. The decision on the branching variables to be finally selected is made according to the product rule (Achterberg 2007). At the root node, the maximum size of the candidate set is 25, and for each level of the B&B tree, the size is reduced by two. We also experimented with incorporating the item weights into the selection of the branching candidates as proposed by Heßler and Irnich (2022a). However, pretests have not revealed any significant benefit of this idea.

4.C Modification of Item-based SPPRC Graph for Branching

We identify a Ryan-and-Foster branching decision by the set $I = \{i, j\}$ of two items. The type of decision, *separate* or *together*, is irrelevant for now. Denote by $\mathcal{I} = \{I_1, \ldots, I_p\}$ the set of active branching decisions at a given B&B node. Let $\mathcal{I}^1, \ldots, \mathcal{I}^q$ be a partition of $\mathcal{I}$ into subsets, i.e., groups of branching decisions, such that the different subsets do not overlap with respect to the items involved, i.e, $(\bigcup_{I \in \mathcal{I}^g} I) \cap (\bigcup_{I \in \mathcal{I}^h} I) = \varnothing$ applies to all pairs $g, h \in \{1, ..., q\}, g \neq h$, while each individual subset consists of overlapping branching decisions. The branching decisions of a set $\mathcal{I}^g = \{I_1^g, \ldots, I_r^g\}$ overlap if they can be arranged such that $(I_1^g \cup \cdots \cup I_{h-1}^g) \cap I_h^1 \neq \varnothing$ for all $h \in \{1, ..., r\}$. For each $\mathcal{I}^g, g \in \{1, \ldots, q\}$, denote by $I(\mathcal{I}^g) = \bigcup_{I \in \mathcal{I}^g} I$ the set of items involved. Furthermore, let $\mathcal{I}_0(\mathcal{I}^g) = \varnothing, \mathcal{I}_1(\mathcal{I}^g), \ldots, \mathcal{I}_s(\mathcal{I}^g)$ be all feasible combinations (i.e., subsets) of the items in $I(\mathcal{I}^g)$ such that the branching decisions and the bin capacity are respected. In the resulting graph G, each set $I(\mathcal{I}^g)$ of items corresponding to a group of branching decisions $\mathcal{I}^g$ is represented by a single vertex v. For each feasible combination of items $\mathcal{I}_k(\mathcal{I}^g), k \in \{0, \ldots, s\}$, an arc a_v^k from vertex $v-1$ to vertex v is created. The components associated with each arc a_v^k are $I_v^k = \mathcal{I}_k(\mathcal{I}^g)$, $E_v^k = \bigcup_{i \in \mathcal{I}_k(\mathcal{I}^g)} E_i$, and $\pi_v^k = \sum_{i \in \mathcal{I}_k(\mathcal{I}^g)} \pi_i$. Each of the items $i \in I \setminus \bigcup_{I \in \mathcal{I}} I$ that is not involved in any branching decision is still associated with a single vertex and two ingoing arcs in G (see Section 4.4.1.2). Note that after branching, the vertices in graph G are sorted non-increasingly by a generalized profit value that is based on the maximum profit of all ingoing arcs.

Example 4.5 (continued)**.** *Consider again the example SUBP instance and dual prices $\pi_{i_1} = 0.3$, $\pi_{i_2} = 0.5$, $\pi_{i_3} = 0.4$, and $\pi_{i_4} = 0.2$. We assume that $\mathcal{I}$ comprises the two active* separate *branching decisions $\{i_1, i_2\}$ and $\{i_2, i_3\}$, resulting in one group $\mathcal{I}^1$ of branching decisions with involved items $I(\mathcal{I}^1) = \{i_1, i_2, i_3\}$. Item i_4 is not involved in any branching decision and is therefore represented by an individual vertex. Figure 4.6 depicts the modified graph G showing all arcs a_v^k with their components (I_v^k, E_v^k, π_v^k). Consider vertex $\mathcal{I}^1$ associated with items $I(\mathcal{I}^1) = \{i_1, i_2, i_3\}$. The five ingoing arcs represent all subsets of items from the set $I(\mathcal{I}^1)$ that respect the mutual branching decisions and the bin capacity $Q = 3$, i.e., that can be feasibly included in a bin. Now consider arc $(\{i_1, i_3\}, \{e_2, e_4, e_5\}, 0.7)$ corresponding with the inclusion of the items i_1 and i_3. It is associated with the set of elements $\{e_2, e_4, e_5\}$ and a dual price of $0.3 + 0.4 = 0.7$. Bin b_1 comprising items i_1 and i_4 from Figure 4.1 (which is still feasible despite the branching decisions) is represented by the path having bold arcs.*

In the greedy pricing heuristic, the branching decisions are taken into account by considering the sets $\mathcal{I}_1(\mathcal{I}^g), \ldots, \mathcal{I}_s(\mathcal{I}^g)$ instead of the single items.

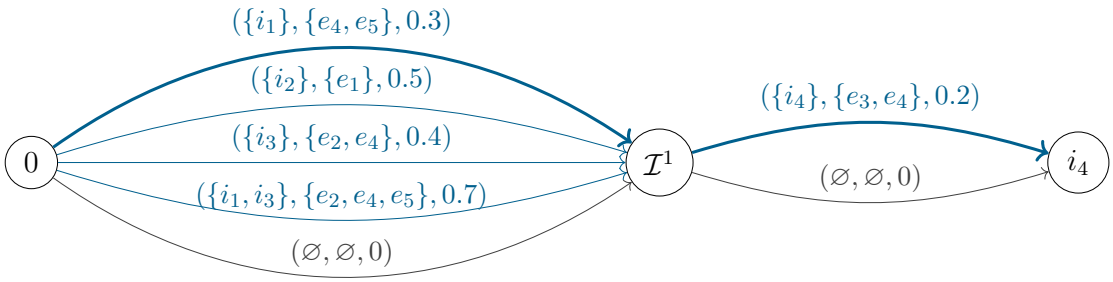


Figure 4.6: Modified linear directed multigraph G with Ryan-and-Foster *separate* branching decisions $\{i_1, i_2\}$ and $\{i_2, i_3\}$

4.D Benchmark Instances

The benchmark by Grange *et al.* (2018) comprises six unit-weight instances for each combination of capacity $Q \in \{15, 20, \ldots, 50\}$, number of elements $m \in \{Q + 5, Q + 10, \ldots, 100\}$, and number of items $n \in \{20, 25, \ldots, 100\}$. For details on their generation, we refer to (Grange *et al.* 2018). Note that five (Q, m, n)-combinations do not exist, in particular, $(15, 85, 20)$, $(15, 90, 20)$, $(15, 95, 20)$, $(15, 100, 20)$, and $(15, 100, 25)$, resulting in a total of 10,986 instances.

In order to assess the performance of our B&P on instances with general weights, we use the SUKP instances by He *et al.* (2018) and interpret them as SUBP instances by reducing the capacity and omitting profits. The original instances are defined by the characteristics number of items n, number of elements m, density $\alpha = \frac{\sum_{i \in I} |E_i|}{nm}$, and capacity ratio $\beta = \frac{Q}{\sum_{e \in E} w_e}$. The benchmark consists of three classes describing the relationship between n and m, namely $n > m$, $n = m$, and $n < m$. The larger of the two values n and m (in case of equality this applies to both) is in $\{100, 200, \ldots, 500\}$, and the smaller one is chosen to be 15 less. Each of the resulting (n, m)-pairs is combined with the two (α, β)-pairs $(0.1, 0.75)$ and $(0.15, 0.85)$, leading to a total of 30 instances. The weights w_e of the individual elements $e \in E$ are within the interval $[11, 331]$, resulting in bin capacities Q reaching from 11,000 to more than 70,000. To obtain reasonable SUBP instances, we redefine for each instance the capacity value depending on the item with the largest total weight and the largest individual element of the instance. More precisely, we set $Q = \max_{i \in I} \sum_{e \in E_i} w_e + \gamma \max_{e \in E} w_e$ for $\gamma \in \{0, 5, 10, 15, 20\}$. Combining the given values for (n, m) and α with the considered values of γ, we obtain a set of 150 general-weight SUBP instances.

4.E Comparison of Completion Bounds

Table 4.6 analyzes the impact of different completion bounds on our item-based labeling algorithm without dominance. Note that similar trends can be observed also for the other labeling variants (with dominance, element-based labeling). When applying the upfront greedy heuristic and looking at individual bounds, Table 4.6 reveals that B_1 and B_2 are clearly superior to the others, with B_4 being not competitive. While B_2 provides slightly faster average computation times, it starts to struggle for larger instances where B_1 is able to solve more LP relaxations. Given these insights, we further investigated two promising combinations of bounds, namely using the KP-based bound B_1 with the upfront application of the computationally cheaper bounds B_2 or B_3. The results indicate that the combined variants do not increase the percentage of solved instances beyond the level achieved by using only B_1. However, applying B_2 prior to B_1 leverages the strengths of both bounds, producing the most favorable overall outcomes. Consequently, we implement the combination of B_2 and B_1 in our B&P.

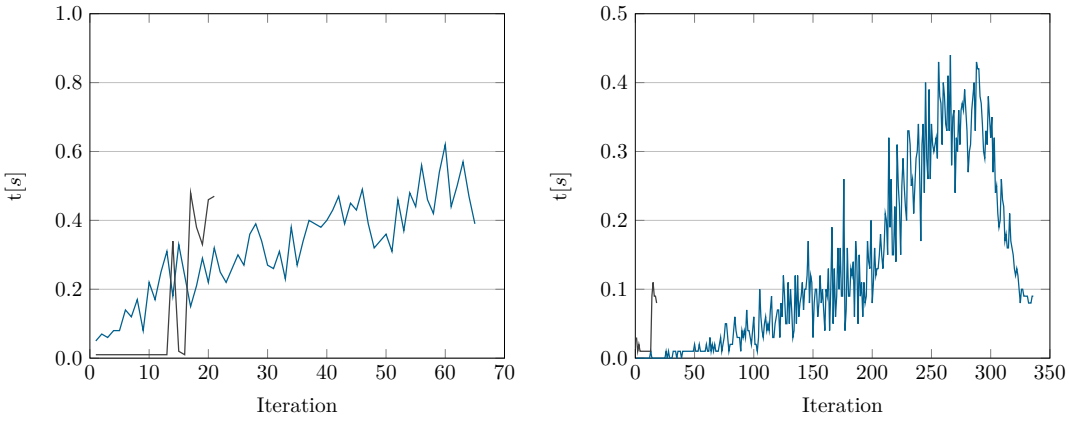
Table 4.6 further reveals that when the greedy heuristic is employed, which means that only a few exact labeling-based pricings are necessary and the most costly exact pricings are replaced by the heuristic (see Figure 4.5 and Appendix 4.F), the item-based labeling algorithm is rather robust with respect to reasonable completion bounds. Without the greedy heuristic, we see that the strong completion bounds B_1 are crucial for the performance of the labeling algorithm and the B&P; as indicated also for the largest instances when applying the greedy heuristic.

	B_1		B_2		B_3		B_4		B_2 and B_1		B_3 and B_1	
n	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}	$\%Sol^{LP}$	t^{LP}
Panel A: No greedy heuristic												
20	**100.0**	**0.0**	50.0	900.0	50.0	900.0	50.0	902.4	**100.0**	**0.0**	**100.0**	**0.0**
25	**100.0**	**0.2**	50.0	900.0	50.0	900.0	50.0	918.0	**100.0**	**0.2**	**100.0**	**0.2**
30	**100.0**	**0.8**	50.0	900.1	50.0	900.1	49.2	961.4	**100.0**	**0.8**	**100.0**	**0.8**
35	**100.0**	**4.5**	50.0	900.5	50.0	900.5	46.4	1,019.5	**100.0**	4.6	**100.0**	4.8
40	**99.9**	**21.4**	50.0	902.3	50.0	902.5	42.2	1,118.1	**99.9**	21.8	**99.9**	22.3
45	**99.2**	**61.0**	50.0	908.5	49.9	909.5	36.1	1,221.7	**99.2**	61.6	**99.2**	63.2
50	**96.9**	**118.0**	49.6	922.0	49.3	921.9	33.0	1,267.4	**96.9**	118.8	96.6	120.0
55	**93.5**	**199.3**	49.0	940.2	48.2	941.9	30.0	1,313.4	**93.5**	199.8	93.4	202.5
60	91.7	**262.9**	48.1	958.5	47.2	961.3	28.5	1,327.0	**91.8**	263.8	91.5	267.1
65	**89.2**	**332.6**	47.7	975.8	46.3	978.7	28.0	1,343.0	89.0	334.6	88.7	338.0
70	**84.6**	**425.6**	46.7	1,003.9	44.6	1,009.2	27.2	1,364.3	84.3	427.1	84.1	430.5
75	**81.3**	**497.2**	45.8	1,029.9	43.0	1,038.2	26.9	1,379.6	80.9	497.8	80.4	502.5
80	**76.4**	**589.5**	44.2	1,055.5	40.9	1,073.8	26.4	1,405.6	75.9	591.4	75.3	595.3
85	**72.7**	**635.5**	43.6	1,074.4	39.7	1,094.9	25.8	1,427.2	72.4	636.2	71.5	640.6
90	**68.4**	**701.6**	41.3	1,108.8	38.0	1,123.9	25.0	1,445.4	67.9	702.5	67.4	706.3
95	**66.2**	**742.3**	39.4	1,136.7	37.0	1,141.7	24.1	1,470.6	65.4	743.8	64.8	747.7
100	**62.7**	**788.3**	38.5	1,160.7	36.0	1,159.0	23.7	1,495.9	61.1	788.8	61.0	791.9
Tot.	**87.2**	**317.4**	46.8	984.6	45.7	984.1	33.6	1,258.6	86.9	318.1	86.7	320.5
Panel B: With greedy heuristic												
20	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	50.0	900.1	**100.0**	**0.0**	**100.0**	**0.0**
25	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	50.0	900.3	**100.0**	**0.0**	**100.0**	**0.0**
30	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	50.0	900.7	**100.0**	**0.0**	**100.0**	**0.0**
35	**100.0**	**0.1**	**100.0**	**0.1**	**100.0**	**0.1**	50.0	901.4	**100.0**	**0.1**	**100.0**	**0.1**
40	**100.0**	0.3	**100.0**	**0.2**	**100.0**	0.3	50.0	903.2	**100.0**	0.3	**100.0**	0.3
45	**100.0**	0.8	**100.0**	**0.6**	**100.0**	0.8	50.0	906.9	**100.0**	0.8	**100.0**	0.8
50	**100.0**	2.1	**100.0**	**1.7**	**100.0**	2.3	50.0	915.0	**100.0**	2.1	**100.0**	2.2
55	**100.0**	4.5	**100.0**	**3.5**	**100.0**	5.1	50.0	929.3	**100.0**	4.5	**100.0**	4.6
60	**100.0**	8.2	**100.0**	**7.0**	99.5	17.9	49.9	950.3	**100.0**	8.2	**100.0**	8.5
65	**100.0**	13.8	**100.0**	**11.9**	98.8	35.5	49.9	976.5	**100.0**	13.7	**100.0**	14.4
70	**100.0**	23.3	**100.0**	**21.1**	97.2	71.5	49.4	1,017.4	**100.0**	23.4	**100.0**	24.5
75	**100.0**	34.8	**100.0**	**31.6**	94.6	121.9	48.2	1,052.3	**100.0**	34.4	**100.0**	36.5
80	**100.0**	60.4	99.9	68.7	91.5	183.9	45.8	1,106.3	**100.0**	**59.8**	**100.0**	63.7
85	**100.0**	82.1	98.8	92.9	89.2	232.1	44.1	1,146.0	**100.0**	**80.5**	**100.0**	86.5
90	**99.1**	136.1	98.0	**131.0**	84.0	331.6	42.5	1,181.0	**99.1**	133.3	98.8	141.6
95	**98.2**	191.0	95.4	**182.8**	78.7	424.6	40.3	1,218.3	97.8	185.8	97.8	196.2
100	96.6	240.9	94.4	**211.1**	73.0	520.3	37.9	1,258.4	**96.9**	232.8	96.6	245.8
Tot.	**99.6**	47.1	99.2	**45.1**	94.5	114.9	47.5	1,009.9	**99.6**	46.0	**99.6**	48.7

Table 4.6: Summary results for different completion bounds of our B&P for `pagination` instances

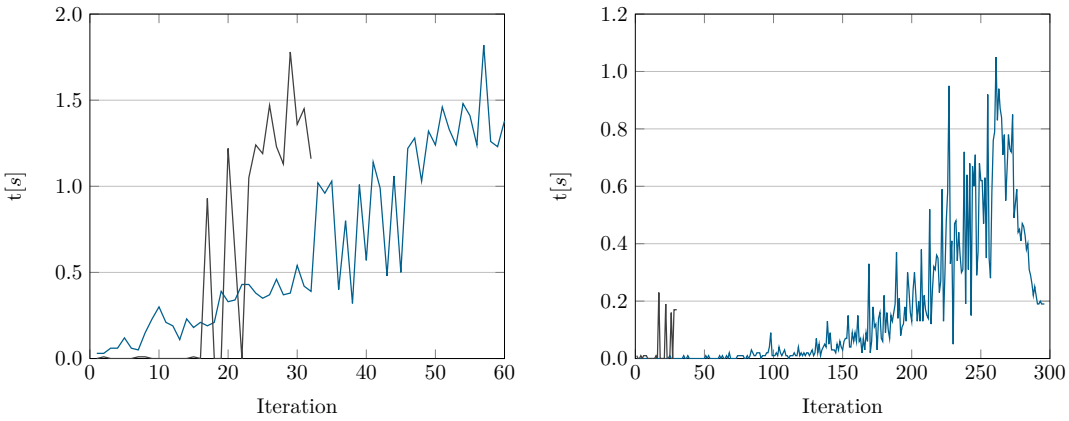
4.F Detailed CG Process for Selected Instances

In Figures 4.7–4.11, we provide the computation times per pricing instance using the IP and item-based labeling both with and without the greedy heuristic as examined in Section 4.5.2 for five additional `pagination` instances.



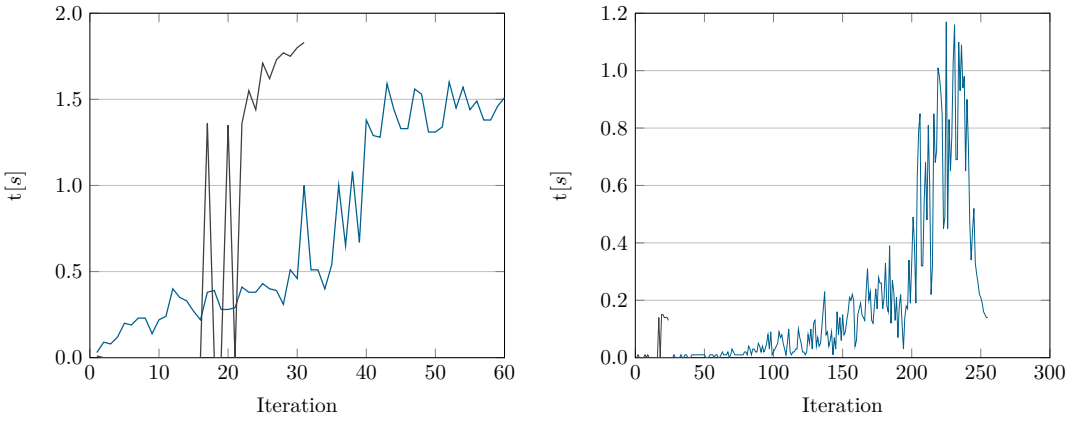
(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.7: Computation time per pricing iteration with (gray) and without (blue) heuristic for the second instance with $Q = 15, m = 20, n = 70$



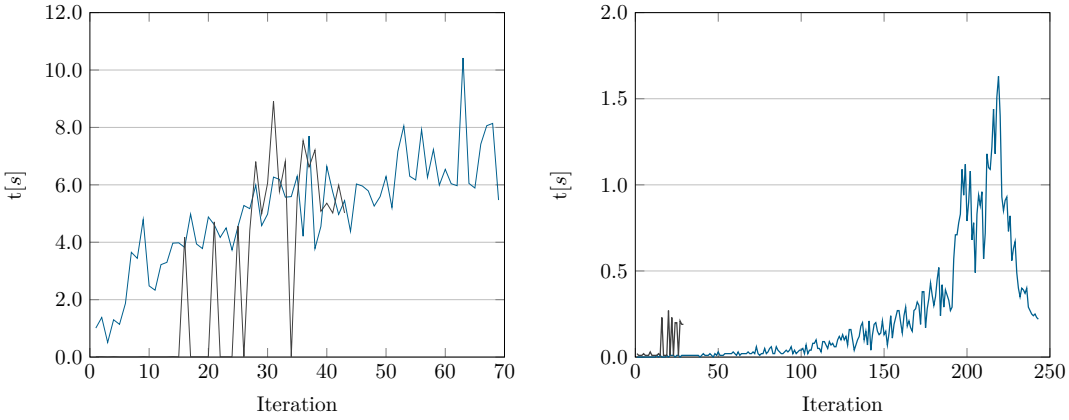
(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.8: Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 20$, $m = 35$, $n = 50$



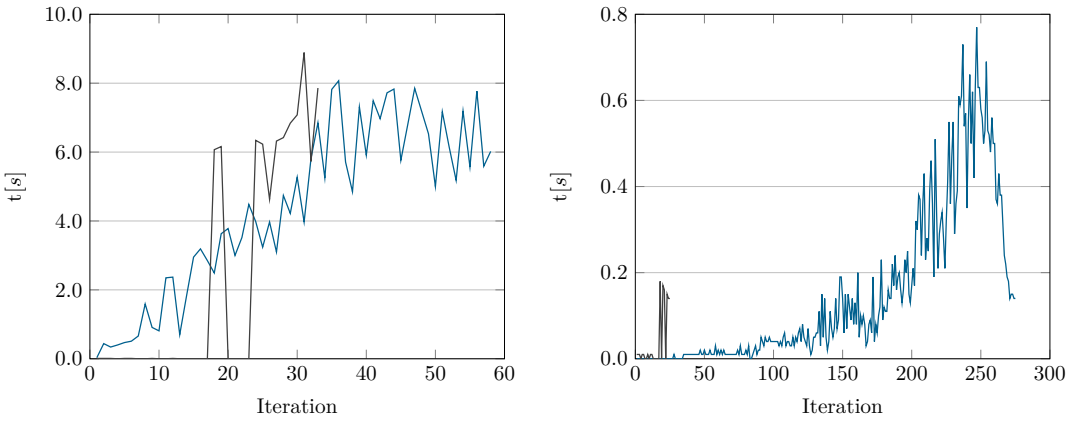
(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.9: Computation time per pricing iteration with (gray) and without (blue) heuristic for the third instance with $Q = 25$, $m = 50$, $n = 50$



(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.10: Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 35$, $m = 40$, $n = 85$



(a) IP. **(b)** Item-based labeling without dominance.

Figure 4.11: Computation time per pricing iteration with (gray) and without (blue) heuristic for the fifth instance with $Q = 50$, $m = 55$, $n = 60$

4.G Influence of Instance Characteristics on B&P Algorithm

In the following, we explore which characteristics influence the complexity of an instance for our B&P. Table 4.2 and Tables 4.7–4.11 display the results of our B&P for the benchmarks `pagination` and `general` averaged by number of items n, number of elements m, capacity Q, average element frequency, average item cardinality, and average number of items per bin in the optimal solution, respectively.

Table 4.2 reveals that the number of items n is a major influencing factor on the performance of our B&P. As expected according to the results from Section 4.5.2 for solving the LP relaxation, `pagination` instances with more items are significantly harder to solve. While all instances with $n \leq 50$ are solved optimally in a very short time, only two thirds of the instances can be solved in the most difficult class $n = 100$. The picture is reversed for the `general` instances: For the largest instances with up to 500 items, more instances are solved to optimality in significantly less computation time compared to the instances with the fewest items.

With an increasing number of elements m, both `pagination` and `general` instances seem to become easier as the percentages of optimally solved instances tend to increase while the average computation times tend to decrease (see Table 4.7). For the `pagination` benchmark, however, the effect is not as strong as with the number of items n, so that the influence of n appears to be more significant than that of m. The results for the `general` set are the same as for n, because n and m are within the same size by definition of the instances.

Table 4.8 shows the results aggregated by capacity Q. It appears that instances tend to become more difficult with increasing Q for `pagination`, but the effect is

limited. It should be noted that Q is related to the number of elements m due to the construction of the instances ($m > Q$). With `general` instances, there seems to be no clear trend. Note that here, the capacity is specified in intervals as there are no predefined capacity classes.

The results grouped by average frequency are shown in Table 4.9. More than half of the `pagination` instances have a very small average frequency of up to ten items per element. The frequency does not appear to have a significant influence on the difficulty of a `pagination` instance, while a higher frequency corresponds with easier `general` instances. Interestingly, the specific characteristic of the SUBP, namely the set-union property which is represented by the frequency of an element, therefore, appears to have less influence on the difficulty than, e.g., the size n of the problem or the bin capacity Q.

Table 4.10 summarizes the results by average cardinality of the items, where the *cardinality* of an item $i \in I$ is defined as the number of elements it requires, i.e., $|E_i|$. We can observe that a larger average cardinality seems to strongly correlate with the difficulty of an instance. This holds for both `pagination` and `general` instances.

Table 4.11 displays the results aggregated by the average number of items per bin in the BKS, i.e., $\frac{n}{BKS}$, which can be considerably larger for `pagination` than for `general` instances. With both instance sets, there is a strong tendency that the complexity of an instance increases the more items are grouped into the bins. This observation is in line with similar findings for the related BP.

m	Inst	%Opt	t[s]
Panel A: `pagination` instances			
20	102	86.3	322.8
25	204	80.9	408.7
30	306	87.3	295.9
35	408	86.5	300.9
40	510	85.7	307.4
45	612	86.1	316.8
50	714	87.8	296.5
55	816	87.7	289.4
60	816	90.3	237.3
65	816	91.9	222.2
70	816	93.3	181.2
75	816	94.4	178.3
80	816	96.3	143.5
85	810	95.9	125.6
90	810	96.8	112.2
95	810	97.8	100.5
100	804	97.8	77.7
Total	10,986	92.1	204.8
Panel B: `general` instances			
$\{85, 100\}$	30	70.0	614.9
$\{185, 200\}$	30	60.0	817.3
$\{285, 300\}$	30	76.7	538.7
$\{385, 400\}$	30	80.0	540.3
$\{485, 500\}$	30	90.0	395.0
Total	150	75.3	581.2

Table 4.7: Summary results of our B&P aggregated by number of elements

Q	Inst	%Opt	t[s]
Panel A: `pagination` instances			
15	1,704	92.6	192.5
20	1,632	95.2	133.2
25	1,530	95.5	136.9
30	1,428	91.5	219.8
35	1,326	90.7	233.4
40	1,224	92.0	215.3
45	1,122	89.3	284.4
50	1,020	87.6	283.6
Total	10,986	92.1	204.8
Panel B: `general` instances			
(0, 5,000]	9	100.0	7.2
(5,000, 10,000]	40	80.0	464.8
(10,000, 15,000]	54	64.8	723.0
(15,000, 20,000]	32	77.1	641.5
(20,000, 25,000]	12	83.3	585.9
Total	150	75.3	581.2

Table 4.8: Summary results of our B&P aggregated by capacity

Frequency	Inst	%Opt	t[s]
Panel A: `pagination` instances			
(0, 10]	6,200	96.4	121.9
(10, 20]	3,222	84.7	363.8
(20, 30]	980	94.7	134.8
(30, 40]	347	78.4	452.2
(40, 50]	140	87.9	229.1
(50, 60]	76	100.0	10.0
(60, 70]	21	100.0	13.7
Total	10,986	92.1	204.8
Panel B: `general` instances			
(0, 10]	15	73.3	563.6
(10, 20]	25	64.0	751.0
(20, 30]	20	65.0	774.0
(30, 40]	20	60.0	782.3
(40, 50]	40	87.5	426.3
(50, 60]	10	90.0	348.7
(60, 70]	5	80.0	535.8
(70, 80]	15	86.7	374.0
Total	150	75.3	581.2

Table 4.9: Summary results of our B&P aggregated by average frequency

Cardinality	Inst	%Opt	t[s]
Panel A: `pagination` instances			
(0, 10]	4,947	89.7	276.4
(10, 20]	4,388	92.8	181.9
(20, 30]	1,397	97.4	60.4
(30, 40]	254	100.0	0.9
Total	10,986	92.1	204.8
Panel B: `general` instances			
(0, 10]	15	73.3	563.6
(10, 20]	25	64.0	751.0
(20, 30]	25	64.0	765.8
(30, 40]	20	65.0	752.4
(40, 50]	30	86.7	420.4
(50, 60]	20	90.0	376.9
(60, 70]	0	–	–
(70, 80]	15	86.7	374.0
Total	150	75.3	581.2

Table 4.10: Summary results of our B&P aggregated by average cardinality

n/BKS	Inst	%Opt	t[s]
Panel A: `pagination` instances			
(0, 3]	3,868	100.0	0.1
(3, 6]	3,245	99.3	24.4
(6, 9]	2,386	87.7	371.8
(9, 12]	940	68.7	787.1
(12, 15]	312	57.8	946.2
(15, 18]	84	42.0	1,167.5
(18, 21]	68	55.7	927.2
(21, 24]	55	46.4	1,082.7
(24, 27]	19	37.5	1,198.5
(27, 30]	4	100.0	618.2
(30, 33]	3	66.7	1,170.6
(33, 36]	2	0.0	1,800.0
Total	10,986	92.1	204.8
Panel B: `general` instances			
(0, 3]	104	86.5	380.8
(3, 6]	30	53.3	989.0
(6, 9]	6	83.3	458.0
(9, 12]	6	16.7	1,586.7
(12, 15]	2	50.0	1,021.7
Total	150	75.3	581.2

Table 4.11: Summary results of our B&P aggregated by average number of items per bin

4.H Influence of Additional UBs on B&P Algorithm

In Table 4.12, we compare the results of our B&P algorithm (*Our B&P*) with the results obtained by initializing our B&P with additional UBs. More precisely, we tested the best UB reported by Grange *et al.* (2018) for the `pagination` instances (*B&P + Gr-UB*) as well as the overall BKS (*B&P + BKS*). For the `pagination` benchmark, the effect is limited: only 19 additional instances can be solved and the improvement in average computation time is marginal. Thus, for the majority of instances, additional effort to provide high-quality primal solutions, seems to not pay off for our B&P. Recall, however, that there exist some unsolved instances for which we assume the BKS to be suboptimal. For these hard instances, improved UBs may be crucial for their solution. For the `general` benchmark, on the other hand, a significant improvement is observed when utilizing the BKS as an initial UB. Our B&P is able to solve 21 additional instances in only 60% of the average computation time. Here, improved primal heuristics may be worthwhile.

		Our B&P		B&P + Gr-UB		B&P + BKS	
n	Inst	Opt	t[s]	Opt	t[s]	Opt	t[s]
Panel A: `pagination` instances							
20	624	**624**	**0.0**	**624**	**0.0**	**624**	**0.0**
25	642	**642**	**0.0**	**642**	**0.0**	**642**	**0.0**
30	648	**648**	**0.1**	**648**	**0.1**	**648**	**0.1**
35	648	**648**	**0.2**	**648**	**0.2**	**648**	**0.2**
40	648	**648**	0.6	**648**	**0.5**	**648**	**0.5**
45	648	**648**	2.1	**648**	**2.0**	**648**	**2.0**
50	648	**648**	10.4	647	**10.1**	647	10.2
55	648	**643**	43.0	**643**	**42.5**	**643**	42.7
60	648	**636**	65.1	635	**63.8**	635	64.6
65	648	622	126.6	623	**116.2**	**624**	117.4
70	648	602	208.9	**604**	**201.3**	603	202.2
75	648	582	279.0	583	**269.8**	**585**	272.0
80	648	565	375.9	565	**372.7**	**567**	375.3
85	648	534	451.4	537	**444.4**	**538**	444.5
90	648	510	553.7	**515**	**542.3**	**515**	544.5
95	648	478	632.1	479	**628.7**	**480**	629.6
100	648	445	723.2	**447**	**722.0**	**447**	722.3
Total	10,986	10,123	204.8	10,136	**201.5**	**10,142**	202.2
Panel B: `general` instances							
$\{85, 100\}$	30	21	614.9			**28**	**198.2**
$\{185, 200\}$	30	18	817.3			**24**	**514.0**
$\{285, 300\}$	30	23	538.7			**25**	**443.9**
$\{385, 400\}$	30	24	540.3			**27**	**359.0**
$\{485, 500\}$	30	27	395.0			**30**	**299.3**
Total	150	113	581.2			**134**	**362.9**

Table 4.12: Summary results of our B&P incorporating different UBs

4.I Instances Not Satisfying IRUP

In this section, we present details on two of the smallest instances that we could find that do not satisfy the IRUP. For both instances, we state the optimal solution together with the optimal solution of the LP relaxation of Formulation (4.4) provided by our B&P.

Example Instance 1

This instance is based on the second `pagination` instance with $Q = 30$, $m = 35$, $n = 20$. The instance could be reduced to $n = 17$ items, while still not satisfying the IRUP.

Instance data: E_i for all $i \in I$:
{7,8,15,18,20,23,28,29,32}
{3,5,8,17,19,29}
{1,6,8,12,15,24,26,34}
{3,5,12,14,17,21,22,23,24,25,28,34}
{0,15,16,17,20,24,29}
{1,14,18,21,25,26,30}
{1,5,6,11,13,18,26,29,30,32,33}
{1,2,3,4,9,11,13,14,16,19,25}
{1,2,4,5,7,8,12,13,14,15,20,33}
{0,4,8,11,13,17,19,20,21,23,25,33,34}
{2,6,31}
{5,6,8,15,16,18,20,22,28}
{5,10,13,16,19,29}
{6,9,10,11,18,20,25,28,31}
{0,10,24,26,27,29,34}
{2,4,7,8,14,16,17,18,19,20,24,30,32}
{5,8,10,11,12,15,17,20,21,23,28,32,33,34}

Optimal solution (3 bins):
Bin [3 7 8 9 15 16], elements {0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 28 30 32 33 34} = 1
Bin [0 2 4 5 6 11 13], elements {0 1 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 21 22 23 24 25 26 28 29 30 31 32 33 34} = 1
Bin [1 10 12 14], elements {0 2 3 5 6 8 10 13 16 17 19 24 26 27 29 31 34} = 1

LP relaxation (obj val = 2.0):
Bin [1 5 6 7 8 10 12 13 15], elements {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 24 25 26 28 29 30 31 32 33} = 0.25
Bin [2 3 4 10 11 13 14 16], elements {0 1 2 3 5 6 8 9 10 11 12 14 15 16 17 18 20 21

22 23 24 25 26 27 28 29 31 32 33 34} = 0.25
Bin [0 1 2 3 4 5 11 14 15], elements {0 1 2 3 4 5 6 7 8 10 12 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 32 34} = 0.25
Bin [0 2 5 6 8 9 14 16], elements {0 1 2 4 5 6 7 8 10 11 12 13 14 15 17 18 19 20 21 23 24 25 26 27 28 29 30 32 33 34} = 0.25
Bin [3 7 9 10 11 13 16], elements {0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 28 31 32 33 34} = 0.25
Bin [0 1 6 7 10 11 12 13 15], elements {1 2 3 4 5 6 7 8 9 10 11 13 14 15 16 17 18 19 20 22 23 24 25 26 28 29 30 31 32 33} = 0.25
Bin [0 1 3 4 7 8 9 12 16], elements {0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 28 29 32 33 34} = 0.25
Bin [2 4 5 6 8 9 12 14 15], elements {0 1 2 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 27 29 30 32 33 34} = 0.25

Example Instance 2

This instance is based on the first `pagination` instance with $Q = 15$, $m = 20$, $n = 30$. The instance could be reduced to $n = 24$ items, while still not satisfying the IRUP.

Instance data: E_i for all $i \in I$:
{0,1,2,3,4,7,9,11,16}
{3,4,5,6,7,9,10,14,17,19}
{5,8,11,12,16,19}
{5,8,16,18,19}
{2,5,3,7,8,10,16,17,19}
{1,4,6,9,12,16}
{5,6,7,11,14,16}
{1,3,5,7,10,15}
{0,1,6,8,12,13}
{2,6,8,12,15,17,19}
{2,4,6,7,11,13,17,19}
{2,3,6,9,12,14,16,17}
{2,3,5,7,11,17}
{3,7,9,11,14,17,18}
{1,5,10,12,13,15,16}
{6,7,9,10,12,16,19}
{3,8,12,15,19}
{3,7,8,9,10,12,16}
{0,4,6,8,11,14,19}
{1,4,7,10,11,12,16}
{1,4,5,9,11,13,14}

{6,7,8,10,11,12,15,18}
{0,1,3,4,6,9,10,13,19}
{0,2,5,10,12,13,19}

Optimal solution (5 bins):
Bin [4 8 9 23], elements {0 1 2 3 5 6 7 8 10 12 13 15 16 17 19} = 1
Bin [0 5 11 13 19], elements {0 1 2 3 4 6 7 9 10 11 12 14 16 17 18} = 1
Bin [1 10 12 20 22], elements {0 1 2 3 4 5 6 7 9 10 11 13 14 17 19} = 1
Bin [2 3 6 7 15 16 17 21], elements {1 3 5 6 7 8 9 10 11 12 14 15 16 18 19} = 1
Bin [14 18], elements {0 1 4 5 6 8 10 11 12 13 14 15 16 19} = 1

LP relaxation (obj val = 3.9512198):
Bin [0 5 11 18], elements {0 1 2 3 4 6 7 8 9 11 12 14 16 17 19} = 0.0609756
Bin [1 18 20 22], elements {0 1 3 4 5 6 7 8 9 10 11 13 14 17 19} = 0.0365854
Bin [0 5 10 15 19 22], elements {0 1 2 3 4 6 7 9 10 11 12 13 16 17 19} = 0.536585
Bin [2 3 4 6 11 12 13 15 17], elements {2 3 5 6 7 8 9 10 11 12 14 16 17 18 19} = 0.170732
Bin [8 18 20 22 23], elements {0 1 2 3 4 5 6 8 9 10 11 12 13 14 19} = 0.304878
Bin [4 7 8 9 14 16 23], elements {0 1 2 3 5 6 7 8 10 12 13 15 16 17 19} = 0.426829
Bin [1 2 4 6 11 12 15 17], elements {2 3 4 5 6 7 8 9 10 11 12 14 16 17 19} = 0.317073
Bin [2 3 4 7 9 12 16 21], elements {1 2 3 5 6 7 8 10 11 12 15 16 17 18 19} = 0.0853659
Bin [5 7 14 17 19 21], elements {1 3 4 5 6 7 8 9 10 11 12 13 15 16 18} = 0.0487805
Bin [2 3 5 7 15 16 17 19 21], elements {1 3 4 5 6 7 8 9 10 11 12 15 16 18 19} = 0.158537
Bin [9 11 13 16 17 21], elements {2 3 6 7 8 9 10 11 12 14 15 16 17 18 19} = 0.304878
Bin [2 3 8 14 21 23], elements {0 1 2 5 6 7 8 10 11 12 13 15 16 18 19} = 0.195122
Bin [0 7 14 19 20], elements {0 1 2 3 4 5 7 9 10 11 12 13 14 15 16} = 0.182927
Bin [2 3 7 8 14 16 21], elements {0 1 3 5 6 7 8 10 11 12 13 15 16 18 19} = 0.0243902
Bin [0 5 6 11 12 20], elements {0 1 2 3 4 5 6 7 9 11 12 13 14 16 17} = 0.146341
Bin [1 3 6 13 18], elements {0 3 4 5 6 7 8 9 10 11 14 16 17 18 19} = 0.365854
Bin [9 10 18 21], elements {0 2 4 6 7 8 10 11 12 13 14 15 17 18 19} = 0.182927
Bin [0 7 14 19 23], elements {0 1 2 3 4 5 7 9 10 11 12 13 15 16 19} = 0.0731707
Bin [1 10 12 13 20], elements {1 2 3 4 5 6 7 9 10 11 13 14 17 18 19} = 0.158537
Bin [1 10 12 20 22], elements {0 1 2 3 4 5 6 7 9 10 11 13 14 17 19} = 0.121951
Bin [2 5 8 14 18 20], elements {0 1 4 5 6 8 9 10 11 12 13 14 15 16 19} = 0.0487805

Bibliography

Achterberg, T. (2007). *Constraint Integer Programming.* Ph.D. thesis, Fakultät II – Mathematik und Naturwissenschaften, Technische Universität Berlin, Berlin, Germany.

Arulselvan, A. (2014). A note on the set union knapsack problem. *Discrete Applied Mathematics*, **169**, 214–218.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, **115**, 351–385.

Crama, Y. and Oerlemans, A. G. (1994). A column generation approach to job grouping for flexible manufacturing systems. *European Journal of Operational Research*, **78**(1), 58–80.

Grange, A., Kacem, I., and Martin, S. (2018). Algorithms for the bin packing problem with overlapping items. *Computers & Industrial Engineering*, **115**, 331–341.

He, Y., Xie, H., Wong, T.-L., and Wang, X. (2018). A novel binary artificial bee colony algorithm for the set-union knapsack problem. *Future Generation Computer Systems*, **78**, 77–86.

Heßler, K. and Irnich, S. (2022). Modeling and exact solution of picker routing and order batching problems. Technical Report LM-2022-03, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Tang, C. S. and Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part II: Minimization of the number of switching instants. *Operations Research*, **36**(5), 778–784.

Wahlen, J. and Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, **57**(3), 756–777.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2020). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*, **32**(2), 428–443.

Chapter 5

Conclusion

The main goal of this thesis was to advance the development of exact solution methods for the *order batching problem* (OBP) in warehousing and the *set-union bin packing problem* (SUBP), which arises in various applications such as flexible manufacturing. This chapter provides a summary of the key results and offers concluding remarks on the findings.

In Chapter 2, we introduced a full-fledged *branch-price-and-cut* (BPC) approach to solve the single-block OBP for the routing strategies traversal, return, midpoint, largest gap, combined, and optimal. The foundation of this approach is a *column generation* (CG) framework, wherein the pricing problem is formulated as a *shortest path problem with resource constraints* (SPPRC) on a linear directed multigraph. A central innovation of our method is the development of a specialized *dynamic programming* (DP) labeling algorithm for solving the SPPRC, which incorporates strong completion bounds. These bounds are essential, as the non-separability of the distance function prevents the application of traditional dominance relations. Our DP algorithm is further enhanced to handle the effects of non-robust valid inequalities, such as subset-row cuts and capacity cuts, as well as Ryan-and-Foster branching decisions. The powerful CG component enabled the development of two additional heuristic approaches for the OBP that are based on the exact BPC. In an extensive computational campaign, our exact BPC approach exhibited superior performance for benchmark instances, reducing average computation times to one percent of those required by the state-of-the-art exact method and identifying over three times as many proven optima. Furthermore, the BPC-based heuristics improved the majority of the *best-known solutions* (BKS) across large-scale benchmark datasets.

Chapter 3 broadened the scope of this work by addressing the multi-block OBP, building upon the BPC method introduced in Chapter 2. To this end, we investigated the necessary monotonicity properties of six established or modified multi-block picker routing strategies: optimal, no-reversal, aisle-by-aisle, traversal, combined, and largest gap. Notably, all strategies, with the exception of largest gap, were proven to exhibit monotonicity in a rectangular parallel-aisle

multi-block warehouse environment. Computational experiments on real-world instances demonstrated that the BPC method is more than three orders of magnitude faster than state-of-the-art exact approaches, solving the majority of all considered instances to proven optimality. Furthermore, the BPC-based heuristics delivered significant improvements in solution quality within a short time for the multi-block OBP, outperforming previous methods.

In Chapter 4, we presented a novel *branch-and-price* (B&P) approach for the *set-union bin packing problem* (SUBP). Various strategies for solving the CG pricing problem, which is a *set-union knapsack problem*, were analyzed. We identified the best-performing strategy as a combination of an upfront greedy heuristic and solving an item-based SPPRC with a DP labeling algorithm. This labeling algorithm does not employ dominance relations but instead relies on dedicated completion bounds. Extensive computational experiments were conducted on both unit-weight and general-weight benchmark sets. Our B&P approach identified optimal solutions for 92% of over 11,000 instances that were previously addressed solely with heuristic methods, improving the BKS for more than half of the benchmark set. Additionally, we observed that the vast majority of optimally solved instances satisfy the *integer round-up property* (IRUP), while all solved instances adhere to the modified IRUP.

Altogether, the versatility and remarkable computational performance of the CG-based methods developed in this thesis – demonstrated across different problem types featuring non-linear objective functions and knapsack-type substructures – underscore their potential to effectively address a wide range of combinatorial optimization problems with similar characteristics in future research.

Bibliography

Achterberg, T. (2007). *Constraint Integer Programming.* Ph.D. thesis, Fakultät II – Mathematik und Naturwissenschaften, Technische Universität Berlin, Berlin, Germany.

Adjiashvili, D., Bosio, S., and Zemmer, K. (2015). Minimizing the number of switch instances on a flexible machine in polynomial time. *Operations Research Letters*, **43**(3), 317–322.

Aerts, B., Cornelissens, T., and Sörensen, K. (2021). The joint order batching and picker routing problem: Modelled and solved as a clustered vehicle routing problem. *Computers & Operations Research*, **129**, 105168.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1995). *Network flows: Theory, algorithms, and applications.* Prentice Hall, Hoboken, NJ.

Anily, S., Bramel, J., and Simchi-Levi, D. (1994). Worst-case analysis of heuristics for the bin packing problem with general cost structures. *Operations Research*, **42**(2), 287–298.

Archetti, C., Bouchard, M., and Desaulniers, G. (2011). Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science*, **45**(3), 285–298.

Arulselvan, A. (2014). A note on the set union knapsack problem. *Discrete Applied Mathematics*, **169**, 214–218.

Azadeh, K., de Koster, R., and Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, **53**(4), 917–945.

Bahçeci, U. and Öncan, T. (2022). An evaluation of several combinations of routing and storage location assignment policies for the order batching problem. *International Journal of Production Research*, **60**(19), 5892–5911.

Baldacci, R., Christofides, N., and Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, **115**, 351–385.

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

Baldacci, R., Coniglio, S., Cordeau, J.-F., and Furini, F. (2024). A numerically exact algorithm for the bin-packing problem. *INFORMS Journal on Computing*, **36**(1), 141–162.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., and Vance, P. (1998).

Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, **46**(3), 316–329.

Boysen, N., Briskorn, D., and Emde, S. (2017). Sequencing of picking orders in mobile rack warehouses. *European Journal of Operational Research*, **259**(1), 293–307.

Boysen, N., de Koster, R., and Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, **277**(2), 396–411.

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A.-L., and Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, **285**(2), 497–512.

Bué, M., Cattaruzza, D., Ogier, M., and Semet, F. (2019). A two-phase approach for an integrated order batching and picker routing problem. In M. Dell'Amico, M. Gaudioso, and G. Stecca, editors, *A View of Operations Research Applications in Italy, 2018*, volume 2, pages 3–18. Springer International Publishing, Cham, Switzerland.

Burger, A. P., Jacobs, C., van Vuuren, J. H., and Visagie, S. E. (2015). Scheduling multi-colour print jobs with sequence-dependent setup times. *Journal of Scheduling*, **18**, 131–145.

Burkard, R. E., Deineko, V. G., van Dal, R., van der Veen, J. A., and Woeginger, G. J. (1998). Well-solvable special cases of the traveling salesman problem: A survey. *SIAM Review*, **40**(3), 496–546.

Calmels, D. (2019). The job sequencing and tool switching problem: State-of-the-art literature review, classification, and trends. *International Journal of Production Research*, **57**(15-16), 5005–5025.

Cambazard, H. and Catusse, N. (2018). Fixed-parameter algorithms for rectilinear Steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, **270**(2), 419–429.

Campêlo, M., Campos, V. A., and Corrêa, R. C. (2008). On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics*, **156**(7), 1097–1111.

Cano, J. A., Correa-Espinal, A. A., and Gómez-Montoya, R. A. (2020). Mathematical programming modeling for joint order batching, sequencing and picker routing problems in manual order picking systems. *Journal of King Saud University – Engineering Sciences*, **32**(3), 219–228.

Caron, F., Marchet, G., and Perego, A. (2000). Optimal layout in low-level picker-to-part systems. *International Journal of Production Research*, **38**(1), 101–117.

Çelik, M. and Süral, H. (2014). Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE Transactions*, **46**(3), 283–300.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**(4), 305–337.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, **12**(4), 568–581.

Crama, Y. and Oerlemans, A. G. (1994). A column generation approach to job grouping for flexible manufacturing systems. *European Journal of Operational Research*, **78**(1), 58–80.

Crama, Y., Moonen, L. S., Spieksma, F. C., and Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, **182**(2), 952–957.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, **8**(3), 250–255.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, **8**(1), 101–111.

de Koster, R., van der Poort, E., and Wolters, M. (1999a). Efficient orderbatching methods in warehouses. *International Journal of Production Research*, **37**(7), 1479–1504.

de Koster, R., Roodbergen, K. J., and van Voorden, R. (1999b). Reduction of walking time in the distribution center of De Bijenkorf. In M. Speranza and P. Stähly, editors, *New Trends in Distribution Logistics*, pages 215–234. Springer, Berlin.

de Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, **182**(2), 481–501.

Delorme, M. and Iori, M. (2020). Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. *INFORMS Journal on Computing*, **32**(1), 101–119.

Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, **255**(1), 1–20.

Denizel, M. (2003). Minimization of the number of tool magazine setups on automated machines: A Lagrangean decomposition approach. *Operations Research*, **51**(2), 309–320.

Desrosiers, J., Jans, R., and Adulyasak, Y. (2013). Improved column generation algorithms for the job grouping problem. *Les Cahiers du GERAD*. G-2013-26.

Desrosiers, J., Lübbecke, M., Desaulniers, G., and Gauthier, J. B. (2024). *Branch-and-Price*. Les Cahiers du GERAD, Montréal, Canada.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, **42**(5), 977–978.

Dror, M. and Haouari, M. (2000). Generalized Steiner problems and other variants. *Journal of Combinatorial Optimization*, **4**, 415–436.

Establish Inc. (2013). Establish Davis logistics costs and service 2013. Presentation. CSCMPS Annual Global Conference, Denver.

Frazelle, E. (2001). *World-Class Warehousing and Material Handling*. McGraw-Hill Book Company, New York.

Gademann, N. and van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, **37**(1), 63–75.

Gademann, N., van den Berg, J., and van der Hoff, H. (2001). An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Transactions*, **33**(5), 385–398.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, volume 174. W. H. Freeman and Company, San Francisco.

Ghiani, G., Laporte, G., and Musmanno, R. (2013). Introduction to logistics systems management. In L. V. Snyder, Y. Yuan, and J. Cochran, editors, *Wiley Series in Operations Research and Management Science*. John Wiley & Sons, 2nd edition.

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**(6), 849–859.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 436–451.

Goetschalckx, M. and Ratliff, H. (1988). Order picking in an aisle. *IIE Transactions*, **20**(1), 53–62.

Gokgur, B. and Özpeynirci, S. (2022). Minimization of number of tool switching instants in automated manufacturing systems. *Journal of Science*, **35**(1), 113–130.

Goldschmidt, O., Nehme, D., and Yu, G. (1994). Note: On the set-union knapsack problem. *Naval Research Logistics*, **41**(6), 833–842.

Goldschmidt, O., Hochbaum, D. S., Hurkens, C., and Yu, G. (1996). Approximation algorithms for the k-clique covering problem. *SIAM Journal on Discrete Mathematics*, **9**(3), 492–509.

Gomory, R. (1963). An algorithm for integer solutions to linear programs. In R. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill Book Company.

Grange, A., Kacem, I., and Martin, S. (2018). Algorithms for the bin packing problem with overlapping items. *Computers & Industrial Engineering*, **115**, 331–341.

Grange, A., Kacem, I., Martin, S., and Minich, S. (2023). Fully polynomial time approximation scheme for the pagination problem with hierarchical structure of tiles. *RAIRO – Operations Research*, **57**(1), 1–16.

Grosse, E. H., Glock, C. H., and Ballester-Ripoll, R. (2014). A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. *International Journal of Operations and Quantitative Management*, **20**(2), 65–83.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Gschwind, T., Irnich, S., Furini, F., and Calvo, R. W. (2021). A branch-and-price framework for decomposing graphs into relaxed cliques. *INFORMS Journal on Computing*, **33**(3), 1070–1090.

Gu, J., Goetschalckx, M., and McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, **177**(1), 1–21.

Gu, J., Goetschalckx, M., and McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, **203**(3), 539–549.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, **25**(4), 76–87.

He, Y., Xie, H., Wong, T.-L., and Wang, X. (2018). A novel binary artificial bee colony algorithm for the set-union knapsack problem. *Future Generation Computer Systems*, **78**, 77–86.

Henn, S. and Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*, **222**(3), 484–494.

Henn, S., Koch, S., and Wäscher, G. (2012). Order batching in order picking warehouses: A survey of solution approaches. In R. Manzini, editor, *Warehousing in the Global Supply Chain*, pages 105–137. Springer, London.

Heßler, K. and Irnich, S. (2021). A branch-and-cut algorithm for the soft-clustered vehicle-routing problem. *Discrete Applied Mathematics*, **288**, 218–234.

Heßler, K. and Irnich, S. (2022a). Modeling and exact solution of picker routing and order batching problems. Technical Report LM-2022-03, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Heßler, K. and Irnich, S. (2022b). A note on the linearity of Ratliff and Rosenthal's algorithm for optimal picker routing. *Operations Research Letters*, **50**(2), 155–159.

Heßler, K. and Irnich, S. (2024). Exact solution of the single-picker routing problem with scattered storage. *INFORMS Journal on Computing*. Advance online publication.

Heßler, K., Gschwind, T., and Irnich, S. (2018). Stabilized branch-and-price algorithms for vector packing problems. *European Journal of Operational Research*, **271**(2), 401–419.

Hintsch, T. and Irnich, S. (2020). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, **280**(1), 164–178.

Hirabayashi, R., Suzuki, H., and Tsuchiya, N. (1984). Optimal tool module design problem for NC machine tools. *Journal of the Operations Research Society of Japan*, **27**(3), 205–229.

Hong, S. and Kim, Y. (2017). A route-selecting order batching model with the s-shape routes in a parallel-aisle order picking system. *European Journal of Operational Research*, **257**(1), 185–196.

Hong, S., Johnson, A. L., and Peters, B. A. (2012). Large-scale order batching in parallel-aisle picking systems. *IIE Transactions*, **44**(2), 88–106.

Hu, Q., Wei, L., and Lim, A. (2018). The two-dimensional vector packing problem with general costs. *Omega*, **74**, 59–69.

Hwang, H. and Kim, D. G. (2005). Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system. *International Journal of Production Research*, **43**(17), 3657–3670.

Hwang, H., Oh, Y. H., and Lee, Y. K. (2004). An evaluation of routing policies for order-picking operations in low-level picker-to-part system. *International Journal of Production Research*, **42**(18), 3873–3889.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer Science & Business Media, Boston.

Irnich, S., Toth, P., and Vigo, D. (2014). The family of vehicle routing problems. In P. Toth and D. Vigo, editors, *Vehicle Routing: Problems, Methods, and Applications*, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition.

Izumi, T., Yokomaru, T., Takahashi, A., and Kajitani, Y. (1998). Computational complexity analysis of set-bin-packing problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **81**(5), 842–849.

Jans, R. and Desrosiers, J. (2010). Binary clustering problems: Symmetric, asymmetric and decomposition formulations. *Les Cahiers du GERAD*. G-2010-44.

Jans, R. and Desrosiers, J. (2013). Efficient symmetry breaking formulations for the job grouping problem. *Computers & Operations Research*, **40**(4), 1132–1142.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Junglas, D. (2007). *Optimised grid-partitioning for block structured grids in parallel computing*. Ph.D. thesis, Fachbereich Mathematik, Technische Universität Darmstadt, Darmstadt, Germany.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin, Heidelberg.

Kochetov, Y. and Kondakov, A. (2017). VNS matheuristic for a bin packing problem with a color constraint. *Electronic Notes in Discrete Mathematics*, **58**, 39–46.

Konak, A. and Kulturel-Konak, S. (2007). An ant colony optimization approach to the minimum tool switching instant problem in flexible manufacturing system. In *2007 IEEE Symposium on Computational Intelligence in Scheduling*, pages 43–48, Honolulu, HI. IEEE.

Konak, A., Kulturel-Konak, S., and Azizoğlu, M. (2008). Minimizing the number of tool switching instants in flexible manufacturing systems. *International Journal of Production Economics*, **116**(2), 298–307.

Korte, B. and Vygen, J. (2018). *Combinatorial Optimization: Theory and Algorithms*. Springer, Berlin, Heidelberg, 6th edition.

Kulak, O., Sahin, Y., and Taner, M. E. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, **24**, 52–80.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Locatelli, A. (2023). Optimization methods for knapsack and tool switching problems. *4OR*, **21**(4), 715–716.

Lüke, L., Heßler, K., and Irnich, S. (2024). The single picker routing problem with scattered storage: modeling and evaluation of routing and storage policies. *OR Spectrum*, **46**, 909–951.

Lysgaard, J., Letchford, A. N., and Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical programming*, **100**, 423–445.

Marchet, G., Melacini, M., and Perotti, S. (2015). Investigating order picking system adoption: A case-study-based approach. *International Journal of Logistics Research and Applications*, **18**(1), 82–98.

Martello, S. and Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics*, **28**(1), 59–70.

Martinelli, R., Poggi, M., and Subramanian, A. (2013). Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research*, **40**(8), 2145–2160.

Marvizadeh, S. Z. and Choobineh, F. (2013). Reducing the number of setups for CNC punch presses. *Omega*, **41**(2), 226–235.

Menéndez, B., Bustillo, M., Pardo, E. G., and Duarte, A. (2017). General variable neighborhood search for the order batching and sequencing problem. *European Journal of Operational Research*, **263**(1), 82–93.

Michel, R. (2016). 2016 Warehouse/DC Operations Survey: Ready to confront complexity. Supply Chain Management Review. `https://www.scmr.com/article/2016_warehouse_dc_operations_survey_ready_to_confront_complexity`, November 8.

Muter, İ. and Öncan, T. (2015). An exact solution approach for the order batching problem. *IIE Transactions*, **47**(7), 728–738.

Nemhauser, G. L. and Wolsey, L. A. (2014). *Integer and Combinatorial Optimization*. John Wiley & Sons, Hoboken, NJ.

Öncan, T. (2015). MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem. *European Journal of Operational Research*, **243**(1), 142–155.

Padberg, M. and Rinaldi, G. (1988). Branch-and-cut approach to a variant of the

traveling salesman problem. *Journal of Guidance, Control, and Dynamics*, **11**(5), 436–440.

Pansart, L., Catusse, N., and Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, **100**, 117–127.

Pardo, E. G., Gil-Borrás, S., Alonso-Ayuso, A., and Duarte, A. (2024). Order batching problems: Taxonomy and literature review. *European Journal of Operational Research*, **313**(1), 1–24.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. *Decision Sciences Institute Proceedings*, **3**, 1614–1616.

Petersen, C. G. (1997). An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, **17**(11), 1098–1111.

Petersen, C. G. and Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, **92**(1), 11–19.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, **31**(3), 507–521.

Richards, G. (2017). *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. Kogan Page, London, 3rd edition.

Roodbergen, K. J. (2001). *Layout and routing methods for warehouses*. Ph.D. thesis, Erasmus University Rotterdam, Rotterdam, the Netherlands.

Roodbergen, K. J. and de Koster, R. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, **39**(9), 1865–1883.

Roodbergen, K. J. and de Koster, R. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, **133**(1), 32–43.

Ryan, D. M. and Foster, B. A. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport*, pages 269–280. North-Holland Publishing Company, Amsterdam.

Schiffer, M., Boysen, N., Klein, P. S., Laporte, G., and Pavone, M. (2022). Optimal picking policies in e-commerce warehouses. *Management Science*, **68**(10), 7497–7517.

Scholz, A. and Wäscher, G. (2017). Order batching and picker routing in manual order picking systems: The benefits of integrated routing. *Central European Journal of Operations Research*, **25**(2), 491–520.

Shirazi, R. and Frizelle, G. (2001). Minimizing the number of tool switches on a flexible machine: An empirical study. *International Journal of Production Research*, **39**(15), 3547–3560.

Sindelar, M., Sitaraman, R. K., and Shenoy, P. (2011). Sharing-aware algorithms for virtual machine colocation. In *Proceedings of the twenty-third annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 367–378, San Jose, CA. Association for Computing Machinery.

Tang, C. S. and Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part II: Minimization of the number of switching instants. *Operations Research*, **36**(5), 778–784.

Thia, F. (2008). MySQL Foodmart Database. Pentaho Wiki. `http://pentaho.dlpage.phi-integration.com/mondrian/mysql-foodmart-database`, May 8.

Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons, Hoboken, NJ, 4th edition.

Valle, C. A. and Beasley, J. E. (2020). Order batching using an approximation for the distance travelled by pickers. *European Journal of Operational Research*, **284**(2), 460–484.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2016). Modelling and solving the joint order batching and picker routing problem in inventories. In R. Cerulli, S. Fujishige, and A. R. Mahjoub, editors, *Combinatorial Optimization*, volume 9849 of *Lecture Notes in Computer Science*, pages 81–97, Cham, Switzerland. Springer International Publishing.

Valle, C. A., Beasley, J. E., and da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, **262**(3), 817–834.

van Gils, T., Caris, A., Ramaekers, K., and Braekers, K. (2019). Formulating and solving the integrated batching, routing, and picker scheduling problem in a real-life spare parts warehouse. *European Journal of Operational Research*, **277**(3), 814–830.

Vaughan, T. (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, **37**(4), 881–897.

Wahlen, J. and Gschwind, T. (2023). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*, **57**(3), 756–777.

Wäscher, G. (2004). Order picking: A survey of planning problems and methods. In H. Dyckhoff, R. Lackes, and J. Reese, editors, *Supply Chain Management and Reverse Logistics*, pages 323–347. Springer, Berlin, Heidelberg.

Wei, L., Luo, Z., Baldacci, R., and Lim, A. (2020). A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing*, **32**(2), 428–443.

Wei, Z. (2021). *Optimization algorithms for two knapsack problems*. Ph.D. thesis, Optimization and Control [math.OC], Université d'Angers, Angers, France.

Won, J. and Olafsson, S. (2005). Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, **43**(7), 1427–1442.

Zhang, K. and Gao, C. (2023). Improved formulations of the joint order batching and picker routing problem. *International Journal of Production Research*, **61**(21), 7386–7409.

Žulj, I., Kramer, S., and Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu search for the order-batching problem. *European Journal of Operational Research*, **264**(2), 653–664.

Academic Curriculum Vitae

Julia Wahlen

2021 – 2025	**Research Associate and PhD Student** Chair of Logistics, Prof. Dr. Timo Gschwind School of Business and Economics University of Kaiserslautern-Landau
2017 – 2019	**Master of Science** Management, specialized in Information and Logistics Johannes Gutenberg University Mainz Semester abroad at SGH Warsaw School of Economics
2013 – 2016	**Bachelor of Science** Mathematics in Business and Economics University of Mannheim