Contributions to Exact Algorithms for Picker Routing and Packing Problems

Dissertation
zur Erlangung des Grades eines Doktors der
wirtschaftlichen Staatswissenschaften
(Dr. rer. pol.)
des Fachbereichs Rechts- und Wirtschaftswissenschaften
der Johannes Gutenberg-Universität Mainz

vorgelegt von
M.Sc. Laura Lüke (geb. Korbacher)
in Mainz

im Jahre 2026

Tag der mündlichen Prüfung: 02.03.2026

Contents

List of Papers

- Laura Lüke[1], Katrin Heßler[2], Stefan Irnich[1] (2024). The Single Picker Routing Problem with Scattered Storage: Modeling and Evaluation of Routing and Storage Policies. *OR Spectrum* **46**(3), 909–951.
- Laura Lüke[1], André Hessenius[1], Stefan Irnich[1] (2025). A Linear-Size Model for the Single Picker Routing Problem with Scattered Storage. *European Journal of Operational Research* **332**(1), 132–143.
- Stefan Irnich[1], Laura Lüke[1] (2025). The Single Picker Routing Problem in Parallel-Aisle Warehouses with Multiple Blocks. Technical Report LM-2025-04, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany. Submitted to *Networks*.
- Laura Lüke[1] (2025). Exact Solution of Picker Routing Problems in Zoned Warehouses with Scattered Storage. Technical Report LM-2025-02, Chair of Logistics Management, Johannes Gutenberg University Mainz, Mainz, Germany. Submitted to *Networks*.
- Laura Korbacher[1], Stefan Irnich[1], John Martinovic[3], Nico Strasdat[3] (2023). Solving the Skiving Stock Problem by a Combination of Stabilized Column Generation and the Reflect Arc-Flow Model. *Discrete Applied Mathematics* **334**, 145–162.

[1]Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, 55128 Mainz, Germany
[2]Global Data & AI, Schenker AG, Kruppstr. 4, 45128 Essen, Germany
[3]Institute of Numerical Mathematics, Technische Universität Dresden, Zellescher Weg 12-14, 01069 Dresden, Germany

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Combinatorial optimization seeks to find an optimal solution from a finite set of possible solutions (Schrijver, 2003). It plays a central role in discrete decision-making problems, where the goal is to minimize costs or maximize profits while satisfying a set of predefined constraints (Korte and Vygen, 2018). The range of problem classes in combinatorial optimization in logistics is very broad. In addition to routing and packing, it includes assignment, cutting, location, network design, scheduling, and more.

In this thesis, we focus on routing problems arising in warehouse operations, as well as on a classical packing problem. The *single picker routing problem* (SPRP) is a well-established problem in the domain of manual (non-automated) warehouse operations. It is a special case of the *(Steiner) traveling salesman problem* (TSP, Cornuéjols *et al.* (1985); Roodbergen (2001)). The classical version of the SPRP is defined on a rectangular, single-block warehouse with parallel aisles and a single depot. Given a warehouse layout and a list of articles, each associated with a specific storage location, the SPRP seeks to determine a minimum-length tour for a single picker to collect all required articles, starting from and ending at the depot. We consider an extension of the problem known as the *single picker routing problem with scattered storage* (SPRP-SS), where individual articles may be stored at multiple storage locations (Weidinger, 2018). This setting introduces additional decision complexity, as it requires choosing both the order in which to visit storage locations and the specific location from which to retrieve each article. We address the SPRP-SS in both single-block and multi-block warehouses. In addition, we introduce the *multi-zone picker routing problem* (MZPRP) and the *balanced multi-zone picker routing problem* (BMZPRP). In both cases, the warehouse is partitioned into multiple disjoint zones, each containing a depot and assigned a picker with a predefined capacity that limits the number of articles picked in that zone. The objective in both problems is to determine a single picker tour per zone with each tour restricted to its designated zone, so that all required articles are collectively retrieved. The MZPRP aims to minimize the total travel distance of all picker tours. In contrast, the BMZPRP seeks to achieve a balanced

workload distribution by minimizing the maximum tour length among all pickers.

The packing problem addressed in this thesis is the *skiving stock problem* (SSP), which serves as a natural counterpart of the well-known *cutting stock problem*. However, the two are not dual formulations from a mathematical optimization perspective (Martinovic *et al.*, 2020). Given a set of item types, each characterized by its size and frequency of occurrence, the SSP seeks to combine these items into the maximum possible number of larger units, each satisfying a predefined minimum length requirement (Assmann *et al.*, 1984).

1.1 Exact Solution Methods

The set of possible solutions of a given combinatorial optimization problem is typically extensive and, in many cases, increases exponentially with the size of the problem. Consequently, exhaustively evaluating every potential solution is generally inefficient (Schrijver, 2003). The aforementioned problems, along with numerous other combinatorial optimization problems, are classified as NP-hard (Schrijver, 2003), implying that the existence of polynomial-time algorithms for their exact resolution is considered unlikely (Garey and Johnson, 1979). Therefore, advanced algorithmic methods are required to solve problem instances of practical relevance within a reasonable computational time frame. This thesis employs various exact solution approaches to address the problems considered. The fundamental concepts underlying these algorithms are outlined below.

Dynamic programming (DP) is an algorithmic technique used to solve problems with overlapping subproblems. It improves computational efficiency by solving each subproblem only once and storing the results. This prevents redundant calculations and ensures that shared subcomponents are not recomputed multiple times (Cormen *et al.*, 2009).

Branch-and-cut (B&C) integrates the two key components *branch-and-bound* (B&B) and *cutting planes*. B&B is a tree-based search method that systematically explores the set of possible solutions. While branching strategies ensure integer feasible solutions, bounds are used to eliminate subsets of possible solutions that cannot contain an optimal solution. Cutting planes are valid inequalities, also known as cuts, that are added to the problem. These cuts either exclude fractional solutions to strengthen the linear relaxation or represent constraints that were not included in the initial relaxation but are necessary to maintain feasibility (Wolsey, 1998).

Column generation (CG) is a mathematical optimization technique used to solve linear programming relaxations of large-scale problems, particularly those with an exponential number of variables. Instead of solving the full problem with all variables (columns) explicitly included, column generation begins with a *restricted*

master problem (RMP) that includes only a subset of variables. At each iteration, a pricing problem is solved to identify new columns with negative reduced costs that, if added, are likely to improve the objective value. These columns are then added to the RMP, which is re-optimized. The process continues iteratively until no improving columns exist, i.e., when all reduced costs are non-negative, indicating optimality with respect to the linear programming relaxation (Desrosiers *et al.*, 2024).

1.2 Contributions and Outline

The goal of this thesis is to contribute to the development and analysis of new exact solution approaches for various picker routing problems and the SSP. The majority of the thesis is devoted to the picker routing problems described above. All presented solution methods to these problems are based on the seminal work of Ratliff and Rosenthal (1983), who developed a DP for the SPRP, and on Heßler and Irnich (2024), who extended the state space of Ratliff and Rosenthal's DP to solve the SPRP-SS. More precisely, Heßler and Irnich (2024) use the extended state space for an incomplete dynamic programming formulation, which is reformulated as an integer program with additional covering constraints and solved using a *mixed-integer programming* (MIP) solver.

In Chapter 2, we present the first exact solution method for the SPRP-SS considering five heuristic routing policies, and prove that the problem remains NP-hard even if heuristic routing policies are applied. Additionally, we investigate the interplay between routing and storage policies, showing that a suitable combination of routing and storage policy can lead to significant cost reductions, outperforming classical non-scattered storage systems.

Chapter 3 introduces two novel MIP formulations for the SPRP-SS that are linear in both the number of aisles and the number of storage locations. In contrast to the formulation proposed by Heßler and Irnich (2024), which involves a quadratic number of variables due to the structure of the extended state space, our formulations rely on two alternative modifications of the state space that preserve linearity. Specifically, we replace the quadratically growing number of parallel edges with linear-size subnetworks, resulting in distinct state spaces and, consequently, different MIP formulations. We further analyze these formulations with respect to their theoretical properties, including model size and the strength of their linear programming relaxations.

For the SPRP-SS in multi-block warehouse layouts, we propose the first problem-specific exact algorithm in Chapter 4. The provided novel formulation is solved using a B&C algorithm. To this end, we relax the original state space by omitting the connectivity information of the states. Accordingly, several states of the

original state space are aggregated into a single state in the relaxed state space. While on the upside, the number of states per stage grows moderately, on the downside, the tour graph is no longer guaranteed to be connected and may consist of disconnected tour fragments. To prevent this, subtour-elimination constraints are added dynamically to the MIP formulation to ensure that the resulting picker tour is connected.

In Chapter 5, we introduce two new picker routing problems, MZPRP and BMZPRP, for zoned warehouses with scattered storage. To solve these problems, we introduce an efficient algorithm based on a network-flow formulation with various types of covering constraints, which is solved using an MIP solver. Through computational analysis, we demonstrate that zoning can yield significant cost savings in larger warehouse layouts – achieving reductions of up to 12.6% compared to conventional single-zone routing approaches.

In Chapter 6, we adapt the reflect+ algorithm for the SSP, originally developed for the cutting stock problem by Delorme and Iori (2020). Reflect+ is a multi-level approach built upon a specialized flow-based formulation known as the reflect arc-flow model. A key insight underlying this approach is that high-quality solutions can typically be obtained on relatively sparse arc-flow graphs. These graphs are constructed using arc sets derived from a stabilized CG approach, which generates suitable patterns. As a result, most instances can be solved to optimality at early levels of reflect+ using lower- and upper-bounding techniques. Only in rare cases is it necessary to solve the complete integer reflect arc-flow model at the final level of the algorithm.

Chapter 7 summarizes the key findings and presents the conclusions of the thesis.

Chapter 2

The Single Picker Routing Problem with Scattered Storage: Modeling and Evaluation of Routing and Storage Policies

Laura Lüke, Katrin Heßler, Stefan Irnich

Abstract

Despite ongoing automation efforts, most warehouses are still manually operated using a person-to-parts collection strategy. This process of collecting items of customer orders from different storage locations accounts for the majority of the operating costs of the warehouse. Hence, optimizing picker routes is an important instrument to reduce labor costs. We examine the scattered-storage variant of the single picker routing problem in a one-block parallel-aisle warehouse. With scattered storage, an article can be stored at several storage locations within the warehouse, whereas with classic storage, each article has a unique storage location. We use our recently published network-flow model with covering constraints that is based on an extension of the state space of the dynamic-programming formulation by Ratliff and Rosenthal. With modifications in the state graph, this model serves for both exact and all established heuristic routing methods for picker routing. The latter include traversal, return, largest gap, midpoint, and composite. We show that these routing policies can also be implemented through adaptations in the state space. Extensive computational studies highlight a comparison of the different routing and storage policies (in particular class-based storage policies) in the scattered storage context. Analyses demonstrate which combinations of policies are advantageous for the given warehouse layout. For class-based storage policies, we emphasize how the scattering of articles of different classes should be performed: scattering of C-articles is advantageous with reductions of up to 25%. In contrast, when articles are uniformly distributed, A-articles should be scattered.

2.1 Introduction

In this work, we consider the *single picker routing problem* (SPRP), which is one of the fundamental problems in warehouse operations for manual (non-automated) warehouses. For a given warehouse layout and a list of articles with a storage location (=pick position) for each of the articles, the SPRP consists of deciding how a picker travels through the warehouse in order to collect the articles (picker-to-parts) in a minimum-length tour. It is estimated that more than 80% of all order-picking systems in Western Europe are low-level picker-to-parts picking systems (de Koster *et al.*, 2007). A study by Behnisch *et al.* (2017) highlights that 60% of companies in Germany that have small-sized warehouses (<10,000 square meters) use paper-based pick lists so that the pickers are routed by heuristics.

We study the extension of the SPRP to warehouses *with scattered storage* (abbreviated as SPRP-SS in the following) under the combination of different *routing policies* and *storage policies*. The three emphasized terms need to be further elaborated: First, a warehouse operates as a scattered storage warehouse or mixed-shelves warehouse if one or several articles are pickable from more than one pick position. Scattered storage is predominant in modern e-commerce warehouses of companies like Amazon or Zalando (Weidinger, 2018; Weidinger and Boysen, 2018; Boysen *et al.*, 2019; Weidinger *et al.*, 2019; Goeke and Schneider, 2021; Bock and Boysen, 2023; Khan *et al.*, 2024). The main advantage of a scattered storage strategy is "that items of demanded SKUs are found close by irrespective of the position within the warehouse [so that] the distance to be covered for order picking is reduced this way" (Weidinger, 2018, p. 139). Boysen *et al.* (2019, p. 399) point out that warehouses with scattered storage often have multiple I/O points where completed orders are handed over to the central conveyor system. As a result, the length of picker tours further decreases and tight delivery schedules can better be met. Another advantage is that scattered storage comes without excessive automation. In particular an adaption to varying workloads is easily possible by using less or more pickers. Our own study (see Section 2.6.3) reveals that it is of high importance to determine which article should be placed where and how often at different positions in the warehouse. We show that different strategies can easily lead to picker tours that, on average, differ in length by more than 30%.

The term scattered storage should not be confused with *shared storage*: while shared storage allows to allocate a storage location consecutively to different articles in general (Cormier and Gunn, 1992), scattered storage even allows a single load unit to be broken up and distributed to different storage locations (Weidinger and Boysen, 2018).

Second, even for the most simple warehouse layout, i.e., a single-block warehouse with parallel aisles, exact (=minimum length) picker tours can be complicated, counter-intuitive, and difficult to memorize, see, e.g., (Petersen, 1997, p. 1102).

Therefore, the application of routing heuristics can be justified: the pickers can only perform tours defined by some simple rules. We consider the standard rule-based routing policies such as `traversal` (a.k.a. S-shape), `midpoint`, `largest gap` (Hall, 1993), `return`, and `composite` (Petersen, 1997).

Third, articles are either stored at a dedicated pick position, chaotically at random storage locations, or the warehouse is grouped into zones (de Koster *et al.*, 2007). In the latter case, the articles are often assigned to these zones on the basis of their turnover rate (Petersen and Schmenner, 1999). Different *class-based storage policies* can use, e.g., an ABC-classification of articles to determine which article is allocated to which storage locations in the warehouse.

2.1.1 Definition of the SPRP-SS

For a formal definition of the SPRP-SS, we assume that S denotes the set of different articles that a picker must collect in a single picker tour. Moreover, $q_s \in \mathbb{N}$ units (=items) of this article $s \in S$ need to be collected from one or several given positions P_s, where $b_{sp} \in \mathbb{N}$ units of article s are available at position $p \in P_s$. The special case of $q_s = 1$ for all articles $s \in S$ is known as the *unit-demand* case, for which also $b_{sp} = 1$ can be assumed w.l.o.g. The unit-demand case also describes the situation that sufficiently many items of all articles are available so that any demand can be collected from a single position for each article s. Otherwise, if $q_s > 1$ for some $s \in S$, the SPRP-SS instance describes the *general-demand* case. Here, it may be necessary or advantageous to collect the same article from different positions. In the standard variants of the SPRP-SS, pickers start and end their tour at the same point in the warehouse often denoted as the I/O point 0 (=depot).

Let $P_0 = \{0\}$ and $P = P_0 \cup \bigcup_{s \in S} P_s$, where the latter set comprises the set of all positions relevant for the SPRP-SS. A tour is feasible for the given SPRP-SS instance, i.e., it allows the collection of the entire demand, if the tour visits a subset $P' \subseteq P$ such that $0 \in P$ and $\sum_{p \in P' \cap P_s} b_{sp} \geq q_s$ for all $s \in S$. Figure 2.1 shows a feasible tour for an SPRP-SS instance with general demand. Let $D = (d_{pq})$ be the (symmetric) distance between two positions $p \in P$ and $q \in P$. A tour is optimal, if it is feasible and minimizes the distance traveled by the picker using the distance matrix D. In the unit-demand case, the SPRP-SS instance can be modeled and solved as a special case of the *generalized traveling salesman problem* (GTSP, Gutin and Punnen, 2002): The set of cities is given by P, the distances by $D = (d_{pq})$ and the clusters by P_0 and P_s for $s \in S$.

The standard SPRP (non-scattered) is the special case that all articles are available at a unique position, i.e., $|P_s| = 1$ for all $s \in S$. In contrast to the SPRP-SS, there is no selection of positions required, but every tour that covers all given positions is feasible. The SPRP seeks a minimum-length picker tour given the distances and the pick positions from where articles must be collected. Hence, it is

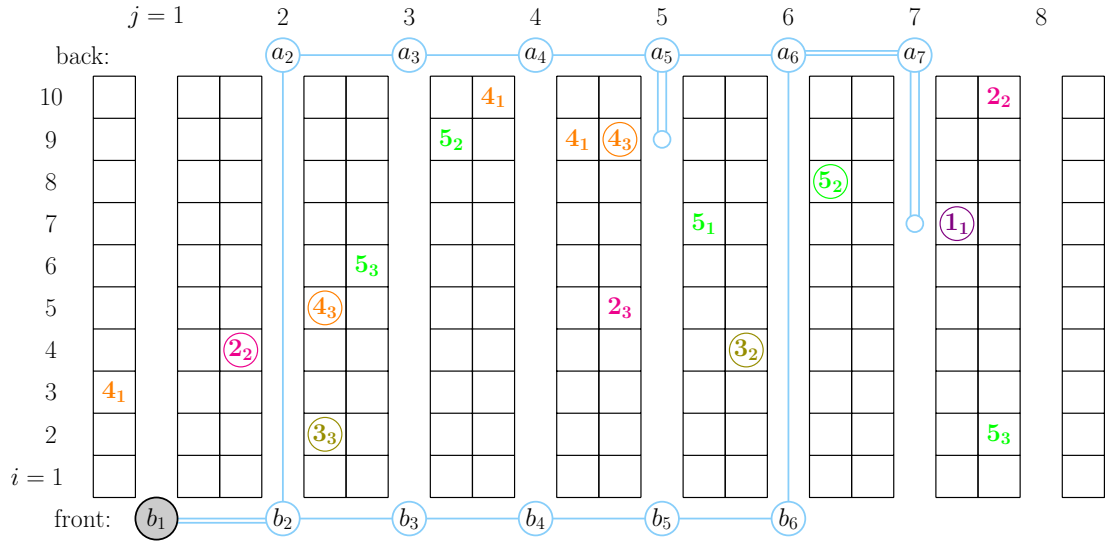


Figure 2.1: A feasible tour for a general-demand SPRP-SS instance with demand $q_1 = q_2 = q_5 = 1$ and $q_3 = q_4 = 5$. The pick positions of the five articles $S = \{1, 2, 3, 4, 5\}$ are indicated with the article s and an index. The index shows the available number b_{sp} of items of the article s at the pick position p. Pick positions from where the picker collects items are encircled.

a special case of the *(Steiner) traveling salesman problem* (TSP, Cornuéjols *et al.*, 1985; Roodbergen, 2001). While the TSP is generally NP-hard, the modeling and solution approaches for picker routing heavily rely on the fact that typical warehouses have a well-defined structure that can be exploited. We discuss this in the following literature review.

2.1.2 Literature Review

Literature on warehouse operations is extensive. For the sake of brevity, we restrict ourselves to the same three central topics already discussed in the introduction, i.e., picker routing, scattered storage, and class-based storage policies. Intentionally, we do not discuss integrated problems in warehouse operations management where picker routing constitutes a well-defined subproblem such as in order batching and batch scheduling (van Gils *et al.*, 2019) or warehouse layout and storage design problems (Henn *et al.*, 2013).

For a single-block parallel-aisle warehouse, the seminal work of Ratliff and Rosenthal (1983) has shown that the SPRP is a well-solvable special case of the TSP. Recently, Heßler and Irnich (2022b) stated the complexity of the SPRP

more precisely as linear in the number of aisles and the number of pick positions. The generalization of this dynamic-programming (DP) algorithm to the case of a two-block parallel-aisle warehouse was presented by Roodbergen and de Koster (2001b). For an arbitrary number of blocks, Pansart *et al.* (2018) presented a DP as well as a MIP-based approach. Despite the low computational complexity of computing optimal picker tours with DP algorithms, the application of heuristic policies is well justified in settings where pickers prefer to perform tours defined by some simple rules. Accordingly, *heuristic routing policies* are rule-based heuristics (see Section 2.2). In the following, we also consider the use of minimum-length tours and denote the routing policy by `exact`. Both exact and heuristic techniques have been extended into many different directions, e.g., to various warehouse layouts (Roodbergen and de Koster, 2001b,a; Öztürkoğlu *et al.*, 2012; Çelk and Süral, 2014), non-identical start and endpoints (Masae *et al.*, 2020a; Löffler *et al.*, 2022), and multiple end depots (de Koster and van der Poort, 1998; Goeke and Schneider, 2021). The most recent survey on picker routing is the one of Masae *et al.* (2020b).

In Table 2.1, we provide an overview of SPRP and SPRP-SS variants that have been discussed in the literature. Variants address different problem dimensions such as the layout of the warehouse, the number of I/O points (=depots), and the routing policies.

Dimension	Attribute	References	DP known
Layout	single-block	Ratliff and Rosenthal (1983)	yes
	two-block	Roodbergen and de Koster (2001b)	yes
	multi-block	Pansart *et al.* (2018)	yes
	flying-V	Gue and Meller (2009)	yes
	butterfly	Öztürkoğlu *et al.* (2012)	no
	fishbone	Çelk and Süral (2014)	yes
	chevron	Masae *et al.* (2019)	yes
	discrete cross-aisles	Öztürkoğlu and Hoser (2019)	yes
	leaf	Masae *et al.* (2021)	yes
I/O point(s)	one unique point (=depot)	Ratliff and Rosenthal (1983)	yes
	multiple drop-off points	de Koster and van der Poort (1998)	yes
	non-identical start and drop-off point	Masae *et al.* (2020a); Löffler *et al.* (2022)	yes
Routing policy	exact	Ratliff and Rosenthal (1983)	yes
	`traversal`, `midpoint`, `largest gap`	Hall (1993)	yes
	`return`,	Petersen (1997)	yes
	`composite`	Petersen (1997)	no/yes[¶]
	`combined`	Roodbergen and de Koster (2001a)	yes[‡]
Demand[†]	unit demand	Daniels *et al.* (1998)	n.a.
	general demand	Daniels *et al.* (1998)	n.a.

[¶]: Different definitions of the policy were given by Petersen (1997), Scholz and Wäscher (2017), and *this paper*, DP known only for the last definition.
[‡]: Different type of DP. [†]: Only relevant for the SPRP-SS.

Table 2.1: Problem dimensions of the SPRP and SPRP-SS. (Similar to Table 1 in (Heßler and Irnich, 2024))

Scattered storage was first addressed by Daniels *et al.* (1998), who also pointed out that several items of the same article can or must be retrieved from differ-

ent pick positions whenever the demanded number of items is greater than one (*general-demand case*). For the general-demand case, the authors proposed a TSP-type of formulation with demand fulfillment constraints as well as a solution approach using a tabu search metaheuristic for the selection of pick positions. For the *unit-demand* case, Daniels *et al.* note that the resulting SPRP-SS is a special case of *generalized traveling salesman problem* (GTSP, Gutin and Punnen, 2002). Although Gu *et al.* (2007) already stated the great research potential of the SPRP-SS, it received only little attention up to the middle of the last decade. Weidinger (2018) coined the term *mixed-shelves storage* as a synonym for scattered storage, emphasizing that it is particularly advantageous in large warehouses with many different articles but rather small, time-critical orders. Weidinger showed that, for a single-block parallel-aisle warehouse, the determination of a minimum-length picker tour is NP-hard. Moreover, he compared a decomposition procedure (select the pick position by different priority rules and use the algorithm of Ratliff and Rosenthal to determine a picker tour) with the MIP-approach of Daniels *et al.* (1998) complemented with MTZ-based subtour-elimination constraints (Miller *et al.*, 1960). The effective model of Goeke and Schneider (2021) is tailored to the single-block parallel-aisle warehouse layout. Independently, a rather involved MIP-based approach has been presented by Su *et al.* (2023) for multi-block parallel-aisle warehouses. Unfortunately, this approach has not been compared to any GTSP-based model or the approach of Pansart *et al.*. The currently fastest exact SPRP-SS algorithm is the one of Heßler and Irnich (2024) which is explained and exploited in Sections 2.2 and 2.3. Their approach assumes that the warehouse layout and I/O point configuration are such that a DP formulation for the non-scattered SPRP is known and can be extended to capture the options arising from scattered storage. The latter is true as can be seen from the last column of Table 2.1. A rather involved MIP model (solved with a MIP solver) has been presented by Su *et al.* (2023) for the SPRP-SS defined for multi-block parallel-aisle warehouses and the unit-demand case. Unfortunately, this approach has not been compared to any GTSP-based solution approach. Overall, all exact solution approaches for the SPRP-SS are MIP-based. Complementing exact approaches, Weidinger *et al.* (2019) presented a heuristic solution approach for SPRP-SS with multiple depots.

Class-based storage was first introduced by Hausman *et al.* (1976). Different criteria that are often used for the assignment of articles to classes are the *turnover rate*, the *required space*, and the *cube-per-order index* (COI) (Heskett, 1963) which is the ratio of required space to turnover rate (Gu *et al.*, 2007). Gibson and Sharp (1992) showed that class-based storage leads to significantly shorter picker tours than random storage. A special case of class-based storage is *full-turnover storage* where all articles are ranked according to their turnover rate. The articles are then

assigned to the pick positions in such a way that articles with the highest turnover rate are located closest to the depot and the articles with the smallest turnover rate are most distant to the depot. Thus, in this case, each article is assigned to a different class (de Koster *et al.*, 2007). Full-turnover storage leads to shorter picker tours than class-based storage but is more complex to implement and more detailed data is needed (Petersen *et al.*, 2004). Hence, Petersen *et al.* (2004) recommend class-based storage with two to four classes. Park and Webster (1989) consider three-dimensional storage systems (vertically and horizontally organized storage positions) with specific handling equipment and suggest a so-called 'cubic-in-time' rule for designing a two-class storage warehouse minimizing traveling time.

2.1.3 Contributions

This is the first (exact) approach for picker routing that combines heuristic routing policies with scattered storage. We formally introduce variants of the SPRP-SS in which picker routes must obey the rules of the heuristic routing policies `traversal`, `return`, `largest gap`, `midpoint`, and `composite`, respectively. The contributions of this work are:

- We prove that all variants of the SPRP-SS with the above heuristic routing policies and scattered storage remain NP-hard.
- Adapting the approach of Heßler and Irnich (2024) for the routing policy `exact` to heuristic routing policies, we provide a first exact approach for the SPRP-SS with the routing policies `traversal`, `return`, `largest gap`, `midpoint`, and `composite`. Our solution approach is based on an incomplete DP formulation finally solved as an *integer program* (IP) with the help of a *mixed-integer programming* (MIP) solver. In this context, 'incomplete' means that the DP formulation does not ensure demand covering for articles available in more than a single aisle.
- We analyze combinations of routing and storage policies. In this sense, our work extends classical results of Petersen and Schmenner (1999) to warehouses with scattered storage. We find that larger cost savings are possible with scattered storage by choosing a suitable combination of the routing and storage policy compared to *classical warehouses*, i.e., those without scattered storage. The combination of `largest gap` and *within-aisle* outperforms other combinations in most cases.
- For class-based storage policies, we investigate how the scattering of articles of different classes should be performed. For *uniformly distributed* articles, scattering of A-articles is most beneficial as suggested by Weidinger (2018). We also obtain an unexpected result: scattering of C-articles performs best for the standard class-based storage policies.

2.1.4 Structure

The remainder of this paper is structured as follows. In Section 2.2, we briefly review the DP formulation of Ratliff and Rosenthal (1983) and detail the necessary modifications on the DP state space to enforce picker routes that obey the rules of the respective heuristic routing policy. In Section 2.3, we elaborate which extensions of the state space are required to correctly model the options arising from scattered storage. Together, these results allow us to specify a unified integer programming formulation. The NP-hardness of the SPRP-SS even for simple heuristic routing policies is proven in Section 2.4. Section 2.5 formally introduces the class-based storage policies. The comprehensive computational study is presented in Section 2.6. The work closes with conclusions and an outlook in Section 2.7.

2.2 Solving the SPRP for Different Routing Policies

While the non-scattered standard SPRP can be solved efficiently, the SPRP-SS is provenly NP-hard (Weidinger, 2018). Different solution approaches for optimal routing (hereafter denoted as `exact` routing policy) have been presented in the literature (see Section 2.1.2). To date, we are not aware of any exact algorithm for the SPRP-SS and heuristic routing policies. In the following, we rely on an exact algorithm first presented in (Heßler and Irnich, 2022a) (this technical report will remain unpublished) and Heßler and Irnich (2024) (this companion paper has been submitted for publication). Their approach assumes that the warehouse layout is such that a DP formulation for the non-scattered SPRP is known and can be extended to capture the options arising from scattered storage. The focus of the companion paper is on algorithmic performance, as it compares the solution times of the available solution approaches for SPRP-SS applicable to one-block and two-block parallel-aisles warehouses and the `exact` routing policy. We would like to stress that the paper at hand has a completely different focus, i.e., different heuristic routing policies evaluated in combination with different storage policies. Particularly, computation times are irrelevant because they are in the range of milliseconds (never exceeding one second, see Section 2.6.2). To make the paper at hand self-contained, we present the ideas coined in (Heßler and Irnich, 2022a) in the following paragraph.

Heßler and Irnich (2022a, 2024) have shown that, for the routing policy `exact` in a single-block parallel-aisle warehouse, the SPRP-SS can be solved as a shortest-path problem with additional covering constraints. The underlying network is an extension of the state space of Ratliff and Rosenthal's DP. We adapt the approach to heuristic routing policies. Section 2.2.1 introduces the notation and original

state space, while Section 2.3 explains the necessary extension in the form of additional actions per aisle. In the extended state space, every picker tour is still a path, and vice versa. Exploiting the equivalence of DP and linear programming, a network-flow formulation for an origin-destination shortest path problem can be set up. The requirement to make consistent selections of pick positions can be modeled with additional covering constraints. The resulting IP formulation is also presented in Section 2.3.

In this work, we follow the same solution approach for solving the SPRP-SS but now for the different heuristic routing policies. The new idea presented here is that the picker tour defined by a heuristic routing policy is a shortest path in a DP state space that is adapted according to the routing policy. Accordingly, we construct state spaces for the heuristic routing policies (Sections 2.2.2–2.2.6). We summarize the results of this longer DP-related part in Section 2.2.7.

2.2.1 Routing Policy Exact

Let $J = \{1, 2, \ldots, m\}$ denote the set of aisles (numbered from left to right). The DP formulation of Ratliff and Rosenthal (1983) constructs a distance-minimal picker tour by alternately deciding how the picker moves through an aisle $j \in J$ and through the next cross-aisles connecting aisle j with aisle $j + 1$. Accordingly, the stages of the DP can be denoted by

$$1^-, 1^+, 2^-, 2^+, \ldots, m^-, m^+, (m+1)^-,$$

where j^- (j^+) represents the situation before (after) the aisle $j \in J$ is traversed. Likewise, j^+ and $(j+1)^-$ represent the situations before and after the cross-aisle action connecting aisle j with $j + 1$ has been performed, respectively. The stage $(m+1)^-$ is added to have a unique final state (see below).

For a distance-minimal picker tour, within an aisle, only the actions

$$E^{aisle} = \{\texttt{1pass}, \texttt{2pass}, \texttt{top}, \texttt{bottom}, \texttt{gap}, \texttt{void}\},$$

are possible, where `1pass` (`2pass`) stands for a single (double) traversal through the aisle (in either direction), `top` (`bottom`) for a traversal from the back (front) cross-aisle to the lowest (highest) relevant pick position and back, `gap` for double-sided traversal from front and back cross-aisle leaving a maximum length gap in the middle, and `void` for no traversal of the aisle, which is only possible in aisles from where nothing needs to be picked.

Possible cross-aisle actions are

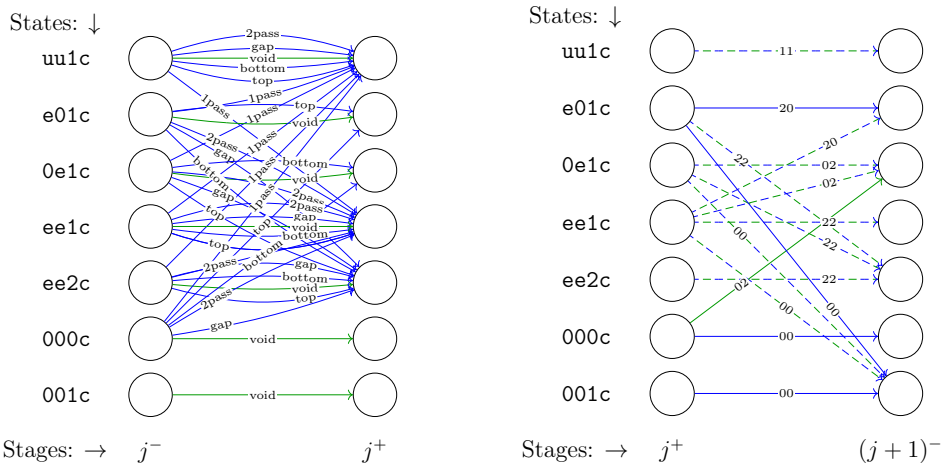
$$E^{cross} = \{\texttt{00}, \texttt{11}, \texttt{20}, \texttt{02}, \texttt{22}\},$$

where the first (second) digit gives the number of traversals of the back (front) cross-aisle.

Ratliff and Rosenthal introduced so-called *partial tour subgraphs* (PTSs) that represent the part of the picker tour containing only actions between aisle 1 and aisle j. A PTS results from introducing vertices a_j and b_j at the crossing points of aisle j and the back and front cross-aisle, respectively (see Figure 2.3). Every PTS can be characterized by the following set of states:

$$\mathscr{S} = \{\texttt{uu1c}, \texttt{e01c}, \texttt{0e1c}, \texttt{ee1c}, \texttt{ee2c}, \texttt{000c}, \texttt{001c}\}.$$

The first two symbols describe the parity of the right-most vertices a_j and b_j with `u`, `e`, and `0` for uneven (=odd), even (with positive degree), and degree 0. The last two symbols describe the number of connected components of the PTS, i.e., `1c` (`2c`) for one (two) connected components, and `0c` for an (empty) PTS without edges.



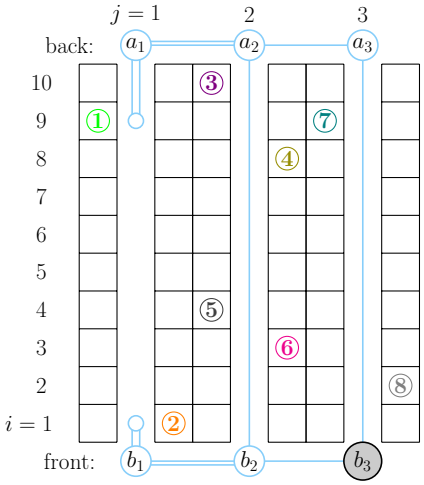
(a) Actions within an aisle j. Note that green (blue) arcs are only valid in (non-)empty aisles.

(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles.

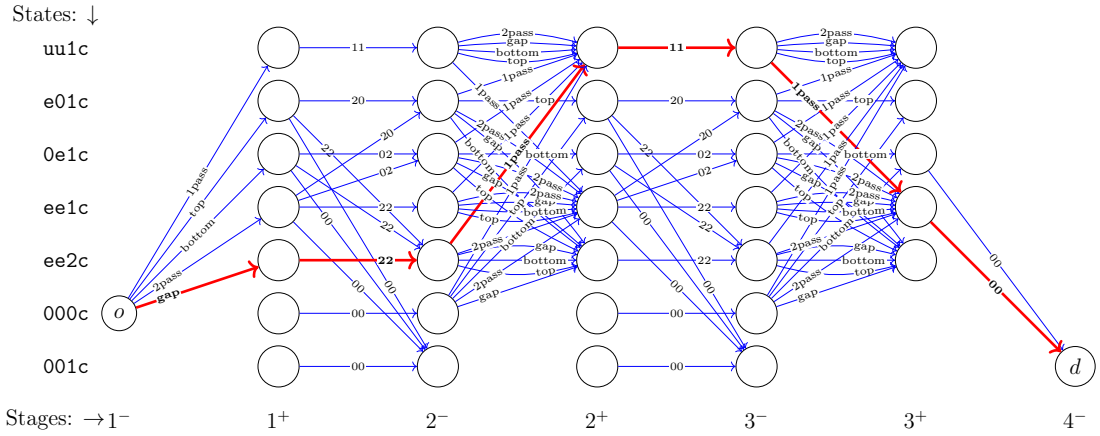
Figure 2.2: Possible actions between states for the routing policy `exact`.

Figure 2.2 visualizes the states and actions for the routing policy `exact`. The state space of Ratliff and Rosenthal's DP can be constructed by joining the building block depicted in Figure 2.2a for $j = 1$ with the building block depicted in

Figure 2.2b for $j = 1$, the latter one with the building block depicted in Figure 2.2a for $j = 2$, etc. The initial state is $o =$ `000c` at stage 1^-, and the final state is $d =$ `001c` at stage $(m+1)^-$. The resulting state space is depicted in Figure 2.3.



(a) Instance and optimal picker tour.



(b) State space and shortest o-d-path corresponding to the optimal picker tour (marked in red/thick).

Figure 2.3: SPRP instance with the optimal picker tour and the corresponding state space.

In contrast to Ratliff and Rosenthal, we do not model the depot 0, which is located at the front or back of the depot aisle j_0, as a pick position. Instead, the possible actions in cross-aisles can be restricted so that the depot is surely connected with the constructed tour, see the caption of Figure 2.2b. For exam-

ple, when being in state `e01c` at stage j^+, the actions `00` and `20` would make b_j disconnected, which is forbidden if j is the aisle j_0 of the depot 0 (in this case b_j represents the depot). Note that these restrictions are based on the assumption that the depot 0 is located at a front cross-aisle. In the case that the depot is at the back cross-aisle, the roles of back and front must be swapped. For simplicity, to not display symmetric cases, we assume in the following figures that the depot 0 is located at the front cross-aisle. Note also that our modeling approach is valid for any *depot aisle* $j_0 \in J$, not only if $j_0 = 1$, i.e., the depot is at the left end. Figure 2.3 illustrates an example SPRP instance where the depot is located in front of aisle $j_0 = 3$. Figure 2.3a pictures the warehouse with the depot, the pick positions of the different (numbered) articles, and the optimal picker tour. The corresponding state space is shown in Figure 2.3b, where the optimal control describing the optimal picker tour is marked in red. It is the shortest path from the initial state o to the final state d. Looking at the optimal picker tour shown in Figure 2.3a, we briefly discuss the PTS of each stage: At stage 1^-, the PTS does not include any parts of the picker tour. At stage 1^+, aisle $j = 1$ is traversed using the action `gap`. Thus, both vertices a_1 and b_1 have an even parity with a degree of two, leading to a PTS with state `ee2c`, as the PTS consists of two components. The subsequent PTS at stage 2^- does not change its state because passing the cross-aisle sections between a_1 and a_2 and b_1 and b_2 twice still results in two components with an even degree. At stage 2^+, aisle $j = 2$ is traversed using the action `1pass` changing the degree of a_2 and b_2 to three. As the PTS becomes connected, the corresponding state is `uu1c`. With the cross-aisle action `11`, a_3 and b_3 have degree one so that the state of stage 3^- is `uu1c`. Following the same procedure with actions `1pass` and `00`, the remaining PTSs of this instance have the states `ee1c` and `001c` for stages 3^+ and 4^-, respectively.

To complete the definition of the DP, one must associate a cost to all actions, i.e., the arcs of the state graph: the cost is defined by the distance traveled (or time consumed) by the picker when performing the associated action. Recently, Heßler and Irnich (2022b) have shown that computing these costs can be done with linear effort measured in the number m of aisles and number n of pick positions. Recall that any o-d-path in the state space represents a feasible picker tour. As a result, the length of the picker tour is exactly the cost of the path. Therefore, solving the SPRP can be interpreted as solving an origin-destination shortest-path problem. Note that the overall complexity is also linear because there is only a linear number of states and actions and the state graph is acyclic.

2.2.2 Routing Policy Traversal

The rules of `traversal` specify that an aisle is completely traversed when it is visited (Hall, 1993). This means that if an aisle is entered from the back (front),

it needs to be left at the front (back) cross-aisle. However, there is an exception regarding the rightmost aisle entered by the picker tour. In case an odd number of aisles is traversed in the course of the picker tour, this last aisle is not traversed completely, but entered and left from the same end. Thus, after collecting the articles, the picker is on the same side of the warehouse as the depot and can move back there without crossing an additional, unnecessary aisle. In particular, if the order only contains articles that are stored in one single aisle, then this aisle is not traversed completely.

For the remainder of this section, we assume the depot 0 to be located within the front cross-aisle at the level of the leftmost aisle, i.e., $j_0 = 1$. Then, possible actions within an aisle and possible cross-aisle actions for the routing policy `traversal` are

$$E^{aisle} = \{\texttt{1pass}, \texttt{bottom}, \texttt{void}\} \qquad \text{and} \qquad E^{cross} = \{\texttt{00}, \texttt{11}, \texttt{02}\}.$$

When the depot is located in the back cross-aisle, actions `top` and `20` replace `bottom` and `02`.

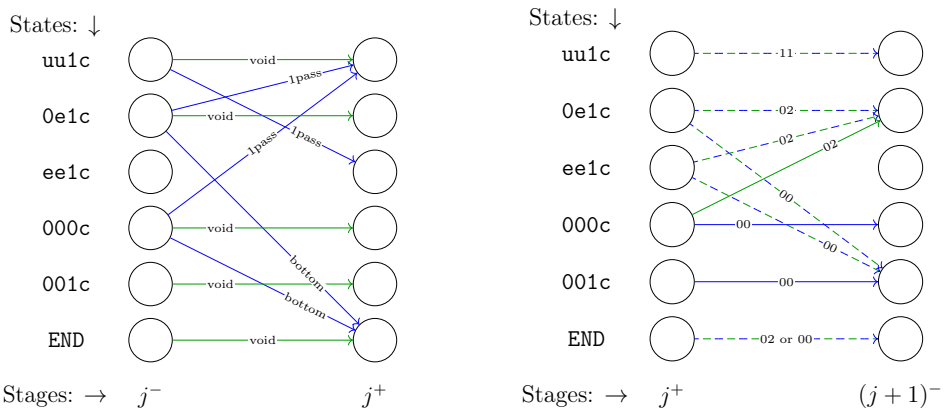
Due to the smaller sets of actions (compared to `exact`), the set of states for `traversal` has fewer elements. State `ee2c` represents a PTS consisting of two components and is impossible here (`ee2c` cannot be reached by any heuristic routing policy). Additionally, state `e01c` is omitted from the state set if the depot is located at the front cross-aisle, since with the allowed actions of `traversal`, the state `e01c` cannot be reached. Instead, `0e1c` is omitted if the depot is sited at the back cross-aisle. Apart from the two excluded states, there is also a supplementary state denoted by `END`. This state can only be reached in combination with the action `bottom` (or `top`, if the depot is located in the back cross-aisle) and is therefore responsible to enforce the above deviating rules for the last aisle entered by the picker tour. The resulting set of states is

$$\mathscr{S} = \{\texttt{uu1c}, \texttt{0e1c}, \texttt{ee1c}, \texttt{000c}, \texttt{001c}, \texttt{END}\}.$$

Figure 2.4 depicts which state-action combinations are possible. Note that some combinations only exist in (non-)empty aisles or (non-)depot cross-aisles. In addition, the action of the cross-aisles between two `END`-states varies depending on the location j_0 of the depot 0. If aisle j is left to the depot aisle j_0 (or identical to it), then the action is `02`: The picker has already collected all demanded items at this point, but still has to move to the depot. In case j is right to j_0, the action is `00`, and the picker does not need to move further but returns directly to the depot.

2.2.3 Routing Policy Return

The opposite routing policy to `traversal` is `return` (Petersen, 1997). Instead of traversing each relevant aisle completely, for `return` each relevant aisle is entered



(a) Actions within an aisle j. Note that `bottom` is only valid in the right-most non-empty aisle and green (blue) arcs are only valid in (non-)empty aisles.

(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles. The action from state `END` to `END` is `02` (`00`) for $j + 1 \leq j_0$ ($j + 1 > j_0$).

Figure 2.4: Possible actions between states for the routing policy `traversal`.

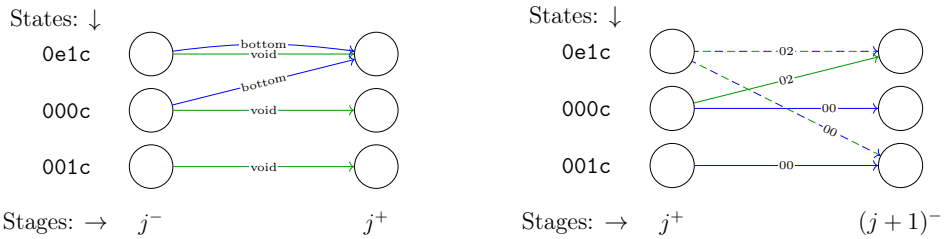
and left from the same end. The change of direction in the visited aisle is performed as soon as all requested articles in this aisle are collected. As the picker starts and ends the tour at the depot, the location of the depot determines the cross-aisle the picker passes through and, consequently, the end from which aisles are entered. This rather simple routing policy requires only two different actions each, within the aisles and for the cross-aisles. Not even for the last aisle of the picker tour is an additional action needed. Given the depot is located in the front cross-aisle, the two sets of actions are

$$E^{aisle} = \{\texttt{bottom}, \texttt{void}\} \qquad \text{and} \qquad E^{cross} = \{\texttt{00}, \texttt{02}\}$$

(if the depot is located in the back cross-aisle, instead of `bottom` the action `top` is used, and `02` is exchanged by `20`). The limited number of actions imposes three necessary states which are

$$\mathscr{S} = \{\texttt{0e1c}, \texttt{000c}, \texttt{001c}\}.$$

(if the depot is placed in the back cross-aisle, `0e1c` is exchanged by `e01c`).



(a) Actions within an aisle j. Note that green (blue) arcs are only valid in (non-)empty aisles.

(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles.

Figure 2.5: Possible actions between states for the routing policy `return`.

Figure 2.5 confirms that the small number of possible actions and states also leads to few possible combinations in the state space.

2.2.4 Routing Policy Largest Gap

The routing policy `largest gap` has rather complex rules (Hall, 1993): An aisle is entered (and left again) from both ends. The two turning points within the aisle are chosen in such a way that the longest possible part of the aisle is not traversed because there are no relevant articles. This part of the aisle is called *largest gap*. When applying `largest gap`, the picker moves through both cross-aisles one after the other and enters relevant aisles in between from the end that borders the cross-aisle. To reach the opposite cross-aisle, the picker traverses the first and last aisle with relevant pick positions completely. For the implementation of these routing rules, the actions

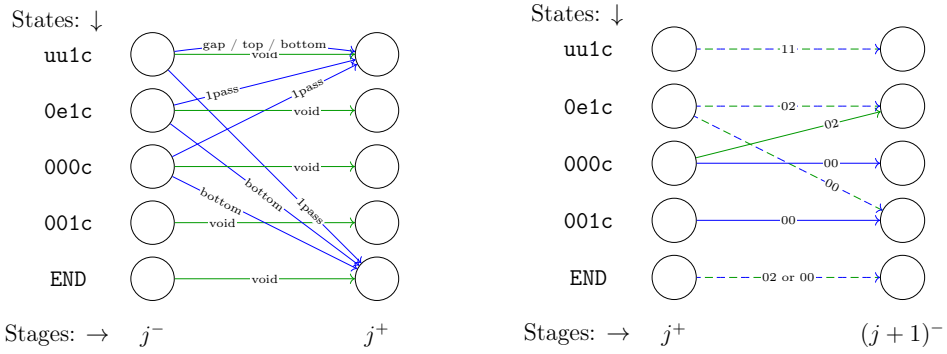
$$E^{aisle} = \{\texttt{1pass}, \texttt{top}, \texttt{bottom}, \texttt{gap}, \texttt{void}\} \qquad \text{and} \qquad E^{cross} = \{\texttt{00}, \texttt{11}, \texttt{02}\}$$

are required. Here, action `1pass` is only used for the first and last aisle. For the aisles in between, the use of `gap` is intended. If only one demanded article is stored in an aisle or if the largest gap of an aisle is connected to one of the two cross-aisles, then this aisle is only entered from one end and the action `top` or `bottom` is used instead of `gap`. Concerning the cross-aisle actions, `02` is replaced by `20`, if the depot is located in the back cross-aisle. The set of possible states necessary

for `largest gap` is

$$\mathscr{S} = \{\texttt{uu1c}, \texttt{0e1c}, \texttt{000c}, \texttt{001c}, \texttt{END}\}.$$

Since the first and last aisle are traversed differently than all the other aisles, this must be taken into account when defining the set of possible states. Thus, the additional state `END` is needed to enable the action `1pass` within the last aisle. For the complete traversal of the first aisle of the tour, no extra state is needed. As the initial state `000c` is only used for this aisle (and for empty aisles left of it), only the actions `1pass` and `void` are allowed at this state (see Figure 2.6a).



(a) Actions within an aisle j. Note that green (blue) arcs are only valid in (non-)empty aisles. Moreover, `1pass` and `bottom` with head `END` are only valid in the right-most non-empty aisle. Between two `uu1c` states is either `gap` or `top` or `bottom` allowed.

(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles. The action from state `END` to `END` is `02` (`00`) for $j+1 \leq j_0$ ($j+1 > j_0$).

Figure 2.6: Possible actions between states for the routing policy `largest gap`.

In case all articles of the pick list are stored in one single aisle, the picker moves through the cross-aisle of the depot until the corresponding aisle is reached (in general, in either direction from left to right or right to left). In Figure 2.6a, this case can be observed in the states `0e1c` and `000c` with action `bottom` assuming the depot is located at the front cross-aisle. This aisle is then served using the action `bottom` (otherwise with action `top`). Another noticeable state is `uu1c` for which it depends on the number and the distribution of the articles which action `gap`, `top` or `bottom` is possible (in a middle aisle of the tour). Looking at Figure 2.6b, the possible action between two `END`-states is again dependent on the depot location.

2.2.5 Routing Policy Midpoint

The routing policy `midpoint` (Hall, 1993) has many similarities to the just-described policy `largest gap`. The picker moves through the warehouse following almost all principles of `largest gap`. The only difference is the criterion for defining the turning points in aisles that are visited from both ends. While for `largest gap` the picker leaves out the maximum length part, the rules for `midpoint` require the picker to not cross the middle of an aisle. This means if the picker enters an aisle from the back (front) cross-aisle, he or she is only allowed to collect articles closer to the back (front) cross-aisle than to the front (back) cross-aisle or articles that are equidistant from both cross-aisles. The location of the turning points of `midpoint` can only lead to identical or longer tour lengths than obtained with `largest gap`. Nevertheless, `midpoint` also brings its advantages. While a picker can easily determine the turning points of each aisle on the go following the rules of `midpoint`, this quickly becomes difficult for `largest gap` as soon as several articles have to be collected in the same aisle.

The following actions are possible for `midpoint`:

$$E^{aisle} = \{\texttt{1pass}, \texttt{top}, \texttt{bottom}, \texttt{mgap}, \texttt{void}\} \qquad \text{and} \qquad E^{cross} = \{\texttt{00}, \texttt{11}, \texttt{02}\}.$$

Compared to the routing policy `largest gap`, the action `mgap` replaces `gap`, thus respecting the rules for the determination of the turning points of `midpoint`. Just as with `largest gap`, the actions `top` or `bottom` are used instead of `mgap` if there is only one requested article in an aisle or if the largest gap of this aisle is located at one end of this aisle.

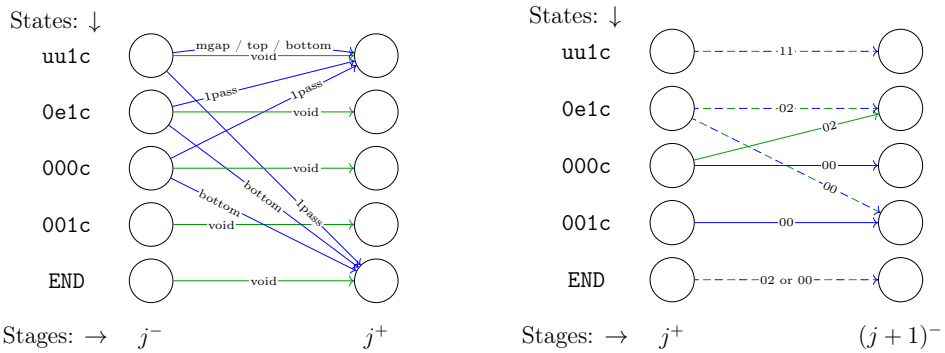
The possible states of `midpoint` remain the same compared to `largest gap`:

$$\mathscr{S} = \{\texttt{uu1c}, \texttt{0e1c}, \texttt{000c}, \texttt{001c}, \texttt{END}\}$$

The resulting possible combinations of states and actions for `midpoint` are almost identical to the combinations of `largest gap` (compare Figures 2.6 and 2.7). The only difference is the use of the aisle action `mgap` instead of `gap` between two states `uu1c` in Figure 2.7a which differ in the definitions of the turning points.

2.2.6 Routing Policy Composite

The policy `composite` is a combination of the routing policies `traversal` and `return`. That means the picker can either traverse an aisle completely or enter the aisle and move through it until all demanded articles are collected before returning to the same end again, where the aisle was entered. These two options are possible for each aisle that needs to be visited, regardless of whether it is one of the two aisles on the edge of the tour or not.



(a) Actions within an aisle j. Note that green (blue) arcs are only valid in (non-)empty aisles. Moreover, `1pass` and `bottom` with head `END` are only valid in the right-most non-empty aisle. Between two `uu1c` states is either `mgap` or `top` or `bottom` allowed.

(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles. The action from state `END` to `END` is `02` (`00`) for $j + 1 \leq j_0$ ($j + 1 > j_0$).

Figure 2.7: Possible actions between states for the policy `midpoint`.

To the best of our knowledge, there are two different definitions of `composite` in the literature. In both cases, the decision whether an aisle is visited with action `1pass`, `bottom`, or `top` is made in a greedy fashion. We first describe the two versions of the policy `composite` and then present the state space of a new version in which the distance-minimal route is generated when only allowing the actions `1pass`, `bottom`, `top`, and `void`.

Petersen (1995) introduced the routing policy `composite`. Here, the distance between the two most distant articles of two successive aisles is used as the decision criterion which action is chosen. To be more precise, in both aisles, this concerns the article that is furthest away from the picker's current position (crossing of the current aisle with back or front cross-aisle). Aisles without relevant articles are ignored. This may result in relevant successive aisles not being contiguous, but separated by irrelevant aisles. If the path resulting from applying `1pass` (`bottom` or `top`) between these two pick positions is shorter than the path resulting the other action, the former is chosen for the current aisle. If the two paths are of equal length, it is irrelevant which action is selected. According to this principle, a decision is made for each aisle on how it is passed through, starting with the aisle next to the depot. After all articles from the pick list are collected, the picker

moves back to the depot through the cross-aisle where the depot is located.

Scholz and Wäscher (2017) define the routing policy `composite` differently. If more than half of an aisle needs to be visited to collect all demanded articles in this aisle, then the action `1pass` is chosen. However, if the picker can collect all necessary articles in the first half of the aisle, the picker follows the action `bottom` or `top` (depending on the current position) and leaves the aisle at the same end again.

It should be noted that neither of the above definitions leads to an easy-to-compute tour. In addition, the resulting tour does not follow a simple pattern that can be easily remembered. We use a third and new version of `composite` in which the actions `1pass`, `bottom`, and `top` are combined optimally. More precisely, a distance-minimal picker tour is constructed, only allowing the possible actions of these two routing policies. For this purpose, a state space is set up which contains the possible actions of `traversal` and `return`.

This procedure should not be confused with another routing policy called `combined` (Roodbergen, 2001), which also combines the possible actions of `traversal` and `return` in another target-oriented manner. `combined` searches for a distance-minimal path starting at the depot and ending at the pick position of the last demanded article while collecting all other articles on the way. The way back to the depot leads through the cross-aisle of the depot. In contrast, our version of `composite` optimizes the entire picker tour and allows the collection of articles on the way back to the depot.

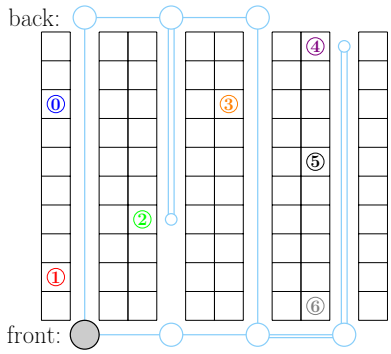
The state space resulting from our new definition of `composite` includes the picker tours according to the previous interpretations of `composite` as well as tours according to the `combined` policy. Thus, the corresponding picker tours dominate the others.

A visual representation of the differences in the definitions of `composite` and the routing policy `combined` can be found in Figure 2.8. In each version, the same instance is used in which the distance between two neighboring pick positions as well as the distance between the first (last) pick position of an aisle and the bordering cross-aisle is one. The routing costs show clear differences for the different definitions. The picker tour has a length of 68 when constructed according to the definition of Petersen but a length of 56 results from the definition of Scholz and Wäscher. Here, the picker tour is identical to the one resulting from the routing policy `traversal`. Our new version of `composite` leads to a tour length of 48. In contrast, the picker tour resulting from `combined` has a length of 54.

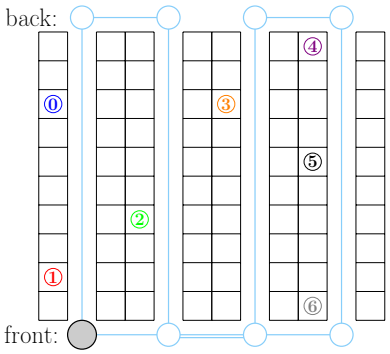
Our version of `composite` has the following the sets of actions:

$$E^{aisle} = \{\texttt{1pass}, \texttt{top}, \texttt{bottom}, \texttt{void}\} \qquad \text{and} \qquad E^{cross} = \{\texttt{00}, \texttt{11}, \texttt{20}, \texttt{02}\}.$$

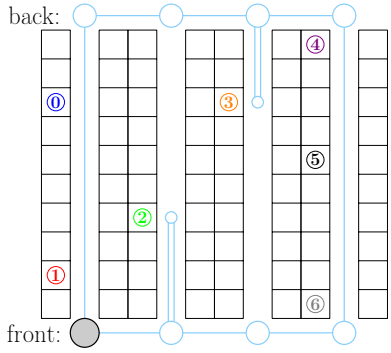
These two sets include all relevant actions of `traversal` and `return` for the both



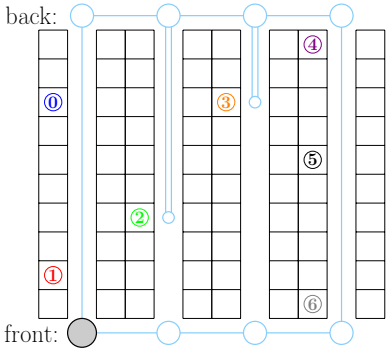
(a) Routing policy `composite` as defined by Petersen (1995).



(b) Routing policy `composite` as defined by Scholz and Wäscher (2017).



(c) Our new definition of `composite`.



(d) Routing policy `combined` (Roodbergen, 2001).

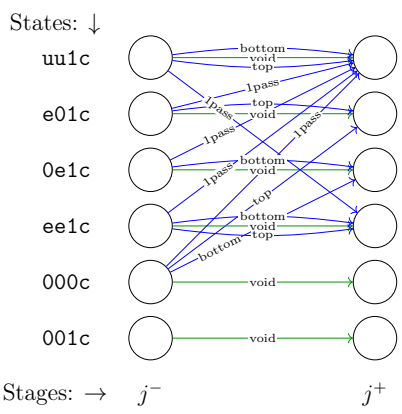
Figure 2.8: Different definitions of the routing policy `composite` in comparison with the routing policy `combined`.

possible scenarios of the depot being located in the back or front cross-aisle. The possible actions of `composite` are not dependent on the position of the depot. The set of states is

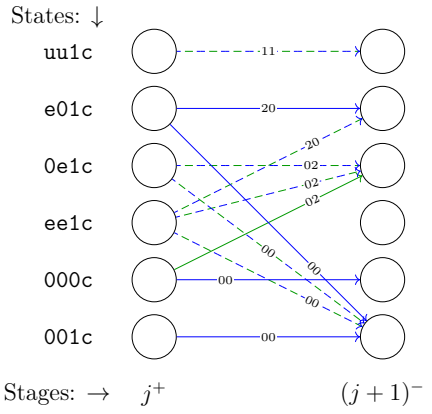
$$\mathscr{S} = \{\texttt{uu1c}, \texttt{e01c}, \texttt{0e1c}, \texttt{ee1c}, \texttt{000c}, \texttt{001c}\}$$

and contains almost all states required for the routing policy `exact`. Solely `ee2c` is missing. Furthermore, no additional state is needed for the first or last aisle of the picker tour because the rules of `composite` treat every aisle equally.

Although this routing policy already allows several different actions and needs only one state less than `exact`, the number of possible state-action combinations is substantial smaller for `composite`. Comparing Figures 2.2 and 2.9, this difference



(a) Actions within an aisle j. Note that green (blue) arcs are only valid in (non-)empty aisles.



(b) Actions of the cross-aisles. Note that green (blue) arcs are only valid in (non-)depot cross-aisles, i.e., if $j = j_0$ ($j \neq j_0$). Dashed arcs are valid in all cross-aisles.

Figure 2.9: Possible actions between states for the routing policy `composite`.

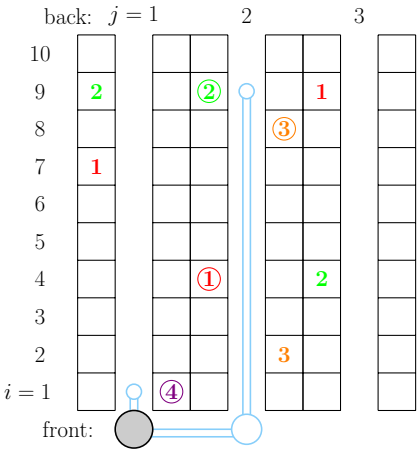
is clearly visible.

2.2.7 Summary

Up to now, we have established DP formulations for the SPRP restricting picker routes to the rules of the different heuristic routing policies used in practice. These DP formulations may seem dispensable, since heuristic routing policies are simple route-construction procedures. However, a recent result of Heßler and Irnich (2022b) is applicable here showing that, if properly implemented, all DPs can be constructed and solved in linear time (measured in the number of aisles of the warehouse and the number of given pick positions). Even more, the following section shows that also the more difficult variants of the problem with scattered storage can be solved with a structurally simple integer-programming formulation based on the state spaces that have been presented in this section.

2.3 Extensions of the State Spaces and IP Formulation for Scattered Storage

The SPRP-SS distinguishes from the basic SPRP by the additional flexibility that one can select a pick position for a requested article whenever this article is stored at two or more positions. For capturing these options, the state space must be extended (Heßler and Irnich, 2024). We again present this idea as in Section 2.2 for a standard warehouse with parallel aisles and a single block layout. Compared to the SPRP, the number of stages and the states within each stage remain identical, i.e., those of the state space of Ratliff and Rosenthal. Moreover, cross-aisle actions E_j^{cross} remain identical (connecting stages j^+ and $(j+1)^-$), and also the aisle actions `1pass` and `2pass` remain unchanged. However, additional aisle actions have to be added what we explain with the help of the following example.



(a) The warehouse and optimal picker tour.

Aisle	Type of	additional aisle actions
$j=1$	`bottom`(i)	Cells $i \in \{1,7\}$
	`gap`(h,i)	Cell $(h,i)=(1,9)$
$j=2$	`top`(i)	Cells $i \in \{4,8\}$
	`bottom`(i)	Cells $i \in \{2,4,8^*\}$
	`gap`(h,i)	Cells $(h,i) \in \{(2,8)^\ddagger,(2,9),(4,9)\}$
$j=3$	`top`(i)	Cell $i=9$
	`bottom`(i)	Cell $i=4$
	`void`	

(b) Additional aisle actions compared to non-scattered storage. $*$: dominated by `bottom`(4). ‡: dominated by `gap`$(2,9)$.

Figure 2.10: Instance of the SPRP-SS. The pick list contains four articles $S = \{1,2,3,4\}$ (unit demand); pick operations are encircled.

Example 1. *An instance of the SPRP-SS with four requested articles* $S = \{1,2,3,4\}$ *is depicted in Figure 2.10a. For simplicity, we discuss the unit-demand case first, i.e.,* $q_s = 1$ *for all* $s \in S$*. In aisle* $j = 1$*, it is possible to pick only article 4 because articles 1 and 2 are also available in other aisles. The associated new aisle action is* `bottom`(1) *which is the traversal from the front cross-aisle with a U-turn in cell* $i = 1$*. Moreover, picking only articles 1 and 4 or articles 2 and 4 is feasible leading to the additional aisle actions* `bottom`(7) *and* `gap`$(1,9)$ *(the latter is leaving out the position* $i = 7$ *where article 1 is stored).*

These are all additional aisle actions as listed in the table in Figure 2.10b. Note that any action $\texttt{gap}(i,k)$ *can be disregarded if* i *and* k *are neighboring positions with a non-maximal gap. As in the DP of Ratliff and Rosenthal, these actions are dominated by one with maximum gap.*

Also in aisle $j = 2$*, it is not required to collect articles 1 and 2. However, article 3 must be collected, either from position* $i = 2$ *or* $i = 8$ *making* `void` *and* $\texttt{top}(9)$ *impossible. With* $\texttt{top}(4)$ *and* $\texttt{gap}(4,9)$ *we can collect all three articles 1, 2, and 3. The aisle actions* $\texttt{top}(8)$*,* $\texttt{gap}(2,8)$*, and* $\texttt{gap}(2,9)$ *can pick articles 2 and 3 but not 1. In this case,* $\texttt{gap}(2,8)$ *is dominated by* $\texttt{gap}(2,9)$ *in the sense that the latter is less costly but can collect the same articles (in the unit-demand case). Articles 1 and 3 are collected with* $\texttt{bottom}(4)$ *and* $\texttt{bottom}(8)$ *where the latter is dominated by the first. Lastly, with* $\texttt{bottom}(2)$*, only article 3 can be picked.*

It is possible to not enter the last aisle $j = 3$*, making* `void` *a feasible option here. Moreover, additional aisle actions collecting only article 1 or article 2 are* $\texttt{top}(9)$ *and* $\texttt{bottom}(4)$*, respectively.*

Overall, the set of aisle actions becomes aisle dependent. Hence, we denote by E_j^{aisle} the resulting set of arcs connecting stages j^- and j^+ for all $j \in J$. Several aisle actions of the same type referring to the same aisle can be considered as parallel arcs in the DP state space (in the above example, two parallel aisle actions `top` in aisle 2 differing in cost and articles that can be collected). Please note that, compared to the basic SPRP discussed in Sections 2.2.1–2.2.6, the aisle actions referring to empty aisles (depicted in green in Figures 2.2–2.7 and 2.9) are now also possible for (some) non-empty aisles.

As mentioned above, the set of additional aisle actions can be reduced by dominance considerations. In the example in aisle $j = 2$, $\texttt{gap}(2,8)$ is dominated by $\texttt{gap}(2,9)$ because of its higher cost. This is only true in the unit-demand case because otherwise, $\texttt{gap}(2,8)$ could provide more items of article 3. Likewise, $\texttt{bottom}(8)$ is dominated by $\texttt{bottom}(4)$ in aisle $j = 2$.

For scattered storage with general-demand, any aisle action $e \in E_j^{aisle}$ that leaves out some positions in aisle j is feasible if and only if the quantity that can be collected from this aisle together with all quantities stored in other aisles $j' \neq j$ is sufficient to fulfill the requested demand of the pick list.

In all cases (unit-demand and general-demand), we define b_{se} as the quantity of article $s \in S$ that can be collected when traversing the aisle via aisle action $e \in E_j^{aisle}$. Cross-aisle actions have zero supply $b_{se} = 0$ for all $e \in E_j^{cross}$ and $j \in J$. For a given article $s \in S$, we denote the set of edges with a non-negative quantity b_{se} by E_s.

Example 2. *(continued from Example 1) For the SPRP-SS instance depicted in Figure 2.10, we now assume that each position stores three units of each articles of the respective indicated type (the general-demand case). Then, overall, nine, nine,*

six, and three articles 1, 2, 3, and 4 are stored in the warehouse, respectively. Any demand $q = (q_1, q_2, q_3, q_4) \leq (9, 9, 6, 3)$ *can be retrieved from a picker tour. The picker tour depicted in Figure 2.10a can collect* $(3, 3, 6, 3)$ *units. This picker tour uses in aisle* $j = 1$ *the aisle action* `bottom`(1) *with positive supply value* $b_{se} = 3$ *for* $s = 4$ *and* $e =$ `bottom`$(1) \in E_1^{aisle}$*. In aisles* $j = 2$*, the aisle action* `bottom`(9) *has positive supply values* $b_{1e} = 3$*,* $b_{2e} = 3$*, and* $b_{3e} = 6$ *for* $e =$ `bottom`$(9) \in E_2^{aisle}$*. In aisle* $j = 3$*, the aisle action* `void` *has no positive supply values.*

We can now formalize the DP-based formulation for the SPRP-SS. Recall that we have stages $1^-, 1^+, 2^-, 2^+, \ldots, m^-, m^+, (m+1)^-$. Let V denote the set of all states of all stages, i.e., the vertices of state space. In particular, there are two distinct states, the initial state $o =$ `000c` at stage 1^- and the final state is d at stage $(m+1)^-$. The latter state is either $d =$ `001c` or $d =$ `END` depending on the routing policy, see Section 2.2. Moreover, let E denote the (disjoint) union of all aisle and cross-aisle actions, i.e.,

$$E = \bigcup_{j=1}^{m} \left(E_j^{aisle} \cup E_j^{cross} \right).$$

For each $e \in E$, let c_e be the length of the corresponding part of a picker tour described by e. The discrete linear formulation uses binary variables x_e for all $e \in E$ to indicate whether the optimal picker tour contains the respective part of a walk described by e. Heßler and Irnich (2024) propose the following model:

$$\min \sum_{e \in E} c_e x_e \tag{2.1a}$$

$$\text{subject to} \quad \sum_{e \in \delta^+(\sigma)} x_e - \sum_{e \in \delta^-(\sigma)} x_e = \begin{cases} +1, & \text{if } \sigma = o \\ -1, & \text{if } \sigma = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall \sigma \in V \tag{2.1b}$$

$$\sum_{e \in E_s} b_{se} x_e \geq q_s \qquad \forall s \in S \tag{2.1c}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{2.1d}$$

The objective (2.1a) is the minimization of the length of the resulting picker tour. The flow-conservation constraints (2.1b) ensure that the selected aisle and cross-aisle actions together describe a o-d-path in the extended state space (V, E). For any state $\sigma \in V$, $\delta^+(\sigma)$ and $\delta^-(\sigma)$ denote the set of arcs leaving and entering state σ, respectively. Demand fulfillment, i.e., that the requested number q_s of articles $s \in S$ can be collected with the constructed tour, is guaranteed by (2.1c). The domain of the flow variables x_e is given by (2.1d). Note that (2.1a), (2.1b), and (2.1d) is the standard model of an origin-destination shortest-path problem.

Example 3. *(continued from Example 2) We pursue the example depicted in Figure 2.10 and this time use the different routing policies presented in Section 2.2. Here, we have the seven stages* 1^-, 1^+, 2^-, 2^+, 3^-, 3^+, *and* 4^-.

For the routing policy `exact`, *the initial state is* $o =$ `000c` *at stage* 1^-, *and the final state is* $d =$ `001c` *at stage* 4^-. *The optimal solution depicted in Figure 2.10a can be described by the corresponding sequence*

$$(\texttt{000c}, \texttt{0e1c}, \texttt{0e1c}, \texttt{0e1c}, \texttt{001c}, \texttt{001c}, \texttt{001c})$$

of states. The optimal control is $(\texttt{bottom}(1), \texttt{02}, \texttt{bottom}(9), \texttt{00}, \texttt{void}, \texttt{00})$.

For the routing policy `return`, *the exact same control* $(\texttt{bottom}(1), \texttt{02}, \texttt{bottom}(9), \texttt{00}, \texttt{void}, \texttt{00})$ *is also optimal (feasibility can be tested with the help of Figure 2.11), leading to the same sequence of states that the o-d-path travels through and the same final picker tour.*

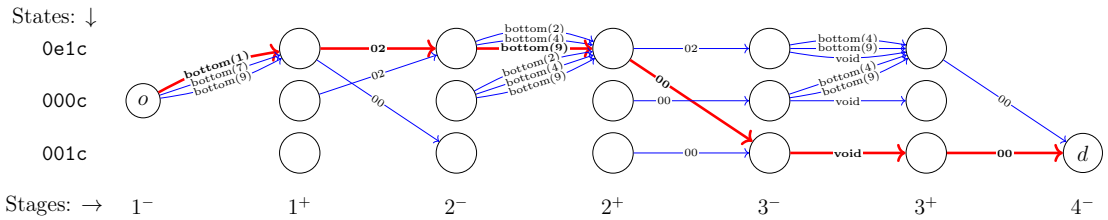


Figure 2.11: State space for the SPRP-SS instance depicted in Figure 2.10a and routing policy `return`. The shortest o-d-path corresponding to the optimal picker tour is marked in red/thick. Note that only paths collecting all articles $S = \{1, 2, 3, 4\}$ are feasible.

For the routing policies `traversal`, `largest gap`, *and* `midpoint`, *the picker tour depicted in Figure 2.10a is not feasible. Applying the same control* $(\texttt{bottom}(1), \texttt{02}, \texttt{bottom}(9), \texttt{00}, \texttt{void}, \texttt{00})$ *is infeasible, since the action* $\texttt{bottom}(1)$ *would let the path transit through the state* `END` *where it remains with action* `02`. *From state* `END`, *the action* $\texttt{bottom}(9)$ *is not allowed, see Figures 2.4a, 2.6a, and 2.7a.*

Finally, we consider the routing policy `composite` *with the three different underlying definitions explained in Section 2.2.6. All three definitions lead to the picker tour depicted in Figure 2.10a. The control* $(\texttt{bottom}(1), \texttt{02}, \texttt{bottom}(9), \texttt{00}, \texttt{void}, \texttt{00})$ *consists of two* `bottom` *actions and one* `void` *action within the aisles. In general, these actions are possible when applying* `composite`. *Regarding our new definition of* `composite`, *the state space constructed from the actions of Figure 2.9 allows the solution shown in Figure 2.10a. Following the definitions of* `composite` *by Petersen (1995) and Scholz and Wäscher (2017), this picker tour is also feasible.*

Algorithm 1 summarizes the solution approach described so far. The construction of the state space of the SPRP in Step 0 depends on the warehouse layout,

the routing policy `plc` as described in Sections 2.2.1–2.2.6, and the given positions of the SKUs. Note that the destination vertex d depends on the routing policy `plc` (Step 2, state `001c` or `END`). The computation of additional aisle actions in Step 3 has been explained in Section 2.3. In Steps 5 and 6, the costs c_e and the supply values b_{se} depend on the given positions of the SKUs and the warehouse layout. The final picker tour is a closed walk w over all parts that result from the chosen aisle actions, i.e., variables x_e with value one of the IP-formulation (2.1) solved in Step 7. The efficient computation of the walk w constructed in Step 8 is explained in (Ratliff and Rosenthal, 1983, Sect. 4).

Algorithm 1 Exact Algorithm for SPRP-SS

Input: SPRP-SS instance, i.e., layout $(m, C,$ distances$)$, routing policy `plc`, demands q_s, and positions of SKUs with supply quantities
1: Construct DP state space (V, E^{SPRP}) for SPRP
2: Set origin o and destination d
3: Construct additional aisle actions E^{SS} for SPRP-SS
4: Set $E = E^{SPRP} \cup E^{SS}$
5: Compute c_e for all $e \in E$
6: Compute b_{se} for all $s \in S$ and $e \in E$
7: Solve formulation (2.1) over (V, E, o, d) with a MIP solver
8: Construct picker tour w from MIP solution
Output: Picker tour w

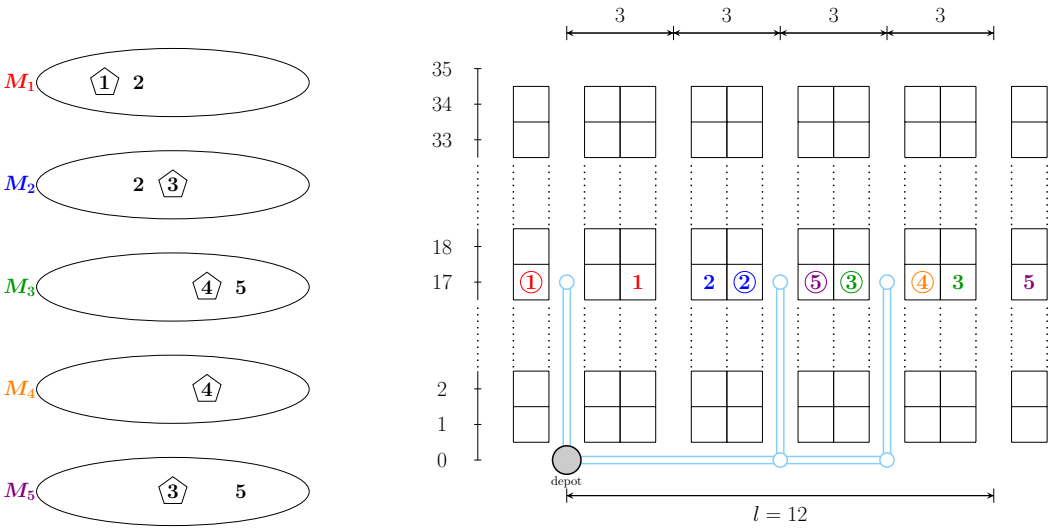
2.4 NP-Hardness of the SPRP-SS

As mentioned above, Weidinger (2018, Theorem 1) has shown that computing a distance-minimal picker tour (policy `exact`) in a one-block parallel-aisle warehouse for the SPRP-SS is NP-hard. We now show an even stronger result, i.e., the problem remains NP-hard even if a shortest picker tour for one of the commonly used heuristic routing policies has to be determined. The following proposition summarizes our findings about the computational complexity of the SPRP-SS.

Proposition 1. *The SPRP-SS is strongly NP-hard for the routing policies* `traversal`, `return`, `midpoint`, `largest gap`, *and* `composite` *considering a one-block parallel-aisle warehouse.*

Proof. The proof is analog to (Weidinger, 2018, Theorem 1) through a reduction from the hitting set problem that is strongly NP-complete. We show that the existence of a hitting set is equivalent to the existence of a picker tour of a certain maximal length.

The hitting set problem is defined as follows (Garey and Johnson, 1979): Given a finite set T, a positive integer $k \leq |T|$, and a set of subsets $\{M_1, \ldots, M_n\}$ with $M_i \subseteq T$ for all $i \in \{1, \ldots, n\}$, find a set $H \subseteq T$ such that $|H| \leq k$ and $H \cap M_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$.



(a) Hitting set problem with $T = \{1, 2, 3, 4, 5\}$, $k = 3$, and sets $M_1, \ldots, M_5$ as shown. The solution is the hitting set $H = \{1, 3, 4\}$.

(b) The corresponding picker routing problem with $m = 5$ aisles and $C = 34$ cells per aisle. Article i is stored in aisle j if and only if $j \in M_i$. The length of the plotted optimal picker route for the policies `return`, `composite`, and `midpoint` is $120 \leq kC + 2l = 126$. Aisles 1, 3, and 4 are entered.

Figure 2.12: Example for the transformation between the hitting set problem and the picker routing problem. The hitting set corresponds to the entered aisles.

Given an instance $\langle T, k, \{M_1, \ldots, M_n\}\rangle$ of the hitting set problem, a corresponding instance of the SPRP-SS can be constructed as follows. We consider a single-block warehouse with $m = |T|$ aisles and $C = 2l + 2|T|$ cells per aisle. The depot is located at the front-end of the leftmost aisle. Let the (vertical) distance between two neighboring cells be 1 and the (horizontal) distance between two neighboring aisles be 3. Then, $l = 3(|T| - 1)$ is the distance between the first and last aisle (see Figure 2.12). All n articles that need to be collected are stored in the lower middle cell $c = l + |T| + 1$. Article i is stored in aisle j if and only if $j \in M_i$. The figure illustrates how the SPRP-SS instance is constructed. We now prove that the existence of a hitting set of size $|H| \leq k$ is equivalent to the existence of a picker tour of maximal length $k(C + 1) + 2l$.

First, we consider the routing policies `return`, `composite`, and `midpoint`.

'⇒': Given a hitting set of size $|H| \leq k$. According to the routing policy, we construct the corresponding picker tour that enters only the aisles $i \in H$, i.e., not more than k aisles. For the policies `return`, `composite`, `midpoint`, and the constructed instance, this picker tour consist of `bottom` actions only for collecting articles. The traveled distance is upper bounded by

$$\underbrace{|H|C}_{\texttt{bottom actions}} + \underbrace{2l}_{\text{cross-aisle actions}} \overset{|H|\leq k}{\leq} k(C+1)+2l, \tag{2.2}$$

where the first summand is the distance of $|H|$ `bottom` traversals and the second summand is an upper bound on the horizontal cross-aisle traversal distance. This shows the first direction.

'⇐': Given a picker tour of maximal length $k(C+1)+2l$ collecting all articles. It must be shown that the picker tour enters not more than k different aisles. We prove this direction by contradiction: If $k+1$ or more aisles were entered, at least one article was collected in each aisle. This results in a picker tour of length not shorter than

$$\underbrace{(k+1)C}_{\texttt{bottom actions}} + \underbrace{2 \cdot 3 \cdot k}_{\text{cross-aisle actions}} >$$

$$kC + C = kC + 2l + 2|T| \overset{k\leq|T|}{\geq} kC + 2l + k = k(C+1) + 2l \tag{2.3}$$

contradicting with the maximal tour length of $k(C+1)+2l$. Consequently, the picker collects the articles from not more than k different aisles. Defining the hitting set H as the set of all entered aisles, we have proven this direction, too.

Second, we consider the routing policy `traversal`.

'⇒': The shape of the `traversal` picker tour that enters only the aisles $i \in H$ depends on whether $|H|$ is even or odd. If $|H|$ is even, the length is at most $|H|(C+1)+2l \leq k(C+1)+2l$, where the first summand is the distance of $|H| \leq k$ `1pass` traversals and the second summand is an upper bound on the horizontal cross-aisle traversal distance. If $|H|$ is odd, the picker tour consists of $k-1$ `1pass` traversals and one `bottom` traversal, which amounts a shorter tour length of $(|H|-1)(C+1)+C+2l \leq k(C+1)+2l$.

'⇐': As in the first case, if $k+1$ or more aisles were entered, the contradiction results from replacing (2.3) by

$$\underbrace{k(C+1)}_{\texttt{1pass actions}} + \underbrace{C}_{\texttt{bottom actions}} + \underbrace{2 \cdot 3 \cdot k}_{\text{cross-aisle actions}} > k(C+1) + C \overset{C=2l+2|T|}{>} k(C+1)+2l. \tag{2.4}$$

Third, we consider the routing policy `largest gap`.

'⇒': The `largest gap` picker tour that enters only the aisles $i \in H$ has the following shape: 2 `1pass` traversals in the first and last non-empty aisle and $|H| - 2$ `largest gap` traversals that are actually `bottom` traversals because only one position (at x) stores articles in a non-empty aisle. The distance of this optimal picker tour is at most

$$2(C+1) + (|H|-2)C + 2l \overset{|H|\leq k}{\leq} 2(C+1) + (k-2)C + 2l \leq k(C+1) + 2l, \qquad (2.5)$$

where the first summand is the two traversals in the first and last non-empty aisle, the second summand is the distance of $|H| - 2$ `bottom` traversals, and the third summand is an upper bound on the horizontal cross-aisle traversal distance. This proves the first direction.

'⇐': As in the first case, if $k+1$ or more aisles were entered, the contradiction results from replacing (2.3) by

$$\underbrace{2(C+1)}_{\texttt{1pass}\text{ actions}} + \underbrace{(k-1)C}_{\texttt{gap}\text{ actions}} + \underbrace{2 \cdot 3 \cdot k}_{\text{cross-aisle actions}} > k(C+1) + C \overset{C=2l+2|T|}{>} k(C+1) + 2l.$$

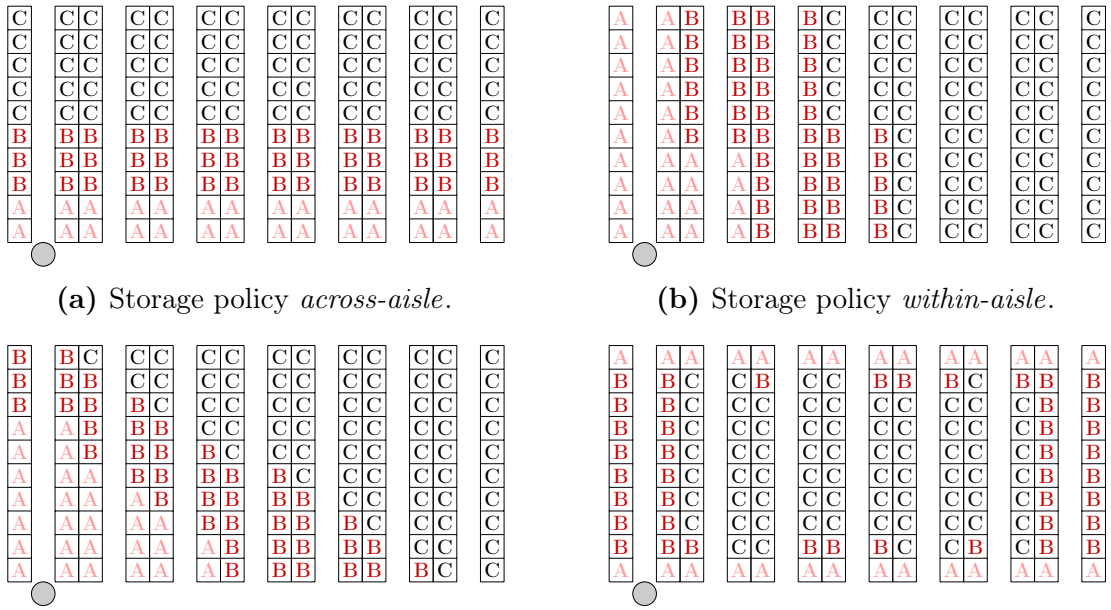
□

2.5 Class-Based Storage Policies

De Koster *et al.* (2007) categorize several methods for the assignment of articles to pick positions. One type of these methods are the *class-based storage policies*, where each article is assigned to a pick position according to its value, e.g., the turnover rate, see Section 2.1.2. For this purpose, articles are classified into one of the classes A, B, or C based on their value. As commonly assumed, 20% of all articles with the highest value are assigned to class A, the next 30% are assigned to class B, and the remaining 50% of the articles belong to class C. We often observe then that classes A, B, and C are composed of approximately 80%, 15%, and 5% of the articles, respectively.

The pick positions of the warehouse are grouped into three zones (one zone for each class), and the different class-based storage policies vary in how the zones are defined. As in (Petersen and Schmenner, 1999), we consider the typical class-based policies *across-aisle*, *within-aisle*, *perimeter*, and *diagonal*, for which the corresponding distribution schemes are illustrated in Figure 2.13. As a counterpart to the class-based policies, we include the storage policy *uniformly distributed* in the evaluation, which has already evolved into the most common storage policy in the last millennium (Petersen and Schmenner, 1999, p. 483). For the latter policy *uniformly distributed*, the articles of all three classes are distributed randomly in

the warehouse regardless of the classification.



(a) Storage policy *across-aisle*.

(b) Storage policy *within-aisle*.

(c) Storage policy *diagonal*.

(d) Storage policy *perimeter*.

Figure 2.13: Class-based storage policies.

The policies *across-aisle*, *within-aisle*, and *diagonal* arrange the zones in the warehouse such that zone A is positioned closest to the depot. Zone B follows after zone A, and zone C generally comprises pick positions that are farthest from the depot. The specific assignment of the pick positions to the zones depends on the distance measure used. In policy *across-aisle*, zone A includes the pick positions of each aisle that are closest to the cross-aisle of the depot. The remaining pick positions are allocated to zones B and C according to the same principle. In policy *within-aisle*, pick positions are assigned to zones aisle by aisle and pick position by pick position (as a secondary criterion): Starting with the depot aisle and the position closest to the depot cross-aisle, the remaining pick positions are assigned to the zones. When zone A (B) comprises sufficiently many pick positions, the next pick position is assigned to zone B (C). In policy *diagonal*, all pick positions are sorted by ascending distance to the depot in a first step. Afterwards, the first 20% of the pick positions are allocated to zone A, and the remaining 30% and 50% of the pick positions are added to zone B and C, respectively.

In policy *perimeter*, the warehouse is split up into zones from the outside (zone A) to the inside (zones B and C): Zone A comprises the pick positions around the perimeter of the warehouse, while zone C stores articles in the center

of the warehouse. Zone B contains the remaining pick positions in between, arranged in a ring-shaped fashion. Overall, pick positions are assigned to zones A to C by moving from the outside to the inside of the warehouse. To exactly respect the predefined proportions of the zones (20%, 30%, and 50%), we used the following procedure for generating the three zones A, B, and C:

1. *Select alternately the outer horizontal and vertical layer that are not yet assigned:*
   The horizontal layer k includes the kth and the kth-last pick position of each aisle. In contrast, the vertical layer k includes all pick positions in the kth and the kth-last aisle. The selection of layers is then: horizontal layer 1, vertical layer 1, horizontal layer 2, vertical layer 2 etc.
2. *Assign zones to the (unassigned) pick positions of the chosen layer in the order A, B, C:*
   If all unassigned pick positions of the layer can be added to the considered zone without exceeding the given size, they are assigned to the zone. Otherwise, if only a portion of the pick positions of the layer can be assigned to the zone, randomly select pick positions from the layer and assign them to the considered zone until the given size is reached. In the latter case, repeat the procedure with the remaining unassigned pick positions of the layer and the next zone.

2.6 Computational Study

In this section, we present the results of our extensive computational study and analyze which combinations of routing and storage policies are beneficial for a given warehouse layout and length of order list. The state space and the resulting IP formulation are implemented in C++ and compiled in release mode into a 64-bit single-thread code under MS Visual Studio 2017. We use the callable library `CPLEX` 20.1 for optimizing the IP formulation (2.1a)–(2.1d). The default values are kept for all parameters except from setting the number of threads to one. The computational study was executed on a 64-bit Microsoft Windows 10 computer with an Intel® Core™ i7-5930k CPU clocked at 3.5 GHz and 64 GB of RAM.

2.6.1 Instances

To the best of our knowledge, there are no benchmark instances combining scattered storage with class-based storage policies publicly available. Therefore, we generated an extensive set of instances for our computational study. We consider a single-block warehouse with parallel aisles and two bordering cross-aisles, see Figures 2.10a, 2.12b, and 2.13. The depot is located in the front cross-aisle at

the level of the first aisle from the left, and the distance between the depot and the bottom cell in the first aisle is 1 unit. Likewise, the distance between two neighboring pick positions of an aisle is always 1 unit. The distance between two neighboring aisles is 3 units. Finally, switching between consecutive aisles amounts to 5 units (first cell to first cell or last cell to last cell).

Parameter	Value(s)
Number of aisles m	10
Number of pick positions C (per aisle)	40
Max. storage capacity per pick position	10 articles
Number of different articles n	1000
Relative sizes of zones A, B, and C	20%, 30%, and 50%
Overall scatter factor α	2
Number of different pairs of article and pick position αn	2000
Scatter factor settings $(\alpha_A/\alpha_B/\alpha_C)$	(6/1/1), (1/1/3), (2/2/2)
Number of articles to pick = length of pick list	5, 15, 50
Distribution of A-, B-, and C-articles in pick list	80%, 15%, and 5%

Table 2.2: Parameters used for the generation of SPRP-SS instances.

For the entire computational study, we fixed the warehouse layout to $m = 10$ aisles with $C = 40$ pick positions per aisle. At each pick position, there are two racks from which stored articles can be collected (left- and right-hand side are irrelevant for distance calculations). In total, there are 800 pick positions in the warehouse. We assume the articles to be rather small so that up to ten different articles can be stored at one pick position at the same time. Furthermore, we presume that a sufficiently large quantity of each article is available at a pick position so that each line of the pick list can be collected from one pick position (the unit-demand case, see Section 2.1.2).

The number of different articles (SKUs) stored in the warehouse is fixed to $n = 1000$ for all instances to ensure comparability. Hence, there are $n_A = 200$ class-A, $n_B = 300$ class-B, and $n_C = 500$ class-C articles.

For the storage policy *uniformly distributed*, scattering is performed as follows. If the global scatter factor is α there must be $n\alpha$ different pairs (a, p) of article a and pick position p. More precisely, for a given scatter factor setting $(\alpha_A/\alpha_B/\alpha_C)$, we generate $n_A\alpha_A + n_B\alpha_B + n_C\alpha_C = n\alpha$ pairs in the following way: First, each article a is assigned to a randomly chosen pick position p (while making sure that no position contains more than 10 different articles). This guarantees that every article is available and stocked at least once.

Second, for each class $X \in \{A, B, C\}$, we generate $n_X(\alpha_X - 1)$ additional pairs (a, p) (excluding duplicates and again making sure that no position contains more

than 10 different articles).

Third, for class-based storage policies (*across-aisle*, *within-aisle*, *perimeter*, and *diagonal*), we use the same instance generation mechanism but extend the procedure by the following additional step: The warehouse is divided into three zones, one zone for each class A, B, and C depending on the storage policy and according to the sizes of the zones, see Section 2.5. Hence, zone A comprises 20%, zone B 30%, and zone C 50% of all pick positions, respectively. Next, the pairs (a, p) generated in the second step are randomly relocated to feasible positions p' where the position p' must belong to the class of article a. As a result, the new set of pairs (a, p') consistently stores all articles according to the given class-based storage policy.

Fourth and finally, the pick list must be created. The length of the pick list is fixed to 5, 15, or 50. Given the length, we first assign each order line with a probability of 80%, 15%, and 5% to class A, B, or C, respectively. Next, we randomly choose an article from the class (rejecting duplicate articles in the pick list).

Table 5.2 summarizes the parameters used for generating instances for the computational experiments. This includes the different scatter factor settings and lengths of pick lists. For each parameter combination, we generated 50 instances. In total, this gives $5 \cdot 3 \cdot 3 \cdot 50 = 2250$ instances (five storage policies, three scatter factor settings, and three lengths of pick lists) to be analyzed with six routing policies (see Section 2.2). The instances are online available at `https://logistik.bwl.uni-mainz.de/reserach/benchmarks/`.

2.6.2 Computational Performance

As noted in Section 2.4, solving the SPRP-SS is NP-hard, even when heuristic routing policies are used. However, all instances of the computational study can be solved within milliseconds. Table 2.3 shows the average and maximum computation times (in milliseconds) for the different routing policies and pick list lengths. Times include the construction of the IP model (which is negligible) and the actual solution time consumed by the MIP solver (`CPLEX` is restricted to use a single CPU thread only). One can see that more complex routing policies like `exact` or `composite` lead to longer computation times than fairly simple routing policies like `return` or `traversal`. The reason for this are the differently sized state spaces, see Section 2.2. While complex routing policies allow (almost) all states and many different actions within an aisle for the different states, the state space of a simple routing policy is rather small and sparse. The order size and the length of the pick list also influence the computation time (note that in the unit-demand case, order size equals pick list length; we will use the shorter term 'order size' in the following). In particular, if the considered routing policy allows many different

actions within an aisle for many different states, a larger order size can result in a tenfold increase in runtime. In contrast, for small and sparse state spaces, this effect is very small or disappears completely. Despite a maximum runtime of 634.91 milliseconds for an instance with a pick list length of 50 and routing policy `exact`, the computation times are very short overall. Therefore, we do not go into further algorithmic details and neglect a more in-depth analysis of the runtimes. Instead, we will focus on evaluating different routing and storage policy settings in the following.

		Average time			Max. time		
Order size		5	15	50	5	15	50
Routing policy	`exact`	6.60	15.57	72.78	48.64	227.38	634.91
	`composite`	3.27	5.47	11.38	26.53	112.33	160.16
	`largest gap`	1.54	2.20	7.93	14.43	20.57	141.76
	`midpoint`	1.40	1.78	4.03	5.86	16.97	75.66
	`return`	1.42	1.59	1.75	15.21	10.44	16.77
	`traversal`	2.01	1.83	1.29	16.01	40.92	7.87

Table 2.3: Average and maximum computation times in milliseconds for different routing policies and order sizes.

2.6.3 Evaluation of Combinations of Routing and Storage Policies

Next, we evaluate combinations of routing policies and storage policies for the SPRP-SS. To this end, each of the earlier introduced routing policies is combined with every storage policy. In addition, both the scatter factor settings ($\alpha_A/\alpha_B/\alpha_C$) and the order sizes are varied.

Table 2.4 provides aggregated results for each combination. (For the sake of readability, some routing and storage policies are abbreviated in the tables: *uniformly distributed* is denoted as *unif. distr.*, `largest gap` as `gap`, *across-aisle* as *across*, and *within-aisle* as *within*.) The upper part of the table shows the average tour lengths. A striking observation are the large differences in the tour lengths. While the combination of `return` and *uniformly distributed* leads to an average tour length of 351.31, the policies `exact` and *within-aisle* induce tours that are nearly 60% shorter with a mean length of 149.04. Even if the optimal routing policy `exact` is replaced by `composite`, the tour hardly gets any longer on average. On the contrary, for a given storage policy, the average tour lengths differ considerably for all heuristic routing policies. Similar observations can be made when

the routing policy is fixed and the storage policy is varied. Thus, we conclude that both the choice of the routing policy and the storage policy have a major impact on the tour length.

When comparing the routing policies in Table 2.4, it is noticeable that there are two policies each that behave very similarly. These are `composite` and `exact` as well as `midpoint` and `largest gap`. The lower part of the table shows that the deviation of `composite` (`midpoint`) from `exact` (`largest gap`) is less than 5% for every storage policy, except for the single combination with *perimeter* (with a deviation of nearly 12%). As the tour lengths hardly differ for these routing policies, we exclude `composite` and `midpoint` from further analyses. Routing policy `exact` is preferred over `composite` to be able to compare optimal rule-based and minimum-length tours. Besides, as already discussed in Section 2.2.6, the heuristic routing policy `composite` has rather difficult rules. Thus, the other routing policies are more attractive, because they are easier to remember and are expected to lead to fewer human errors. The rules for `largest gap` and `midpoint` only differ in the turning point within an aisle. Tours resulting from `largest gap` are always shorter or as long as the tour resulting from `midpoint` (this known result (Hall, 1993) for the non-scattered case directly transfers to the case of scattering articles). Therefore, we limit the further evaluation to `largest gap` and disregard `midpoint` in the following.

		Storage policy				
		unif. distr.	*within*	*across*	*perimeter*	*diagonal*
Routing policy	`exact`	233.27	149.04	163.31	165.68	162.20
	`composite`	238.82	151.61	164.30	185.46	163.99
	`largest gap`	256.91	165.93	197.24	174.61	186.83
	`midpoint`	268.50	169.86	198.80	176.36	189.54
	`return`	351.31	204.38	197.45	287.40	204.45
	`traversal`	290.00	181.35	284.22	249.73	227.52
Deviation	`composite` / `exact`	2.38%	1.72%	0.61%	11.93%	1.10%
	`midpoint` / `largest gap`	4.51%	2.37%	0.79%	1.00%	1.45%

Table 2.4: Average picker tour length for all combinations of routing policies and storage policies.

Table 2.5 shows detailed low-level results. For each combination of a routing and storage policy, average tour lengths are listed per scatter factor setting and order size. In addition, for each combination of routing policy, scatter factor setting and order size, the storage policies leading to the shortest average tour length (min) are given. Vice versa, for each storage policy, the best routing policy is given per setting (in this case, we exclude the routing policy `exact` from the min-grading as the optimal routing always generates the shortest average tour lengths; policy

`exact` is shaded in gray in Table 2.5). In this spirit, the routing policy `exact` is rather used as an optimal reference solution, not as one of the compared routing policies for the subsequent evaluations.

We first concentrate on the global findings, before we analyze each class-based storage policy separately concerning the different routing policies and the variation of the scatter factor setting and the order size. In both cases, Table 2.5 is the basis for the evaluation, and Table 2.6 provides complementing data summarized across all order sizes.

General observations can be summarized as follows:

- Storage policy *uniformly distributed* performs worse compared to the class-based storage policies: It generates the longest picker tours for nearly all settings with scatter factor setting (1/1/3) and (2/2/2), see Table 2.5. Only for (6/1/1), *uniformly distributed* can keep up and is in the midfield compared to the other storage policies. This can also be seen in Table 2.6, which summarizes the average tour lengths across all order sizes. These observations can be explained by the fact that an order consists of 80% A-articles, and as *uniformly distributed* does not take advantage of the classification of the articles, it is most advantageous to scatter only the articles that appear most frequently in an order. We disregard the storage policy *uniformly distributed* in the following analysis and concentrate on the class-based storage policies.
- Looking at the performance of the different scatter factor settings, it can be stated that, for every order size, the scattering of C-articles leads to the shortest picker tours. We indicate the best result per order size and scatter factor setting by using bold font in Table 2.5. Furthermore, each routing policy also generates the shortest picker tours for each order size, if scatter factor setting (1/1/3) is applied.
  Comparing the tours generated by the different routing policies per scatter factor setting and order size, it is noticeable that the scatter factor settings behave differently. For scatter factor setting (1/1/3), the tour lengths resulting from the different routing policies vary considerably. In contrast, if only the A-articles are scattered, the choice of the routing policy only has a minor influence on the tour length. While A-articles are stored at beneficial pick positions close to the depot, C-articles are located far from the depot. Thus, an A-article in the order list normally implies a shorter supplementary path segment in the picker tour than a C-article. And similarly, scattering A-articles usually saves less distance than scattering C-articles, since the choice of one or several alternative pick positions within zone A potentially leads to smaller savings than in zone C.
- Comparing the different order sizes, there are particularly poor and good

Scatter factor setting		*unif. distr.*	*perimeter*	*across*	*within*	*diagonal*	min
(1/1/3)	exact	155.20	139.08	70.60	89.12	72.52	*across*
	gap	167.36	141.08	139.44	90.24	114.92	*within*
	return	220.80	186.92	**70.60**	110.36	72.88	*across*
	traversal	214.56	158.76	200.88	90.40	127.16	*within*
	min	gap	gap	return	gap	return	
(2/2/2)	exact	130.56	127.84	81.52	87.64	95.48	*across*
	gap	141.16	138.84	130.76	104.76	126.44	*within*
	return	166.36	169.12	**83.04**	95.08	97.12	*across*
	traversal	171.24	166.08	144.20	110.68	141.44	*within*
	min	gap	gap	return	return	return	
(6/1/1)	exact	98.12	93.48	98.04	92.68	94.76	*within*
	gap	112.28	110.76	109.84	109.60	114.76	*within*
	return	106.16	100.60	108.72	**100.08**	100.64	*within*
	traversal	115.60	114.52	114.80	115.08	117.84	*perimeter*
	min	return	return	return	return	return	

(a) Order size 5.

Scatter factor setting		*unif. distr.*	*perimeter*	*across*	*within*	*diagonal*	min
(1/1/3)	exact	278.88	152.40	137.00	114.76	130.12	*within*
	gap	301.16	152.44	189.72	**122.76**	160.80	*within*
	return	447.40	266.56	137.80	156.92	134.56	*diagonal*
	traversal	373.24	241.12	385.12	131.24	212.76	*within*
	min	gap	gap	return	gap	return	
(2/2/2)	exact	214.08	164.32	159.12	152.60	153.68	*within*
	gap	230.76	170.96	192.68	**162.72**	178.60	*within*
	return	328.08	254.72	170.00	211.92	168.64	*diagonal*
	traversal	291.64	242.00	297.84	189.60	238.40	*within*
	min	gap	gap	return	gap	return	
(6/1/1)	exact	160.64	154.08	166.96	151.44	167.08	*within*
	gap	172.32	169.20	178.64	**165.00**	175.72	*within*
	return	215.08	214.64	221.24	198.04	228.68	*within*
	traversal	206.84	205.04	208.04	199.80	208.08	*within*
	min	gap	gap	gap	gap	gap	

(b) Order size 15.

Scatter factor setting		*unif. distr.*	*perimeter*	*across*	*within*	*diagonal*	min
(1/1/3)	exact	434.24	181.80	210.68	162.88	201.00	*within*
	gap	502.32	**181.80**	248.32	187.12	218.72	*perimeter*
	return	706.40	555.04	220.56	241.64	231.24	*across*
	traversal	462.88	440.08	464.00	197.16	292.24	*within*
	min	traversal	gap	return	gap	gap	
(2/2/2)	exact	359.48	232.68	257.28	240.60	263.80	*perimeter*
	gap	396.68	**233.52**	280.60	270.76	290.20	*perimeter*
	return	562.68	452.56	298.72	354.12	315.60	*across*
	traversal	427.24	369.04	397.80	283.24	353.84	*within*
	min	gap	gap	gap	gap	gap	
(6/1/1)	exact	268.20	245.48	288.56	249.64	281.32	*perimeter*
	gap	288.12	**272.92**	305.16	280.40	301.28	*perimeter*
	return	408.80	386.44	466.36	371.28	490.72	*within*
	traversal	346.72	310.92	345.32	314.92	355.96	*perimeter*
	min	gap	gap	gap	gap	gap	

(c) Order size 50.

Table 2.5: Break down of average tour lengths on the level of scatter factor settings and order sizes.

routing policies. With an order size of 5, in eleven of 15 combinations, the routing policy `traversal` leads to the longest tours on average. However, for order size of 50, the situation changes and `return` generates the longest picker tours in most of the combinations. The reason is that `return` is more suitable for smaller order sizes, while `traversal` is advantageous for larger order sizes. If only a few pick positions need to be visited, it is shorter to enter the aisles only from the front cross-aisle. If many positions that are distributed across many aisles need to be visited, traversing each aisle leads to shorter picker tours than entering and leaving each aisle from the same end.
Looking at best-performing routing policies, the routing policy `return` generates the shortest picker tours on average for order size 5. As already justified, the smaller the order, the better results achieves `return`. For the larger order sizes 15 and 50, the routing policy `largest gap` leads to the shortest picker tours in 24 of 30 settings.

- As highlighted in Table 2.6, the routing policies behave rather homogenous regarding the different scatter factor settings. Over all order sizes, `largest gap` performs best (aside from the `exact` routing) for all three scatter factor settings. The average tour lengths of the other two routing policies `return` and `traversal` differ only slightly, especially for (1/1/3) and (2/2/2).
- Compared to the results of Petersen and Schmenner (1999) for classical warehouses (without scattered storage), the tour lengths vary over a wider range. In Petersen and Schmenner (1999, Table 6), the largest deviation from optimal routing is 50.5% for *perimeter* and `return`, while for the scatter factor setting (1/1/3) the corresponding value is significantly larger at 113%. Such a greater difference can be observed for most of the routing and storage policy combinations. Consequently, larger cost savings are possible with scattered storage by choosing a suitable combination of the routing and storage policy compared to classical warehouses. However, note that the warehouse layout of Petersen and Schmenner differs from ours. Even though the number of aisles coincides, the warehouse dimension is approximately $2:1$ in their approach, in contrast to $3:4$ in ours. Nevertheless, both evaluations find the combination *within-aisle* and `largest gap` the best performing one.

2.6.3.1 Storage Policy Perimeter

Given the storage policy *perimeter*, `largest gap` always provides the best results except for one setting that combines (1/1/6) with the order size of 5 (see Table 2.5). The picker tours generated from the policies `largest gap` and *perimeter* are only about 6% longer than the optimal tours resulting from `exact` and *perime-*

Scatter factor setting			*unif. distr.*	*perimeter*	*across*	*within*	*diagonal*	Subtotal
	(1/1/3)	`exact`	289.44	157.76	139.43	122.25	134.55	168.69
		`gap`	323.61	158.44	192.49	**133.37**	164.81	**194.55**
		`return`	458.20	336.17	142.99	169.64	146.23	250.65
		`traversal`	350.23	279.99	350.00	139.60	210.72	266.11
		Subtotal	355.37	233.09	206.23	141.22	164.08	
	(2/2/2)	`exact`	234.71	174.95	165.97	160.28	170.99	181.38
		`gap`	256.20	181.11	201.35	**179.41**	198.41	**203.30**
		`return`	352.37	292.13	183.92	220.37	193.79	248.52
		`traversal`	296.71	259.04	279.95	194.51	244.56	254.95
		Subtotal	285.00	226.81	207.80	188.64	201.94	
	(6/1/1)	`exact`	175.65	164.35	184.52	164.59	181.05	174.03
		`gap`	190.91	**184.29**	197.88	185.00	197.25	**191.07**
		`return`	243.35	233.89	265.44	223.13	273.35	247.83
		`traversal`	223.05	210.16	222.72	209.93	227.29	218.63
		Subtotal	208.24	198.17	217.64	195.66	219.74	

Table 2.6: Average picker tour lengths summarized across all order sizes.

ter. This is because the rules of `largest gap` are harmonized with the definition of the zones of *perimeter*. Since A-articles are stored around the perimeter and an order consists of 80% A-articles, the procedure of `largest gap` is advantageous where the outer positions of each aisle are cheap to visit. Combined with *perimeter*, the routing policies `return` and `traversal` perform significantly worse than `largest gap`. However, an exception is the already mentioned setting with (1/1/6) and an order size of 5. Here the routing policy `return` leads to better results than `largest gap`. Because only very few articles need to be collected, the typical transition of `largest gap` (where an aisle is entered from both sides and the picker traverses both cross-aisles) enforces an unfavorable selection of pick positions to visit. In contrast, applying the routing policy `return` instructs the picker to traverse only one cross-aisle while entering the necessary aisles to collect the demanded articles. This leads to shorter picker tours for small order sizes. The situation is exactly the opposite with the order size of 50. In this case, many articles from several pick positions need to be collected so that, in the worst case, a typical transition of `return` causes entire aisles to be traversed almost twice, as an aisle must be entered and exited from the same side.

In addition, it can be stated that *perimeter* is particularly suitable for large order sizes. Table 2.5 shows that, for the order size of 50, *perimeter* is the best choice for half of the settings.

2.6.3.2 Storage Policy Across-Aisle

The storage policy *across-aisle* performs best in combination with the routing policy `return`, as can be seen from Table 2.5. Particularly, this affects orders with few C-articles (i.e., small orders) or orders with scattered C-articles where advantageous pick positions are chosen to collect C-articles. For the large order size of 50 and C-articles not being scattered exclusively, it is beneficial to use routing policy `largest gap`. Here, the collection of additional C-articles can easily be appended to the tour. However, for routing policy `return`, the visit of numerous pick positions in zone C induces additional costly path sections, since most aisles must be traversed twice to collect an article of class C. Policy `return` can hardly gain any advantages from the scattering of the A-articles, since only little distance can be saved by selecting favorable positions of A-articles. As a result, `largest gap` generates shorter tours on average.

In seven of nine settings, the routing policy `traversal` is the worst policy for *across-aisle* because each aisle entered must be traversed completely while many aisles need to be entered to collect all the demanded A-articles. For the order size of 5, storage policy *across-aisle* delivers the best results among all storage policies if combined with `exact` routing.

2.6.3.3 Storage Policy Within-Aisle

Comparing all 36 combinations of Table 2.5, *within-aisle* is the storage policy performing best. For nearly 60% of the settings (21 settings), the application of *within-aisle* leads to the shortest picker tours on average. Compared to the other storage policies, *within-aisle* is less sensitive to the rules of the different routing policies. The average tour lengths of *within-aisle* combined with all the routing policies deviate significantly less among each other than the tour lengths produced by any other storage policy in combination with the different routing policies. Nevertheless, `largest gap` is the best routing policy to be combined with *within-aisle*. In seven of nine settings, this combination generates the best results.

For order sizes 15 and 50, the shortest (policy `exact`) picker tours can be obtained with *within-aisle*. In both cases, the best results can be achieved if the C-articles are scattered. The reason is that pick positions for the demanded C-articles can be selected in aisles close to the depot and the picker does not even have to pass through some more distant aisles of the warehouse. For the same reason, *within-aisle* works best, independent of the order size, if the scatter factor setting (1/1/3) is used. Overall, storage policy *within-aisle* leads to the shortest picker tours compared to combinations in which another storage policies is applied.

2.6.3.4 Storage Policy Diagonal

The storage policy *diagonal* is a mixture of *across-aisle* and *within-aisle*. Depending on the warehouse layout, it resembles one of the two storage policies more than the other: If the aisles of the warehouse are rather short or the pick positions are arranged narrowly and without spacing, then *diagonal* has strong similarities to *within-aisle*. Otherwise, the warehouse tends to have long aisles and/or large distances between two consecutive aisles so that *diagonal* tends to resemble the storage policy *across-aisle*.

The warehouse layout that we use for the computational study (10 aisles and 40 pick positions per aisle) effectuates that *diagonal* tends to resemble the storage policy *across-aisle*. In seven of the ten aisles, pick positions of zone A can be found. This is also reflected in the results, where the best and worst routing policies for *diagonal* and *across-aisle* are identical except for a single setting (order size of 50 and storage factors (1/1/3)).

2.7 Conclusions and Outlook

This work has focused on the single picker routing problem with scattered storage (SPRP-SS) in combination with standard routing and storage policies. Routing policies restrict possible picker tours to those that obey policy-specific rules, which should preferably allow a fast tour computation and be easy to memorize for the picker. On the theoretical side, we have proven in Section 2.4 that even under these routing policies, the SPRP-SS, i.e., the problem of selecting pick positions in combination with the determination of a shortest picker tour, remains an NP-hard problem.

On the practical side, we have developed an exact solution algorithm for these variants of the SPRP-SS that is both effective and relatively simple to implement. It can be summarized as follows: First, we construct the state space of the DP for the non-scattered SPRP. In Section 2.2, we have derived and detailed these state spaces as modified and restricted versions of the well-known DP of Ratliff and Rosenthal (1983). Second, we introduce into the DP additional aisle actions (in particular for `top`, `bottom`, and `gap`) that become possible in the scattered storage case (see Section 2.3). Third, we set up an IP that models the resulting DP as an origin-destination shortest-path problem. We add demand-covering constraints to ensure that a sufficient number of units of each requested article is collected. Fourth and finally, a standard MIP solver computes an optimal solution to the IP.

The computational evaluation has shown that the main drivers for the practical difficulty of the SPRP-SS are firstly the number of different articles to be picked, secondly the routing policy (the size of the respective state space). The average

computation times for pick lists of lengths 5, 15, and 50 are below 49, 228, and 635 milliseconds, respectively, for all routing policies and for a single-block warehouse with ten parallel aisles of 40 cells each. Hence, computation times become irrelevant when the SPRP-SS is considered as a stand-alone problem.

The new and fast exact solution approach has enabled us to precisely analyze and compare different combinations of routing and storage policies. Note that the storage policy, i.e., at which locations in the warehouse articles are stored, determines how a typical instance of the SPRP and SPRP-SS is composed. It does not require a storage policy-specific adaptation of the above solution approach. Therefore, the comparison of all combinations of routing and storage policies focuses on the average tour lengths. The main findings for instances of a standard one-block parallel-aisle warehouse are:

- Neglecting the routing policy `exact`, the combination of the `largest gap` routing policy and the *within-aisle* storage policy lead to the best overall results.
- For class-based storage policies, scattering of C-articles is most advantageous. On the contrary, for uniformly distributed articles, scattering of A-articles performs best.
- In general, if only C-articles are scattered, different routing policies lead to picker tours with remarkable differences in tour length. However, if only A-articles are scattered, the impact of the different routing policies is substantially less pronounced and leads to only small differences among the tour lengths.

It must be stressed that the presented savings and performance ratios depend on the instance data, which includes the composition of an average pick list as well as the concrete warehouse geometry (number of aisles, number of cells per aisle, distances between aisles and cells and I/O point(s), respectively). Even more, for different warehouse layouts and I/O point configurations (see Table 2.1), we probably get different results than those obtained in Section 2.6.3 per combination of routing and storage policy. However, we expect similar general findings as the three highlighted above. In this sense, we see our work as a fundamental step to introduce a methodology that is widely applicable to perform experiments and quantify savings in average picker tour lengths for a particular warehousing setting.

Accordingly, we see several avenues for interesting further research: Alternative warehouse settings other than the single-block parallel-aisle warehouse with a unique I/O point should be analyzed. Since DP algorithms are known for most of them (see Table 2.1), a solution approach adapted from the one presented here should also be tested and similar combinations of routing and storage policies should be analyzed. One complication is that, to our knowledge, class-based storage policies have not yet been defined for two-block, multi-block, and other ware-

house layouts. Even for warehouses where the I/O point is not in one of the corners and for warehouses with multiple drop-off points, the definition of the storage policies is unclear. This necessary specification should be done first. In addition, the storage policies could be refined and alternative storage policies could be empirically analyzed using the methodology proposed in this paper. For example, articles that are ordered together more often than other articles could be assigned to pick positions that are closer together. For class-based storage policies, one may vary the scatter factors within class C. An analysis of how article-specific scatter factors should be chosen is worth further investigation when class-based storage policies are replaced by full-turnover storage. Finally, if SPRP-SS appears as a subproblem, e.g., in the joint order batching and picker routing problem (Wahlen and Gschwind, 2023), the presented IP-based solution algorithm might become too slow. Therefore, finding an even more effective solution algorithm for the SPRP-SS remains an algorithmic challenge.

Bibliography

Behnisch, P., Glock, C. H., Grosse, E. H., and Ries, J. M. (2017). Auf dem Weg zum Warehouse 4.0? – Zum aktuellen Stand der Automatisierung in der Lagerhaltung. Technical Report 84808, Institue for Business Studies (BWL), Department of Business Administration, Economics and Law, Darmstadt Technical University. (in German).

Bock, S. and Boysen, N. (2023). Routing replenishment workers: The prize collecting traveling salesman problem in scattered storage warehouses. *INFORMS Journal on Computing*, **36**(1), 3–20.

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.

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

Cormier, G. and Gunn, E. A. (1992). A review of warehouse models. *European Journal of Operational Research*, **58**(1), 3–13.

Cornuéjols, G., Fonlupt, J., and Naddef, D. (1985). The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, **33**(1), 1–27.

Daniels, R. L., Rummel, J. L., and Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, **105**(1), 1–17.

de Koster, R. and van der Poort, E. (1998). Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. *IIE Transactions*, **30**(5), 469–480.

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.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

Gibson, D. R. and Sharp, G. P. (1992). Order batching procedures. *European Journal of Operational Research*, **58**(1), 57–67.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 419–835.

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.

Gue, K. R. and Meller, R. D. (2009). Aisle configurations for unit-load warehouses. *IIE Transactions*, **41**(3), 171–182.

Gutin, G. and Punnen, A., editors (2002). *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, Dordrecht.

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

Hausman, W. H., Schwarz, L. B., and Graves, S. C. (1976). Optimal storage assignment in automatic warehouse systems. *Management Science*, **22**, 629–638.

Henn, S., Koch, S., Gerking, H., and Wäscher, G. (2013). A U-shaped layout for manual order-picking systems. *Logistics Research*, **6**(4), 245–261.

Heskett, J. L. (1963). Cube-per-order index–a key to warehouse stock location. *Transportation and Distribution Management*, **3**(1), 27–31.

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. `https://logistik.bwl.uni-mainz.de/files/2022/04/LM-2022-03.pdf`.

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*, **36**(6), 1417–1435.

Khan, I., Maurer, O., Pätzold, J., Pszona, P., and Salchow, J.-D. (2024). Joint order selection, allocation, batching and picking for large scale warehouses. Technical report, arXiv 2401.04563. https://doi.org/10.48550/arXiv.2401.04563.

Löffler, M., Boysen, N., and Schneider, M. (2022). Picker routing in AGV-assisted order picking systems. *INFORMS Journal on Computing*, **34**(1), 440–462.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2019). Optimal order picker routing in the chevron warehouse. *IISE Transactions*, **52**(6), 665–687.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2020a). Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour. *International Journal of Production Research*, **58**(17), 5337–5358.

Masae, M., Glock, C. H., and Grosse, E. H. (2020b). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, **224**, 107564.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2021). A method for efficiently routing order pickers in the leaf warehouse. *International Journal of Production Economics*, **234**, 108069.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problems. *Journal of Association for Computing Machinery*, **7**, 326–329.

Öztürkoğlu, Ö., Gue, K. R., and Meller, R. D. (2012). Optimal unit-load warehouse designs for single-command operations. *IIE Transactions*, **44**(6), 459–475.

Öztürkoğlu, Ö. and Hoser, D. (2019). A discrete cross aisle design model for order-picking warehouses. *European Journal of Operational Research*, **275**(2), 411–430.

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

Park, Y. H. and Webster, D. B. (1989). Design of class-based storage racks for minimizing travel time in a three-dimensional storage system. *International Journal of Production Research*, **27**(9), 1589–1601.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. In *Decision Sciences Institute Proceedings*, volume 3, pages 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 Schmenner, R. W. (1999). An evaluation of routing and volume-based storage policies in an order picking operation. *Decision Sciences*, **30**(2), 481–501.

Petersen, C. G., Aase, G. R., and Heiser, D. R. (2004). Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, **34**(7), 534–544.

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, RSM Erasmus University, 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.

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.

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., and Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, **305**(1), 222–239.

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.

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

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, **95**, 139–150.

Weidinger, F. and Boysen, N. (2018). Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, **52**(6), 1412–1427.

Weidinger, F., Boysen, N., and Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, **274**(2), 501–515.

Chapter 3

A Linear-Size Model for the Single Picker Routing Problem with Scattered Storage

Laura Lüke, André Hessenius, Stefan Irnich

Abstract

We present a new approach of solving the single picker routing problem with scattered storage (SPRP-SS), which is a fundamental problem in modern warehouse operations management. The SPRP-SS assumes that *stock keeping units* (SKUs) of an article are stored at possibly many locations. An effective integer programming based approach relies on extending the state space of Ratliff and Rosenthal's dynamic program for the basic single picker routing problem to accommodate the SPRP-SS. As a result, the mixed integer linear programming (MIP) formulation has a quadratic number of variables. We propose two distinct modifications of the extended state space to retain the linearity of the models. Linearity is achieved by replacing the quadratically growing parallel edges of the extended state space by linear-size subnetworks. These replacements lead to different state spaces and herewith different MIP formulations, for which we analyze theoretical properties such as their size and strength of the linear-programming relaxations. We compare the new formulations with the state of the art using a collection of 800 SPRP-SS instances. The results show that the new formulations are more than competitive providing integer optimal solutions of realistic and even large-scale instances in less than two seconds on average. The second formulation outperforms the currently best performing approach regarding the computational speed: For the largest instances with 200 articles to be collected, average speedups reach the factors of 3.18 and 4.87 for general and unit demand, respectively.

3.1 Introduction

We consider order picking operations in warehouses where pickers travel through the warehouse to collect demanded articles from the storage locations (picker-to-parts). Order picking accounts for more than half of the warehouse operating costs (Bartholdi and Hackman, 2019) and therefore represents a good opportunity for expense optimization. We address the underlying single tour routing problem, which is known as the *single picker routing problem* (SPRP) or *order picking problem*. The most basic version of the SPRP considers a rectangular single-block parallel-aisle warehouse with a single depot, which is the start and end point of each tour. The seminal work of Ratliff and Rosenthal (1983) shows that a minimum-length picker tour can be computed with dynamic programming in linear time. More precisely, the overall computational effort is linear in the number of aisles and the number of storage locations (=pick positions) to visit, as shown by Heßler and Irnich (2022b).

Several variants of the SPRP extend the basic problem and require the computation of picker tours for different warehouse layouts (two-block (Roodbergen and de Koster, 2001a), multi-block (Pansart *et al.*, 2018), fishbone (Çelk and Süral, 2014), butterfly (Öztürkoğlu *et al.*, 2012), etc.) and characteristics of the start and drop-off point(s) of a picker tour (Masae *et al.*, 2020b). In addition, simpler variants of the SPRP result from routing policies such as traversal (a.k.a. S-shape), midpoint, largest gap (Hall, 1993), return, and composite (Petersen, 1997). Routing policies address the practical issue that human pickers may not be able to execute all types of optimal tours, which can be complicated, counterintuitive, and difficult to memorize. Instead, the pickers must execute a tour that is defined by a few simple rules. A more detailed overview of SPRP variants and pointers to the literature can be found in (Heßler and Irnich, 2024, Table 1).

In this paper, we address the *SPRP with scattered storage* (SPRP-SS), which is another variant of SPRP. The term scattered storage describes the storage strategy of placing the *stock keeping units* (SKUs) of one or more articles not only at a single but possibly several pick positions in the warehouse (this is also our definition: a SKU is one item/a single unit of an article, where identical SKUs can be stored at different locations/positions). The reasoning behind scattering is that SKUs of a demanded article can be picked up quickly no matter where the picker is (Weidinger, 2018). Scattered storage is the predominant storage strategy in modern e-commerce warehouses of companies like Amazon or Zalando (Weidinger, 2018; Weidinger *et al.*, 2019; Boysen *et al.*, 2019; Khan *et al.*, 2024).

In contrast to the long history of the basic SPRP, the SPRP-SS has only received increased attention since the middle of the last decade. First addressed by Singh and van Oudheusden (1997) and Daniels *et al.* (1998), up to this day, all competitive exact solution approaches are *mixed-integer programming* (MIP) based, except

for the Benders approach of Haouassi *et al.* (2025). Recently, Goeke and Schneider (2021) proposed an effective MIP formulation that solves instances of the SPRP-SS of realistic size to proven optimality. This approach is only outperformed by another MIP-based solution approach of Heßler and Irnich (2024). Their idea is as follows: In the state space of the dynamic-programming (DP) approach of Ratliff and Rosenthal (1983) for the basic SPRP, every feasible picker tour is a path and vice versa. The first property also holds for the SPRP-SS. With an extension in the underlying state space, the authors use a shortest path formulation to solve the problem. Herein, the extension of the state space requires the addition of new edges. The number of edges grows quadratically with the number of relevant pick positions per aisle.

There exist some other approaches for the SPRP-SS: Weidinger (2018) compared a decomposition procedure (select the pick positions by different priority rules and use the algorithm of Ratliff and Rosenthal to determine a picker tour) with the MIP of Daniels *et al.* (1998) complemented with MTZ-based subtour-elimination constraints (Miller *et al.*, 1960). A rather involved MIP-based approach has been presented by Su *et al.* (2023) for multi-block parallel-aisle warehouses. Unfortunately, this approach has not been compared with any other approach, e.g., using the fact that the SPRP-SS with unit demand can be modeled and solved as a *generalized TSP* (GTSP) for any warehouse layout (for the definition of *unit demand* and *general demand*, see Section 3.2). Wildt *et al.* (2025) use exactly this fact, which makes their approach generally applicable, but ignores that warehouses often have a well-structured layout that can be exploited algorithmically. Accordingly, they use a series of transformations (SPRP-SS to GTSP, GTSP to clustered TSP, clustered TSP to asymmetric TSP, asymmetric to symmetric TSP) to finally solve them as ordinary TSP instances, either with a branch-and-cut-based TSP solver like Concorde (Applegate *et al.*, 2003) or with the Lin–Kernighan–Helsgaun (LKH) heuristic (Helsgaun, 2000). The equivalence to the GTSP was also discussed by Heßler and Irnich (2024) who compare a re-implementation of the branch-and-cut algorithm of Fischetti *et al.* (2002) with their approach showing that for small pick lists the GTSP-based solution can be competitive. Note that for general demand the equivalence to the GTSP is no longer given, so that Wildt *et al.* (2025) proposed heuristic transformations for this case.

3.1.1 Contributions

We introduce two new MIP formulations for the SPRP-SS which are linear in the number of aisles and in the number of pick positions. We describe both formulations now:

EG: The first formulation is formally identical to the network-flow model of Heßler and Irnich (2024), but uses a new underlying state space which replaces

parallel edges (a quadratic number of gap actions is possible) by a linear-size subnetwork. Gap actions model the movement of the picker entering an aisle from both cross-aisles to collect items. We further specify actions in Section 3.3.1. Since the subnetwork allows to expand gaps, this formulation is called *enlarged gaps* (EG) formulation.

RG: The second formulation builds a different kind of linear-size subnetwork in which gaps can only be reduced. Accordingly, the second formulation is called the *reduced gaps* (RG) formulation. In case of general demand, the network-flow model of Heßler and Irnich (2024) must be modified resulting in a slightly larger model with auxiliary variables and additional constraints. However, these model extensions keep the formulation linear.

We compare the two formulations to each other and against the state-of-the-art formulation of Heßler and Irnich (2024), in the following referred to as formulation HI. The theoretical and computational analyses provide the following findings:

- Both formulations EG and RG have a generally weaker linear-programming (LP) relaxation than the formulation HI. On the large SPRP-SS instance set used in the computational studies, formulation RG is almost as strong as formulation HI, while formulation EG is weaker.
- Both formulations EG and RG give significantly smaller MIP models than formulation HI. Formulation RG reduces, on average, more variables/edges (between 48 and 86 percent) than formulation EG (between 34 and 81 percent) in comparison to the quadratically sized networks used in formulation HI.
- Compared to the state of the art, formulation HI, the new formulation RG on average accelerates the MIP solution by a factor of 1.89 for general demand and by a factor of 2.42 for unit demand for the SPRP-SS instances in the testbed.

3.1.2 Structure

The remainder of this paper is structured as follows: In Section 3.2, we formally introduce the SPRP-SS. We summarize the relevant solution approaches in Section 3.3 which includes the DP of Ratliff and Rosenthal for the SPRP, the extension of the state space of Ratliff and Rosenthal to incorporate aisle traversal options needed for the SPRP-SS, and the network-flow model of Heßler and Irnich, which is finally used to solve SPRP-SS instances. The subnetworks for formulations EG and RG are presented in Section 3.4. Computational analyses and their results are discussed in Section 3.5. We close the paper by drawing final conclusions in Section 3.6. The Appendix lists the multiple acronyms and symbols used throughout this paper.

3.2 The Single Picker Routing Problem with Scattered Storage

In the SPRP-SS, the set S denotes the different *articles*, where each article $s \in S$ is stored at (possibly several) pick positions $p \in P_s$. The set $P = \bigcup_{s \in S} P_s$ describes all relevant pick positions. At position $p \in P_s$, $b_{sp} \geq 1$ units of article $s \in S$ are available for for collection. The SKUs are stored at the pick positions along both sides of the picking aisles in a single-block parallel-aisle warehouse. Note that the terms SKU and article describe the same object. An article is what is requested by a customer and a SKU refers to a unit of an article stored in the warehouse. The goal of the SPRP-SS is then to find a minimum-length picker tour that starts and ends at the given I/O point 0 and visits a subset of the pick positions such that the given demand q_s for each article $s \in S$ is satisfied, i.e., can be collected from the positions visited. Daniels *et al.* (1998) already stated that apart from the *general-demand case* with $q_s > 1$ for at least some articles $s \in S$, the SPRP-SS can also emerge in the so-called *unit-demand* case. In this case, the demanded number is $q_s = 1$ for all articles $s \in S$. The SPRP-SS with unit demand can be modeled as a *generalized traveling salesman problem* (GTSP, Fischetti *et al.*, 2002) with the pairs in $\{(s, p) \in S \times P : b_{sp} = 1\}$ as the cities and the sets $C_s = \{(s, p) : p \in P_s\}$ for $s \in S$ as the clusters (Daniels *et al.*, 1998). The unit-demand case is equivalent to the situation where each pick position b_{sp} holds sufficient supply to cover the whole demand q_s of the respective article $s \in S$.

The SPRP-SS is an NP-hard problem (Weidinger, 2018). This statement remains true when picker tours are restricted to routing policies such as traversal, return, midpoint, largest gap, and composite (Lüke *et al.*, 2024). Approaches for the exact solution and optimal routing have been proposed by Weidinger (2018), Goeke and Schneider (2021), and Heßler and Irnich (2024). The latter approach is the best-performing exact solution algorithm to date. Since our goal is to improve on the model proposed by Heßler and Irnich, we will present their formulation and approach in the following.

3.3 Exact Solution of the SPRP and SPRP-SS

We start with the description of the DP by Ratliff and Rosenthal (1983), explain how the state space of the DP was extended by Heßler and Irnich (2024) to include the option of not necessarily having to visit all pick positions, and present their binary formulation of the SPRP-SS.

3.3.1 Basic State Space of Ratliff and Rosenthal

We briefly introduce the ideas and assumptions of Ratliff and Rosenthal (1983) that allow to formulate the SPRP and solve it by DP in linear time. They consider a single-block parallel-aisle warehouse. Each aisle is uniformly divided into a number of pick positions of identical size. Each demanded article is stored at a pick position in the warehouse resulting in a set of pick positions P that the picker necessarily has to visit. Picking from either the right-hand side or the left-hand side (or both) is not distinguished, so that the side of the aisle is irrelevant for computing the picker tour length. Therefore, different SKUs located at the same pick position can be seen as a single picking request.

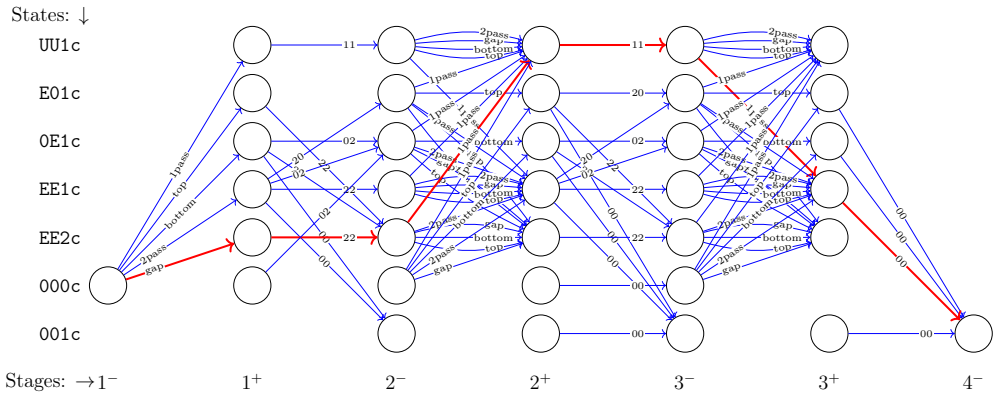
Formally, let $J = \{1, 2, \ldots, m\}$ denote the aisles in a single-block rectangular warehouse with parallel aisles. A picker tour can be constructed by alternately deciding how the picker traverses an aisle and how they process from aisle j to aisle $j + 1$. Moving through an aisle $j \in J$ results in the transition from stage j^- to j^+ while performing a *cross-aisle action* from an aisle j to $j + 1$ leads to the transition from stage j^+ to $(j + 1)^-$. The picker tour ends in a finite state at stage $(m+1)^-$, so that the state space of the DP has $2m+1$ stages. To model the states and transitions of the DP, Ratliff and Rosenthal introduce so-called *partial tour subgraphs* (PTSs). For an aisle $j \in J$, a PTS represents the parts of the picker tour from aisle 1 to aisle j and specifies the vertex parity of the vertices a_j and b_j located at the back and front of each aisle $j \in J$, see Figure 3.1b. The vertex parities in combination with the number of connected components of the PTS characterize the states of the DP. Ratliff and Rosenthal show that only seven states are relevant for optimal picker tours. These states can be denoted as

$$\mathscr{S} = \{\texttt{UU1c}, \texttt{0E1c}, \texttt{E01c}, \texttt{EE1c}, \texttt{EE2c}, \texttt{000c}, \texttt{001c}\}.$$

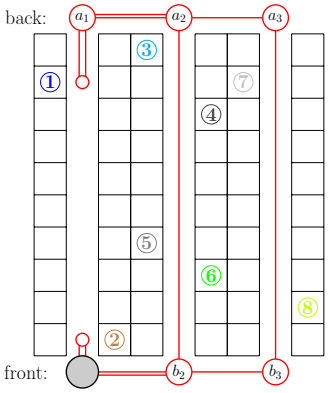
Here, the first (second) symbol indicates the parity of the vertex a_j (b_j) with `0`, `U`, and `E` standing for not reached, odd (i.e., uneven) degree, and even degree, respectively. The number of connected components is shown by the last two symbols: The state `0c` indicates an empty PTS, `1c` one with a single connected component, and `2c` a subgraph with two connected components. Figure 3.1 shows an optimal picker tour with its corresponding PTS and state space (V, E). Within an aisle, six different *aisle actions* can be performed, namely

$$E_j^{aisle} = \{\texttt{1pass}, \texttt{2pass}, \texttt{top}, \texttt{bottom}, \texttt{gap}, \texttt{void}\}. \tag{3.1}$$

Action `1pass` (`2pass`) describes a single (double) traversal through the aisle (either direction), `top` (`bottom`) stands for a traversal from the back (front) cross-aisle to the lowest (highest) relevant pick position and back, and `gap` for entering the aisle from both sides leaving a maximum length gap in the middle while visiting all pick



(a) State space (V, E).



(b) PTS corresponding to the optimal picker tour.

Figure 3.1: State space and PTS for a warehouse with aisles $J = \{1, 2, 3\}$ and articles $S = \{1, 2, \ldots, 8\}$ (unit demand). The highlighted path (red/bold) in the state space represents the optimal picker tour.

positions. When having an empty aisle, `void` represents no traversal through the aisle.

Regarding the *cross-aisle action* from aisle j^+ to $(j+1)^-$, there are five different options to consider:

$$E_j^{cross} = \{\texttt{00}, \texttt{11}, \texttt{20}, \texttt{02}, \texttt{22}\}.$$

The first (second) digit gives the number of traversals of the back (front) cross-aisle. Note that every o-d-path in the state graph represents a feasible solution for the SPRP, where the origin and destination are the states $o = \texttt{000c}$ at stage 0^- and $d = \texttt{001c}$ at stage $(m+1)^- = 4^-$, respectively. Moreover, each edge $e \in E_j^{cross} \cup E_j^{aisle}$ is associated with a cost describing the length of the component

added to the tour when performing the corresponding action. Heßler and Irnich (2022b) have shown that the calculation of these costs can be done with an effort of $\mathcal{O}(m+n)$ with $n = |P|$ while solving the DP also takes linear effort with the same complexity (Ratliff and Rosenthal, 1983).

3.3.2 Scattered Storage and the Extended State Space

For the SPRP-SS, some additional notation is required, since each article may now be stored at several locations instead of having one unique pick position. Recall that q_s denotes the quantity of article $s \in S$ to be collected and b_{sp} denotes the supply (i.e., number of SKUs) of $s \in S$ at pick position $p \in P$. Furthermore, let P_e denote the set of positions visited by an aisle traversal $e \in E_j^{aisle}$ in aisle $j \in J$. Thus, $b_{se} = \sum_{p \in P_e} b_{sp}$ is the quantity of article $s \in S$ that can be collected when traversing the aisle via e. Edges e with a positive supply of article $s \in S$ are denoted by E_s. For the sake of clarity, we distinguish between three types of articles:

(O) articles available in several aisles in the warehouse,
(UA) articles $s \in S$ available in a *unique aisle*, but with $|P_s| > 1$, and
(UP) articles that are only available at one *unique position* in the warehouse.

Note that looking at an arbitrary position in an aisle, different SKUs can be placed on the right-hand side, left-hand side or, when stored in shelves, even vertically over one another. Still, different SKUs located at one position in any of the described manners refer to the same pick position $p \in P$. Therefore, the number

$$n = \left| \bigcup_{s \in S} P_s \right| = |\{p \in P : \exists s \in S \text{ with } b_{sp} > 0\}| \tag{3.2}$$

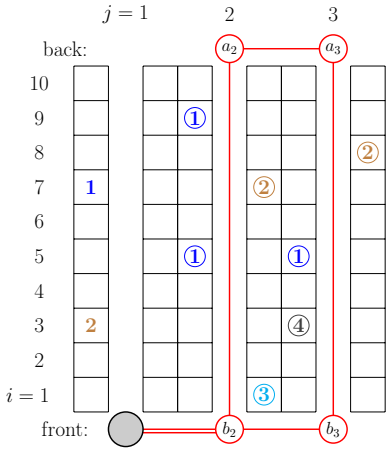
of relevant positions can be smaller than the number of pairs $(s,p) \in S \times P$ with a positive supply $b_{sp} > 0$.

To incorporate the flexibility of not necessarily having to traverse all pick positions with demanded articles in the warehouse, an extension of the state space of Section 3.3.1 is needed. Specifically, additional *aisle actions* of type `top`, `bottom`, `void`, and `gap` (as defined after eq. (3.1)) must be considered, while all other *aisle actions* and *cross-aisle actions* remain identical. These actions are the aisle traversals $e \in E_j^{aisle}$ referring to aisle $j \in J$ leaving out the positions $P_j' \subset P$ (in aisle j), which are feasible if and only if

$$\sum_{p \in P_s \setminus P_j'} b_{sp} \geq q_s \qquad \forall s \in S \tag{3.3}$$

holds. Further details are provided in (Heßler and Irnich, 2024, Sect. 2.3).

Example 1. *Figure 3.2a shows an instance of the SPRP-SS with three aisles* $J = \{1, 2, 3\}$ *and four articles* $S = \{1, 2, 3, 4\}$. *For simplicity, we assume unit supply, i.e.,* $b_{sp} = 1$ *for all* $s \in S, p \in P_s$. *Moreover, we assume demands* $q_1 = 3, q_2 = 2$, *and* $q_3 = q_4 = 1$. *The resulting additional aisle actions are listed in Figure 3.2b.*



(a) Warehouse and optimal picker tour; pick operations are encircled.

Aisle	type of	additional aisle actions
$j = 1$	`top`(i)	cell $i = 7$
	`bottom`(i)	cell $i = 3$
	`void`	
$j = 2$	`bottom`(i)	cell $i \in \{5, 7\}$
	`gap`(h, i)	cells $(h, i) \in \{(1, 7), (1, 9), (5, 9)\}$
$j = 3$	`bottom`(i)	cell $i \in \{3, 5\}$
	`gap`(h, i)	cells $(h, i) = (3, 8)$

(b) Additional aisle actions compared to non-scattered storage.

Figure 3.2: An instance of the SPRP-SS. The pick list contains four articles $S = \{1, 2, 3, 4\}$ with unit supply and demands $q_1 = 3, q_2 = 2$, and $q_3 = q_4 = 1$.

For aisle $j = 1$, *three additional* aisle actions *are to be considered. Standing at the back (front), it is now possible to only traverse the aisle up to position* $i = 7$ *(*$i = 3$*), or skip the aisle entirely, since inequalities* (3.3) *holds for an aisle action leaving out the position where article 2 (article 1) or both are stored. These additional options yield in the aisle actions* `top`(7), `bottom`(3), *and* `void`. *For the second aisle, article 3 must be picked up, since it is only available in one aisle so that no* `void` action *can be performed here, while also preventing any additional* `top`(i) *traversal. Note that the additional* `bottom` *and* `gap` *options always traverse at least one pick position of article 1, since inequalities* (3.3) *would be violated otherwise. Therefore,* `bottom`(1) *is not allowed. For aisle* $j = 3$, *article 4 is unique so that* `void`, `top`(5), *and* `top`(8) *become infeasible. Furthermore, all traversals* `gap`(i, k) *can be disregarded if* i *and* k *are neighboring positions with a non-maximal gap.*

Asymptotically, the number of additional aisle actions in the extended state space is dominated by those of type `gap`(h, i). Indeed, if the majority of the pick positions are located within one or a few aisles, there can be $\mathcal{O}(n^2)$ aisle actions

of type $\texttt{gap}(h,i)$. Therefore, since the original state graph had $\mathcal{O}(m)$ edges, the total number of edges in the extended state space is bounded by $\mathcal{O}(m+n^2)$.

Due to the extension of the state space, an arbitrary o-d-path does not necessarily represent a feasible solution to the SPRP-SS. The following network-flow formulation HI is considering this fact.

3.3.3 Network-Flow Formulation for the SPRP-SS

Formulation HI uses the above extended state space (V,E) to model the SPRP-SS as an o-d-shortest-path problem with additional *demand-covering constraints* (DCCs). Variables $x_e \geq 0$ indicate the (unit) flow for each edge $e \in E = \bigcup_{j\in J}(E_j^{aisle} \cup E_j^{cross})$. Moreover, let c_e be the cost of edge $e \in E$, i.e., the length of the picker tour associated with the action described by e. This leads to the following binary model:

$$z_{\text{SPRP-SS}} = \min \sum_{e\in E} c_e x_e \tag{3.4a}$$

$$\text{subject to} \quad \sum_{e\in\delta^+(\sigma)} x_e - \sum_{e\in\delta^-(\sigma)} x_e = \begin{cases} +1, & \text{if } \sigma = o \\ -1, & \text{if } \sigma = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall \sigma \in V \tag{3.4b}$$

$$\sum_{e\in E_s} b_{se} x_e \geq q_s \qquad \forall s \in S \tag{3.4c}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{3.4d}$$

The objective (3.4a) minimizes the length of the picker tour. Flow conservation is modeled by constraints (3.4b), where $\delta^+(\sigma)$ and $\delta^-(\sigma)$ denote the set of edges leaving and entering vertex $\sigma \in V$, respectively. The DCCs (3.4c) ensure that enough SKUs are collected for each article $s \in S$. The domain of the flow variables is given by (3.4d).

Note that constraints (3.4c) can be strengthened by replacing b_{se} with $\min\{q_s, b_{se}\}$ for all $s \in S$ (we always do this). Moreover, for each article $s \in S$ of type UA, constraints (3.3) ensures that each aisle action of the respective aisle collects a sufficient quantity of s. Therefore, constraints (3.4c) are redundant for articles of types UP and UA. Note that every article of type UP is associated with a unique aisle.

The size of formulation HI, i.e., model (3.4), can be summarized as follows:

Property 1. *Formulation HI has $\mathcal{O}(m+n^2)$ variables, $\mathcal{O}(m+a)$ constraints, and $\mathcal{O}(m+an^2)$ non-zero coefficients (recall that n denotes the number of possible pick positions and $a = |S|$ denotes the number of different articles to be collected).*

Proof. The number of variables coincides with the number of edges in the extended state space which grows quadratically in the relevant pick positions and is bounded by $\mathcal{O}(m + n^2)$. The number of constraints is bounded by $\mathcal{O}(m + a)$, where constraints (3.4b) contribute the summand m and constraints (3.4c) the summand a. The number of non-zero coefficients is bounded by $\mathcal{O}(m + an^2)$, since the incidence matrix has exactly $2|E| = \mathcal{O}(m + n^2)$ non-zeros, and the demand-covering constraints (3.4c) have $\sum_{s \in S} |E_s| = \mathcal{O}(an^2)$ non-zero coefficients. □

Additionally, Heßler and Irnich (2024) define dominance rules to reduce the number of parallel edges in E. Dominance leads to smaller state spaces and herewith a reduced number of variables in formulation HI. We also apply dominance to reduce the edges in the state space and to accelerate the solution of formulation HI by the MIP solver. Even with dominance, Property 1 is valid and it is straightforward to construct SPRP-SS instances for which the presented bounds are sharp.

Model (3.4) of the SPRP-SS can also be interpreted as a *resource-constrained shortest path problem* (RCSP, Pugliese and Guerriero, 2013) or a *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005). This is achieved by transforming the DCCs into upper-bounding constraints which limit the uncollected supply to the difference between the total supply and demand. However, solving the SPRP-SS with an RCSP or SPPRC algorithm can be difficult (even when using powerful techniques such as bidirectional dynamic-programming labeling with completion bounds). The reason is that the number of resource constraints equals the number of articles ($a = |S|$), which becomes prohibitively large for typical SPRP-SS instances with more than a hundred different articles. Therefore, no effective RCSP or SPPRC algorithm has been applied to solve this type of problem to date.

3.4 Linear-Size State Spaces

As stated in Property 1, the number of edges in the state space of Heßler and Irnich is bounded by $\mathcal{O}(m + n^2)$. It grows quadratically with the number n of relevant pick positions, as defined in Eq. (3.2). Only the actions `gap` are responsible for this quadratic growth. Thus, we replace all parallel edges between states connected via actions `gap` by a small subnetwork of linear size. This can be done in different ways. We now present two types of subnetworks of linear size that can be included in the extended state space of formulation HI to fully replace all of the `gap` actions.

3.4.1 Enlarged Gaps Network

Let $\sigma, \sigma' \in V$ be two vertices connected with an action `gap` in the state space $G = (V, E)$. Moreover, let $j \in J$ be the associated aisle and let I_j denote the set of all

relevant pick positions in this aisle (numbered from 1 to C, bottom up). Depending on the state of σ also the actions `top`, `bottom`, and `void` may connect σ with σ', i.e., they are parallel edges. Figure 3.1a shows the details:

- For $\sigma \in \{\texttt{UU1c}, \texttt{EE1c}, \texttt{EE2c}\}$, actions `gap`, `top`, and `bottom` connect to the same vertex σ'.
- For $\sigma = \texttt{0E1c}$, only action `top` connects to the same vertex σ' as action `gap`.
- For $\sigma = \texttt{E01c}$, only action `bottom` connects to the same vertex σ' as action `gap`.
- For $\sigma = \texttt{000c}$, neither `top` nor `bottom` connects to the same vertex as `gap`.

The subnetwork that we construct includes all of the above parallel edges.

The basic idea is that two actions $\texttt{top}(h)$ and $\texttt{bottom}(i)$ are performed sequentially in the subnetwork so that they replicate the respective action $\texttt{gap}(h, i)$. The construction is done in three steps (Figure 3.3 shows an example of a subnetwork):

1. Additional vertices v_{hi} are introduced for each action $\texttt{gap}(h, i)$ where $h \in I_j$ and $i \in I_j$ with $h < i$ are consecutive pick positions in the considered aisle j. Also, let $M > i_{\max} = \max I_j$ and $h_{\min} = \min I_j > 0$.

   Moreover, one or two vertices are added to replicate the original actions `bottom` and `top` connecting vertex σ with vertex σ' (see above for which type of state this is relevant). For the original action `bottom`, the vertex $v_{i_{\max},M}$ is added. For the original action `top`, the vertex $v_{0,h_{\min}}$ is added. Note that for $\sigma \in \{\texttt{UU1c}, \texttt{EE1c}, \texttt{EE2c}\}$ both vertices are included in the network. In case that the action `void` connects σ with σ', it can be omitted in the subnetwork.

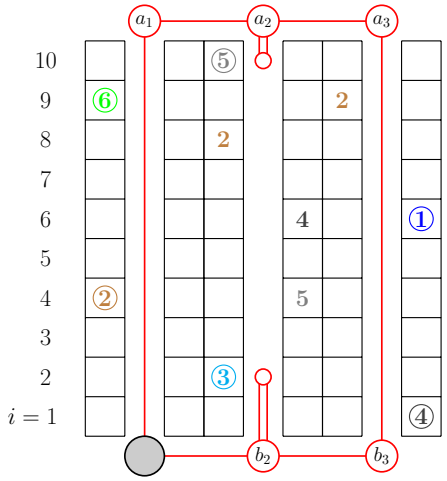
2. For the actions $\texttt{bottom}(h)$, edges are added to connect σ with v_{hi} (ingoing edges of v_{hi}). Similarly, edges for the actions $\texttt{top}(i)$ are added to connect v_{hi} with σ' (outgoing edges of v_{hi}).

   Actions $\texttt{bottom}(0)$ and $\texttt{top}(M)$ have a cost of 0, where M and 0 indicate that the aisle is not entered from the top or the bottom, respectively.

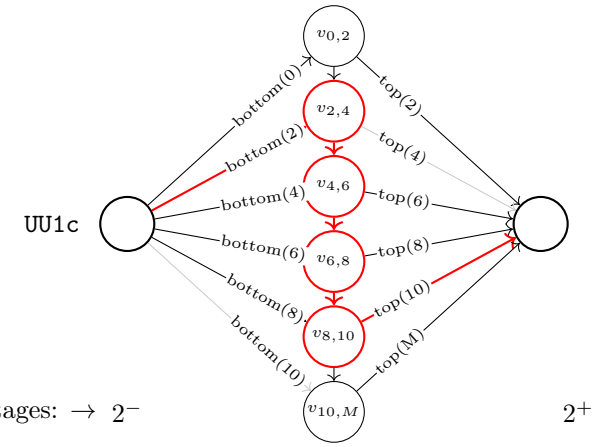
3. Downward edges of cost 0 between the additional vertices created in the first step enable actions `gap` with more space between the turning points. As a result, an action $\texttt{bottom}(h)$ can be combined with an action $\texttt{top}(i)$ also for non-consecutive pick positions h and i.

For a given (original) state space $G = (V, E)$, we denote by $G' = (V', E')$ the state space that replaces all actions `gap`, `top`, `bottom`, and `void` that are represented by parallel edges in G by the respective subnetwork. By construction, the new state space G' has $\mathcal{O}(m + n)$ vertices and $\mathcal{O}(m + n)$ edges.

Example 2. *Figure 3.3 visualizes a warehouse with aisles $J = \{1, 2, 3\}$ and six articles $S = \{1, 2, \ldots, 6\}$ that must be collected (unit demand, i.e., $q_1 = \cdots = q_6 = 1$)*



(a) A warehouse with six articles. Articles 2 and 4 are available in different aisles (O), article 5 has a unique aisle (UA), and articles 1, 3 and 6 have a unique pick position (UP).



(b) Subnetwork EG for aisle $j = 2$ and the state of type `UU1c`. The highlighted path (red/bold) corresponds to `gap`(2,10) in Figure 3.3a. Gray edges can be omitted from the subnetwork by dominance considerations.

Figure 3.3: Example for a network which can enlarge the gaps.

together with the optimal picker tour. We consider the second aisle $j = 2$ with five possible pick positions and the resulting subnetwork for the state $\sigma =$ `UU1c`.

Dominance considerations can reduce the number of edges in the subnetwork. For example, the edge for the action `top`(4) *can be omitted, since article 5 would be collected twice, which is not necessary for unit demand. The same is true for* `bottom`(10).

Figure 3.3b and Example 2 are also helpful to highlight differences when either the original state space G or the new state space G' that includes the subnetworks are used within model (3.4). In formulation HI (using G), the *demand-covering constraints* (DCCs) (3.4c) are redundant for articles of type UP and UA, i.e., $s \in \{1, 3, 5, 6\}$ in the example. However, the structure of the subnetwork allows to use invalid combinations of actions. For example, if articles $s = 3$ and $s = 5$ have no associated constraint (3.4c), then the action `top`(M) can be combined with action `bottom`(0), which represents `void` and is not feasible due to (3.3). As a result, model (3.4) defined over the new state space G' is only valid for the SPRP-SS when all constraints of type (3.4c) are present for all $s \in S$. This version of model (3.4) defines formulation EG that we compare against formulation HI (and a third one defined in the next subsection). Even with more DCCs in formulation EG, the new formulation is linear as stated in the following property:

Property 2. *Formulation EG is of linear size, and it has* $\mathcal{O}(m+n)$ *variables and* $\mathcal{O}(m+n)$ *constraints.*

Proof. The number of vertices and edges per subnetwork is proportional to the number of pick positions in the respective aisle. Hence, the state space G' has not more than $\mathcal{O}(m+n)$ vertices and edges. Assuming that $a = |S| \leq n$, formulation EG has $\mathcal{O}(m+n)$ variables and constraints. □

The size of a model is only one factor that impacts the performance of a MIP solver that tries to solve the model. A second factor is the strength of the LP relaxation. Comparing formulations HI and EG, we can expect that formulation EG has a weaker LP relaxation, since the additional DCCs but not the linear-size state space G' ensure demand coverage for articles of type UA and UP (this allows LP solutions in which, e.g., an action `gap` is combined with action `void` both with value 0.5). Hence, there exists a clear tradeoff between model size and model strength. Only empirical tests can tell which of both formulations is solved faster, see Section 3.5.

Property 3. *The DCCs* (3.4c) *in Formulation EG are redundant for articles* $s \in S$ *of type UA and UP.*

3.4.2 Reduced Gaps Network

Like the subnetwork EG, also the second subnetwork RG is also used to replace parallel edges between two vertices σ and σ' describing actions `gap` (and `top`, `bottom`, and `void` depending on the state of σ). The additional vertices of the RG network are constructed using the notion of *non-extensible gaps* (NEGs). A feasible action $\texttt{gap}(h,i)$ is an NEG, if no other action $\texttt{gap}(h',i')$ with $h' \leq h$, $i' \geq i$, and $(h',i') \neq (h,i)$ is feasible. This definition implies that if $\texttt{gap}(h,i)$ and $\texttt{gap}(k,\ell)$ are NEGs, then either $h < k$ and $i < \ell$ or $h > k$ and $i > \ell$. Note that, depending on the state of σ, we also include the artificial positions 0 and M to represent the actions `top`, `bottom` or `void` if feasible. Formally, this requires the consideration of state-dependent position sets I_j for each aisle $j \in J$. For state `000c`, we consider positions I_j without 0 and M; for state `E01c`, positions $I_j \cup \{0\}$; for state `0E1c`, positions $I_j \cup \{M\}$; for the remaining states `UU1c`, `EE1c`, and `EE2c`, positions $I_j \cup \{0, M\}$. For the sake of simplicity, we will refrain from further formalizing this detail.

The construction of the RG subnetwork is also done in three steps (Figure 3.4 shows an example of a subnetwork):

1. Additional vertices w_{hi} are introduced for each NEG $\texttt{gap}(h,i)$. Note that NEGs may have $h = 0$ or $i = M$.

2. For each $h \in I_j$ (or $h \in I_j \cup \{0\}$ for some states, see above), the edge for the action `bottom`(h) is added to connect σ with $w_{h'i'}$ (ingoing edges of $w_{h'i'}$) where (h', i') is the NEG with maximum index $h' \leq h$. Similarly, for each $i \in I_j$ (or $i \in I_j \cup \{M\}$ for some states, see above), the edge for the action `top`(i) is added to connect $w_{h'i'}$ with σ' (outgoing edges of $w_{h'i'}$) where (h', i') is the NEG with minimum index $i' \geq i$.

   Actions `top`(M) and `bottom`(0) have a cost of 0.

3. Upward edges of cost 0 between the vertices created in the first step enable actions `gap` with less space between the turning points. As a result, an action `bottom`(h) can be combined with an action `top`(i) also for pick positions h and i that do not correspond to an NEG.

For a given (original) state space $G = (V, E)$, we denote by $G^* = (V^*, E^*)$ the state space that replaces all actions `gap`, `top`, `bottom`, and `void` that are represented by parallel edges in G by the respective subnetwork. Intentionally, we disregard the action `2pass` which is, for some states, another parallel edge. Note that this omission does not impact the worst-case behavior. Furthermore, the action `2pass` is always dominated by a `gap` action if it happens to be a parallel edge in the state space. Also note that Revenant *et al.* (2025) have shown that the action `2pass` is redundant in single-block parallel-aisle warehouses. By construction, the new state space G^* has $\mathcal{O}(m + n)$ vertices and $\mathcal{O}(m + n)$ edges.

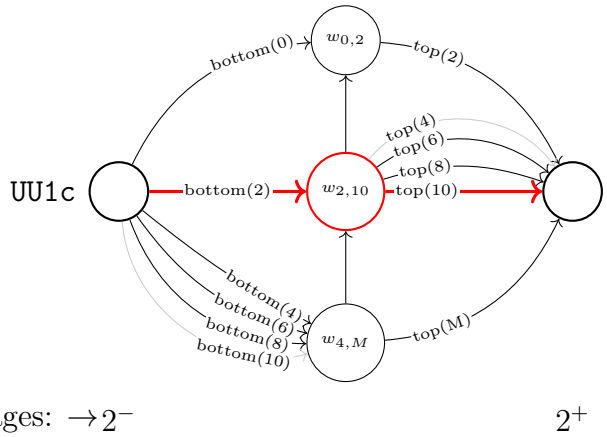


Figure 3.4: Subnetwork RG for aisle $j = 2$ and the state of type `UU1c`. The highlighted path (red/bold) corresponds to `gap`(2,10) in Figure 3.3a. Gray edges can be omitted from the subnetwork by dominance considerations.

Example 3. *(continued from Example 2) In Example 2, we focused on the aisle* $j = 2$ *and the state* $\sigma =$ `UU1c` *(see Figure 3.3). Under the unit-demand assumption, there are three NEGs* (h, i)*, namely*

- $(0, 2)$ *corresponding to the action* $\texttt{top}(2)$*,*
- $(2, 10)$ *corresponding to the action* $\texttt{gap}(2, 10)$*, and*
- $(4, M)$ *corresponding to the action* $\texttt{bottom}(4)$*.*

The three NEGs are to be considered in the construction of the RG network, which is shown in Figure 3.4. We have three additional vertices $w_{0,2}$*,* $w_{2,10}$*, and* $w_{4,M}$*. In addition, there are six edges for possible actions* $\texttt{bottom}(h)$ *and also six edges for possible actions* $\texttt{top}(i)$*. In both cases, four of the six possible actions are parallel edges in the subnetwork* G^**.*

As in Example 2, dominance allows to eliminate actions $\texttt{bottom}(10)$ *and* $\texttt{top}(4)$*.*

Using model (3.4) together with the new state space G^* gives a formulation that is correct for articles s of type UA and UP. The feasibility of the NEGs with respect to conditions (3.3) (by definition of an NEG) ensures that every path through the subnetwork RG allows to collect a sufficient amount of these articles s. There is no need to have the DCCs (3.4c) for them, i.e., they are surely redundant for articles of type UA and UP. Hence, the new RG network fixes one of the shortcomings that we identified for the EG network, see Section 3.4.1.

However, some possible paths in the subnetwork RG do not describe reasonable actions $\texttt{gap}(h, i)$. In Figure 3.4, e.g., one of the actions $\texttt{bottom}(h)$ for a cell $h \in \{4, 6, 8, 10\}$ could be traversed first, and the edge for $\texttt{top}(2)$ for cell $i = 2$ afterwards. These actions "overlap" in the sense that the pick positions 2 and 4 (and depending on h some more cells) are reached from the top as well as from the bottom. When assigning a supply to these edges (this is what we do in model (3.4) when using the coefficients b_{se} in the DCCs (3.4c)), this supply would be counted twice when using such overlapping actions of the subnetwork RG. This is uncritical for unit demand, but leads to an incorrect formulation for general demand.

We can summarize these observations for using model (3.4) together with the new state space G^* defined over the RG network, denoted as Formulation $\text{RG}^{(3.4)}$.

Property 4. *Formulation* $RG^{(3.4)}$ *is a valid formulation for the SPRP-SS with unit demand. It is of linear size, and it has* $\mathcal{O}(m + n)$ *variables and* $\mathcal{O}(m + n)$ *constraints. The DCCs* (3.4c) *are redundant for articles* $s \in S$ *of type UA and UP.*

For the general-demand case, we present a modified MIP model for the SPRP-SS that uses two types of variables: As before, binary variables x_e for all edges in the state space describe the *o*-*d*-path. In addition, continuous variables y_{sp} describe, for each article $s \in S$ and possible pick position $p \in P_s$, the amount that is collected. By bounding $y_{sp} \leq b_{sp}$, it is ensured that supply from this position is not double counted. In addition, let E_p denote the set of all edges and corresponding actions

that traverse position $p \in P$. The new model reads as follows:

$$z_{\text{SPRP-SS}} = \min \sum_{e \in E} c_e x_e \tag{3.5a}$$

$$\text{subject to} \quad \sum_{e \in \delta^+(\sigma)} x_e - \sum_{e \in \delta^-(\sigma)} x_e = \begin{cases} +1, & \text{if } \sigma = o \\ -1, & \text{if } \sigma = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall \sigma \in V \tag{3.5b}$$

$$\sum_{e \in E_p} b_{sp} x_e = y_{sp} \qquad \forall s \in S,\, p \in P_s \tag{3.5c}$$

$$\sum_{p \in P_s} y_{sp} \geq q_s \qquad \forall s \in S \tag{3.5d}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{3.5e}$$

$$0 \leq y_{sp} \leq b_{sp} \qquad \forall s \in S,\, p \in P_s \tag{3.5f}$$

The objective function (3.5a) and the constraints (3.5b) and (3.5e) remain identical to those of the first model (3.4). The DCCs (3.4c) are reformulated into constraints (3.5c) and (3.5d) making sure that a sufficient amount is collected and no supply is counted double. The feasible domain for the supply variables is stated in (3.5f).

We denote by Formulation $\text{RG}^{(3.5)}$ the use of the new model (3.5) defined over the new RG network G^*. We summarize what we know about this formulation:

Property 5. *Formulation $RG^{(3.5)}$ is a valid formulation for the SPRP-SS (for both unit demand and general demand). It is of linear size, and it has $\mathcal{O}(m+n)$ variables and $\mathcal{O}(m+n)$ constraints. The DCCs are redundant for articles $s \in S$ of type UA and UP, i.e., the corresponding variables y_{sp} and constraints* (3.5c), (3.5d), *and* (3.5f) *can be omitted.*

Strength of LP Relaxations For comparing different formulations to the same problem, not only the size of the respective models is important. For unit-demand instances of the SPRP-SS, we will now show that the LP relaxation of formulation $\text{RG}^{(3.4)}$ is at least as strong as the one of formulation $\text{RG}^{(3.5)}$. Note that a stronger LP relaxation can reduce the number of branch-and-bound nodes to be evaluated by the MIP solver. This often lowers the total computation time consumed by the MIP solver. Let $z_{\text{LP}}^{\text{RG}(3.4)}$ and $z_{\text{LP}}^{\text{RG}(3.5)}$ denote the objective value of the LP relaxation of formulation $\text{RG}^{(3.4)}$ and $\text{RG}^{(3.5)}$, respectively.

Property 6. *For all instances of the SPRP-SS with unit-demand, the relation $z_{LP}^{RG(3.4)} \geq z_{LP}^{RG(3.5)}$ holds.*

Proof. Recall that in the unit-demand case, $q_s = 1$ for all $s \in S$ as well as $b_{se} = 1$ and $b_{sp} = 1$ for all $e \in E_s$ and $p \in P_s$, respectively. Therefore, we can always use

b_{se} instead of b_{sp} for the positions $p \in P_s$ an edge $e \in E_s$ visits.

Let the flow $\bar{\mathbf{x}} = (\bar{x}_e) \in [0,1]^E$ denote a feasible solution to the LP relaxation of $\text{RG}^{(3.4)}$. We will prove the statement by showing that this solution can be translated into a feasible solution to the LP relaxation of $\text{RG}^{(3.5)}$ with the same (or lower) cost. To this end, let

$$\bar{y}_{sp} = \min \left\{ 1, \sum_{e \in E_p} \bar{x}_e \right\}$$

for all $s \in S$ and $p \in P_s$.

With these definitions it is guaranteed that the solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies all constraints of (3.5), where only the fulfillment of constraints (3.5d) needs to be shown. We distinguish two cases for each article $s \in S$: If $\bar{y}_{sp} = 1$ for at least one of the positions $p \in P_s$, we trivially have $\sum_{p \in P_s} \bar{y}_{sp} \geq 1$, i.e., constraint (3.5d) holds for article s. Otherwise, $\min\{1, \sum_{e \in E_p} \bar{x}_e\} = \sum_{e \in E_p} \bar{x}_e$ for all $p \in P_s$. It follows:

$$\begin{aligned}
\sum_{p \in P_s} \bar{y}_{sp} &= \sum_{p \in P_s} \min \left\{ 1, \sum_{e \in E_p} \bar{x}_e \right\} \\
&= \sum_{p \in P_s} \sum_{e \in E_p} \bar{x}_e \\
&= \sum_{p \in P_s} \sum_{e \in E_p} b_{se} \bar{x}_e \\
&\geq \sum_{e \in E} b_{se} \bar{x}_e \geq 1,
\end{aligned}$$

where the last inequalities follow from the facts that each edge e with $b_{se} = 1$ may appear more than once in the double summation, and constraints (3.4c) holds for $\bar{\mathbf{x}}$, respectively. Therefore, inequalities (3.5d) are also fulfilled in this case, which completes the proof. □

Note that the above proof does not hold for the general-demand case, because we cannot exploit $b_{sp} = b_{se}$ for all $e \in E_p$ and all $s \in S$.

3.5 Computational Results

In what follows, we describe the generation of the benchmark instances used in the following experiments, we compare the LP relaxations of formulations HI, EG, and RG, we report results for improved formulations in which only some (promising)

parts of the original state spaces are replaced by subnetworks, and we finally compare the best formulations as integer programs.

3.5.1 Benchmark Instances

The instances of the SPRP-SS are generated as described in detail in Heßler and Irnich (2024) and Lüke *et al.* (2024). An instance is characterized by a combination of (m, C, a, α), where m denotes the number of aisles, C the number of locations per aisle, a the number of different articles to be collected, and α the scatter factor. The scatter factor describes the average number of pick positions at which an article is stored in the warehouse. We assume a classical single-block rectangular warehouse with $m = 10$ aisles and $C = 50$ locations per aisle. Additionally, we vary the number of different articles that need to be collected, i.e., $a \in \{30, 50, 100, 200\}$.

For the generation of a pick list and for the assignment of articles to pick positions, we use the following approach that is realistic and simpler than what has been used in Lüke *et al.* (2024) where class-based storage policies were analyzed. First, we randomly assign a different pick position $p \in P$ to every article $s \in S$ defining a pair $(s, p) \in S \times P$. Second, another $(\alpha - 1)a$ pairs (s, p) are randomly drawn while making sure that no identical pairs are generated. This procedure ensures that each article $s \in S$ can be found at least once in the warehouse, and that it can be collected from α different positions on average. Due to randomness, some articles $s \in S$ can be found at more than α positions while others at fewer positions. For the following studies, we use a scatter factor of $\alpha = 5$.

In addition, we generate instances with unit demand, i.e., $q_s = 1$ for all $s \in S$, and instances with general demand. For the latter, we first draw a random supply $b_{sp} \in \{1, 2, 3\}$ for each pair (s, p). Demands q_s are then drawn randomly from $\{1, 2, \ldots, \min\{6, \sum_p b_{sp}\}\}$. In particular, all general-demand instances are feasible by construction.

In order to produce statistically significant results, we generate 100 instances per combination (m, C, a, α). In total, the benchmark set comprises $2 \cdot 4 \cdot 100 = 800$ instances (factor 2 for unit and general demand; 4 combinations (m, C, a, α)). The benchmark is available at `https://logistik.bwl.uni-mainz.de/research/benchmarks/`.

3.5.2 LP Relaxation Results

We first analyze empirically the strength of the LP relaxation of the formulations HI, EG, and RG. Recall that, for a given instance of the SPRP-SS, $z_{\text{SPRP-SS}}$ is the objective value. A lower bound is given by the objective value of the LP relaxation of the respective formulation, in the following denoted by z_{LP}. The *optimality gap*

is defined as $100 \cdot (z_{\text{SPRP-SS}} - z_{\text{LP}})/z_{\text{SPRP-SS}}$, and it describes the strength of a formulation.

Formulation		general demand					unit demand				
	DCCs for	$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
HI	O	4.83	3.47	1.71	0.33	2.58	6.30	8.31	8.60	7.76	7.74
EG	O, UA, UP	8.37	8.83	7.88	5.63	7.68	6.39	8.41	8.63	7.76	7.80
$\text{RG}^{(3.4)}$	O	*n.a.*	*n.a.*	*n.a.*	*n.a.*	*n.a.*	6.40	8.42	8.63	7.76	7.80
$\text{RG}^{(3.5)}$	O	5.28	3.69	1.82	0.35	2.79	12.21	14.03	13.19	11.09	12.63

Table 3.1: Average relative optimality gap in percent, i.e., $100 \cdot (z - z_{LP})/z$ where z is the optimal integer solution and z_{LP} the LP relaxation bound.

Table 3.1 shows average optimality gaps (per group of 100 instances) for the formulations HI, EG, $\text{RG}^{(3.4)}$, and $\text{RG}^{(3.5)}$. Gaps are not available ("*n.a.*") for formulation $\text{RG}^{(3.4)}$ and general demand since the model is not valid in this case. The LP relaxation of formulation HI always achieves the smallest gaps. This is not only true on average, but also for each single SPRP-SS instance. This results from the fact that every feasible flow $\bar{\mathbf{x}} = (\bar{x}_e) \in [0, 1]^E$ of the LP-model HI can be translated into a feasible flow of the other LP formulations. We refrain from a formal proof of this statement as the translation is trivial.

For the unit-demand case, we can observe the direct consequences of Property 6. Indeed, formulation $\text{RG}^{(3.4)}$ can be strictly stronger than formulation $\text{RG}^{(3.5)}$ in the unit-demand case. (The large difference can be explained by the fact that for an edge e that covers two or more positions in which an article s is stored, the relation $b_{se} = 1 < \sum_{p \in P: e \in E_p} b_{sp}$ holds. Note that b_{se} is used in model (3.4) and b_{sp} in model (3.5).) Since formulation $\text{RG}^{(3.5)}$ is composed of more variables and constraints, it is strictly dominated. As a consequence, we will only consider the stronger formulation $\text{RG}^{(3.4)}$ for SPRP-SS with unit demand. Since model (3.5) is the only choice for general demand and the RG network, we can omit the superscript and refer to *formulation RG* in the following (to lighten the notation).

Comparing formulations EG and RG, we see from Table 3.1 that neither formulation dominates the other with regard to the LP relaxation. Average optimality gaps can differ substantially here, see $a = 100$ with gaps of 7.88 and 1.82 percent for general demand and gaps of 8.63 and 13.19 percent for unit demand, respectively. In contrast, for the unit-demand SPRP-SS instances that we consider, the average optimality gaps of formulations EG and RG are very close.

3.5.3 Replacement Rules for Creating Subnetworks

The replacement of parallel edges of the original network by subnetworks of

types EG and RG is reasonable only if the size of the new network is smaller. Indeed, less edges in the resulting EG and RG networks imply less variables in the resulting formulations EG and RG, respectively. However, the subnetworks introduce additional vertices v_{hi} or w_{hi}, one for each pick position or NEG, see Sections 3.4.1 and 3.4.2. Additional vertices require additional flow conservation constraints (3.4b) and (3.5b). Therefore, we propose to fine-control the replacement process by deciding, for each set of parallel edges separately, whether the replacement is performed or the parallel edges of the original network are kept. In this subsection, we will computationally analyze different options of controlling the replacement process.

To this end, for a subset of parallel edges under consideration, let k^{orig} and k^{new} denote the number of edges in the original and the new network. In addition, let ℓ_+^{new} denote the number of additional vertices v_{hi} or w_{hi} that would have to be included if the new replacement were performed. We use the condition

$$k^{\text{new}} + f\ell_+^{\text{new}} < k^{\text{orig}}$$

to decide on the replacement, where the *vertex factor* (VF) f estimates how strong the impact of the number ℓ_+^{new} of additional vertices is. Furthermore, we use the alternative conditions $k^{\text{new}} + (\ell_+^{\text{new}})^2 < k^{\text{orig}}$ (denoted as *quadratic*) and $k^{\text{new}} + (\ell_+^{\text{new}})^3 < k^{\text{orig}}$ (*cubic*) to simulate a very high impact of the additional vertices.

Formulation		general demand					unit demand				
	VF f	$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
EG	0	34.36	45.67	53.94	47.80	45.44	47.67	63.02	75.65	81.05	66.85
	1	32.28	44.12	52.78	45.79	43.74	47.15	62.99	75.64	81.05	66.71
	2	26.69	39.28	48.94	39.66	38.64	44.42	62.69	75.61	81.05	65.94
	3	19.38	33.20	43.01	31.08	31.67	36.97	61.80	75.55	81.04	63.84
	5	5.01	20.47	28.54	18.61	18.16	14.32	55.79	74.34	80.68	56.28
	10	0.00	0.61	8.91	2.62	3.03	0.00	3.54	65.61	72.79	35.48
RG	0	48.66	59.17	66.53	62.34	59.17	59.96	72.21	81.97	86.13	75.07
	1	48.52	59.10	66.51	62.31	59.11	59.95	72.21	81.97	86.13	75.06
	2	48.16	58.94	66.43	62.19	58.93	59.94	72.21	81.97	86.13	75.06
	3	47.69	58.74	66.33	62.03	58.70	59.92	72.21	81.97	86.13	75.06
	5	46.40	57.85	65.92	61.06	57.81	59.88	72.19	81.97	86.13	75.04
	10	42.09	55.13	63.70	56.36	54.32	59.41	72.17	81.97	86.13	74.92
	quadratic	47.50	58.14	65.29	57.66	57.15	59.94	72.20	81.97	86.13	75.06
	cubic	44.64	52.76	55.22	30.62	45.81	59.88	72.18	81.96	86.12	75.03

Table 3.2: Average percentage of edge reduction, replacements depending on VF f or rules *quadratic* and *cubic*.

Table 3.2 shows the average percentage of edge reduction, i.e., $100 \cdot (1 - (|E| -$

$|E^{\text{new}}|)/|E|)$, where E^{new} is the set of edges of the resulting network EG or RG. As before, the results are grouped by network type (EG or RG), the number a of different articles to pick, and the replacement condition used. Please note that the replacement rules *quadratic* and *cubic* did not significantly change the original network when subnetworks of type EG were tested. As a consequence, we do not report results for EG combined with *quadratic* and *cubic*. The most important results are:

(i) The reduction is stronger for the subnetworks RG than for the subnetworks EG.

(ii) Reductions increase with a, with the exception of general demand and $a = 200$, where one can observe a decline. This is due to the increased absolute number of positions that cannot be omitted in a tour. Therefore, the number of feasible `gap` actions per aisle might be smaller.

(iii) For the subnetworks EG, an increasing VF f strongly impacts the percentage reduction.

(iv) For the subnetworks RG, the respective reduction is moderately decreasing with an increasing VF f.

Table 3.3 uses the identical grouping and shows the average computation times of solving the formulations EG and RG, respectively. Compared to the percentage of edges reduced (Table 3.2), the computation times are not directly related to the percentage reduction. This is a strong hint that also the number of vertices is important. However, none of the replacement rules and values of the VF f has a clear advantage over the others. For general demand, it seems that for smaller $a = 30$ and 50 the rule *quadratic* or a large VF f are advantageous for both types of networks EG and RG, while for larger $a = 100$ and 200 a smaller VF f leads to short computation times. For unit demand and network EG, it is difficult to give a good recommendation for the choice of VF f. In contrast, for unit demand and network RG, computation times are almost independent of the choice of VF f. In summary, Table 3.3 shows that using a small vertex factor between 0 and 2 often results in the shortest times (over all replacement rules). As a consequence, we will use the replacement rule based on vertex factor $f = 1$ for the final experiments.

Finally, Table 3.3 shows that average computation times for subnetworks RG are always lower than the corresponding times for subnetworks EG. Especially for general demand and $a = 100$ and 200, differences are substantial. We attribute the better performance of formulation RG in these cases of general demand to the stronger LP relaxation (see Table 3.1) and the larger number of reduced edges (see Table 3.2).

Formulation		general demand					unit demand				
	VF f	$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
EG	0	238	474	977	1158	712	199	456	1100	2197	988
	1	222	437	971	1259	722	205	452	1099	2105	965
	2	220	405	907	1397	732	198	452	1092	2195	984
	3	218	378	877	1553	756	210	441	1092	2190	983
	5	213	397	883	1685	795	245	469	1116	2104	984
	10	204	397	894	1884	845	178	673	1448	2839	1284
RG	0	191	248	375	596	353	160	347	827	1553	722
	1	189	248	376	600	353	160	349	830	1552	723
	2	188	244	372	603	352	158	348	829	1555	722
	3	188	246	377	614	356	159	349	829	1554	723
	5	188	249	376	644	364	159	348	830	1557	724
	10	189	271	401	776	409	164	349	830	1556	725
	quadratic	179	244	374	748	387	160	347	828	1553	722
	cubic	184	279	502	1404	592	160	348	829	1557	724

Table 3.3: Average computation times [in milliseconds], replacements depending on VF f or rules *quadratic* and *cubic*.

3.5.4 Integer Solutions

In the final computational experiments, we compare all three formulations HI, EG, and RG using the vertex factor $f = 1$ for the latter two. For formulations HI and RG, it is still not clear whether redundant DCCs for articles of type UA and UP should be used. To analyze this, we use three settings where DCCs are included either only for articles of type O, or of types O and UA, or for all types. Table 3.4 shows the average computation times of the MIP solver for the seven formulations. The results can be summarized as follows:

(i) For general and unit demand, formulation RG is always solved fastest compared to formulations HI and EG. This is true for all values a and in the average (see column *avg.*).

(ii) Adding redundant DCCs (for articles of type UA and UP) accelerates the solution process for both formulation HI and formulation RG in the unit-demand case. Computation times for the general-demand case are not impacted significantly by the choice of DCCs.

Additionally, we compare the three formulations using the setting with all DCCs (for articles of types O, UA, and UP) by computing a *speedup* factor. The baseline is always formulation HI which represents the status quo. Thus, Table 3.4 presents speedup factors for formulations EG and RG, which are computed as ge-

Formulation		general demand					unit demand				
	DCCs for	$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
HI	O	212	391	933	1933	867	183	632	2638	8165	2905
	O, UA	216	402	996	2048	916	182	638	2646	8986	3113
	O, UA, UP	213	400	957	1997	892	182	621	2574	7573	2737
EG	O, UA, UP	222	437	971	1259	722	205	452	1099	2105	965
	speedup	·0.90	·0.93	·1.13	·1.53	·1.10	·0.94	·1.36	·2.24	·3.40	·1.77
RG	O	189	248	376	600	353	160	349	830	1552	723
	O, UA	189	247	378	609	356	156	342	840	1599	734
	O, UA, UP	190	245	365	599	350	155	338	808	1426	682
	speedup	·1.02	·1.55	·2.56	·3.18	·1.89	·1.22	·1.84	·3.11	·4.87	·2.42

Table 3.4: Average computation times [in milliseconds], final setup with VF $f = 1$. *Note:* Speedups are computed as geometric means of ratios of computation times relative to formulation HI with all DCCs (O, UA, UP).

ometric means of the ratios $t_{HI}(I)/t_{EG}(I)$ and $t_{HI}(I)/t_{RG}(I)$ where $t_{HI}(I)$, $t_{EG}(I)$, and $t_{RG}(I)$ are the computation times of the MIP solver using the respective formulation for solving SPRP-SS instance I. Any speedup value greater than one indicates that the new formulation is superior to the status quo for the subgroup of instances.

These numbers clearly indicate that already formulation EG can be advantageous compared to formulation HI. The speedup for formulation RG is even higher with an overall speedup factor between 1.02 and 3.18 for general demand and between 1.22 and 4.87 for unit demand. Notably, speedups are higher for larger instances with more articles to be collected, i.e., for larger values of a.

3.5.5 Results for Two-Block Warehouse Layout

In this subsection, we show that the network modification according to the EG and RG replacements is applicable also for variants of the SPRP-SS. More precisely, every basic SPRP-SS variant solvable with a DP-based algorithm is suitable if it allows `gap` actions (some routing policies do not) and admits the integration of scattered storage via the extension of Heßler and Irnich (2024). One such variant is for the two-block warehouse layout, i.e., where a middle aisle splits the rectangular warehouse into two blocks such that subaisles can be entered and traversed separately. For this two-block layout, Roodbergen and de Koster (2001b) generalized the DP of Ratliff and Rosenthal (1983) that now comprises 25 instead of only seven states. In the state space of Roodbergen and de Koster, the

six (sub)aisle traversals E_j^{aisle} remain the same, while the cross-aisle traversals $E_j^{cross} = \{\mathtt{000}, \mathtt{110}, \mathtt{101}, \mathtt{011}, \mathtt{112}, \mathtt{121}, \mathtt{211}, \mathtt{220}, \mathtt{202}, \mathtt{022}, \mathtt{222}\}$ model the movements through the three cross-aisles. Heßler and Irnich already analyzed the SPRP-SS in the two-block case.

To compare with the results of the single-block case of Sections 3.5.2–3.5.4, we again assume a scatter factor of $\alpha = 5$, $m = 10$ aisles, and $C = 50$ cells per aisle. We equally divide each aisle so that each subaisle has 25 cells. Since the vertex factor f does not have a significant impact on the computational results (see Section 3.5.3), we set it to $f = 1$ again.

If we keep the number of articles to collect identical to what was assumed in the previous experiments (i.e., $a \in \{30, 50, 100, 200\}$), we can expect that on average only half as many pick positions result per subaisle. Therefore, the subnetworks cannot provide a similar degree of reduction of the edges as shown in Tables 3.2. For the same reason, computation times of formulations EG and RG should be closer to those of formulation HI (see Tables 3.3 and 3.4). To create an additional demand scenario where on average the same number of SKUs are located in each subaisle, we also consider the setting where we double the number of articles (compared to the *standard* scenario). Note that we do however not consider the case with $a = 400$ different articles to be collected, since this picklist size seems unreasonably large.

Formulation		general demand					unit demand				
Scenario		$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
standard	EG	9.71	23.61	37.42	36.34	26.77	14.26	35.92	57.11	66.82	43.53
	RG	29.15	42.45	54.66	54.65	45.23	37.17	53.21	68.41	75.77	58.64
		$a = 60$	100	200		avg.	$a = 60$	100	200		avg.
double	EG	27.55	37.42	36.34		33.77	42.96	57.11	66.82		44.70
	RG	45.95	54.66	54.65		51.76	57.98	68.41	75.77		59.57

Table 3.5: Average percentage of edge reduction for the two-block case using VF $f = 1$.

Table 3.5 shows the *standard* scenario in the upper part and the *double* scenario in the lower part. Compared to the results for the single-block layout in Table 3.2, we can observe a significant drop in the percentage of edges that are replaced in the *standard* scenario, while results in the *double* scenario reductions are closer to those of the single-block layout. We can explain the smaller reduction percentages as follows: In the two-block layout, each subaisle comprises only half of the number of pick positions (=25 cells) compared to an aisle in the single-block layout (=50 cells per aisle). This leads, on average, to more SKUs being located at identical pick

positions (left or right, stacked upon each other). This fact lowers the possibilities to reduce the network size in both the EG and RG formulations.

Similar to the integer results for the single-block layout presented in Section 3.5.4, we finally compare average total MIP solver times for the three formulations HI, EG, and RG. Tables 3.6a and 3.6b summarizes the computational results, again for the *standard* and the *double* scenario, respectively. First and foremost, an obvious outcome is that again formulation RG provides the shortest average computation times. All speedup values are greater than one, except for the small instance setting with $a = 30$ where formulation HI is slightly faster than formulation RG. In contrast to the results of Section 3.5.4, formulation EG performs worse than formulation HI. We attribute this to the also significantly smaller percentage of reduced edges (see Table 3.5).

Formulation		general demand					unit demand				
standard		$a = 30$	50	100	200	avg.	$a = 30$	50	100	200	avg.
HI	O, UA, UP	1659	5198	9696	8536	6272	1452	5303	41231	198686	61668
EG	O, UA, UP	2673	8654	39385	39731	22611	1611	7148	101549	385788	124024
	speedup	·0.93	·0.92	·0.54	·0.59	·0.72	·0.99	·0.94	·0.63	·0.61	·0.77
RG	O, UA, UP	1804‡	2987	3188	2551	2633	1255	3326	22201	89878	29165
	speedup	·1.12	·1.62	·2.79	·3.22	·2.01	·1.27	·1.69	·2.10	·2.91	·1.90

(a) Scenario *standard*. ‡ Here, average computation times are affected by outliers exhibiting relatively long RG computation times. However a proper speedup (> 1) can be observed. Recall that the speedup is computed as a geometric mean of computation time ratios.

Formulation		general demand				unit demand			
double		$a = 60$	100	200	avg.	$a = 60$	100	200	avg.
HI	O, UA, UP	6168	9696	8536	8134	8943	41231	198686	82953
EG	O, UA, UP	13433	39385	39731	30850	12314	101549	385788	166550
	speedup	·0.83	·0.54	·0.59	·0.64	·0.94	·0.63	·0.85	·0.69
RG	O, UA, UP	3187	3188	2551	2975	5408	22201	89878	39162
	speedup	·1.79	·2.79	·3.22	·2.53	·1.60	·2.10	·2.91	·2.14

(b) Scenario *double*.

Table 3.6: Average computation times [in milliseconds] for the two-block case, final setup with VF $f = 1$.

3.6 Conclusions and Outlook

In this paper, we considered the SPRP-SS and its out-of-the-box solution with a MIP model and MIP solver. The approach we took is that of Heßler and Irnich (2024) who first build an extended network (inspired by Ratliff and Rosenthal's dynamic program) and then solved a shortest-path problem with additional DCCs with a MIP solver. This shortest-path network contains parallel edges representing alternative traversal actions of type `gap`, `bottom`, and `top` within the aisle under consideration. The two new formulations EG and RG, which have been developed here, replace sets of parallel edges of the aforementioned types by generally smaller subnetworks. The acronyms signify the possibility to enlarge `gap` (EG) actions or to reduce `gap` (RG) actions. To be more precise, if a path through the subnetwork can represent the action $\texttt{gap}(h, i)$, then it can also represent gaps of type $\texttt{gap}(h', i')$ with $h' \leq h$ or $i' \geq i$ in the subnetwork EG. For the subnetwork RG, the reduction of the action $\texttt{gap}(h, i)$ into $\texttt{gap}(h', i')$ is enabled where $h' \geq h$ or $i' \leq i$.

We analyzed the resulting formulations EG and RG and derived several theoretical properties of the formulations. To accommodate general demand, modifications were required to the structure of the MIP model for subnetworks EG, where we could still ensure that the resulting model remains linear. In particular, we showed that all new MIP models are linear in the number m of aisles and in the number n of possible pick positions. In contrast, the MIP model of Heßler and Irnich (2024) is quadratic in n.

We studied the performance of the MIP solver-based approach with the new formulations in three types of computational experiments, in which 800 SPRP-SS instances were solved with up to $a = 200$ articles. First, the LP relaxations of the new formulations EG and RG have similar integrality gaps in case of unit demand, but they are weaker than that of the formulation of Heßler and Irnich (2024) in case of general demand. Second, the reduction in network size is largest for the new formulation RG, where typically more than two-thirds of the edges can be reduced. These reductions translate to improved computation times of the MIP solver. Third, for a wide range of vertex factors, the simplest replacement rule (we use $f = 1$, i.e., we perform the replacement whenever the resulting network has at least one edge less) was identified working reasonably well, leading to substantially smaller MIP computation times for formulation EG and especially for formulation RG compared to formulation HI. Overall, compared to the MIP of Heßler and Irnich (2024) the average speedup is of factor 1.89 for general demand and of 2.42 for unit demand for the SPRP-SS instances in the testbed. For the largest instances with $a = 200$ articles, average speedups reach the factors of 3.18 and 4.87, respectively. In absolute numbers, average computation times, even for these largest instances, are below two seconds.

We also showed that the network modification technique is generally applicable

for variants of the SPRP-SS, where the basic SPRP can be solved with dynamic programming. The computational experiments the variant of a two-block layout proved the superiority of the new formulation RG.

The new modeling approach may help to develop superior models for integrated in warehouse operations management problems, especially those where many instances of a problem similar to the SPRP-SS must be solved as subproblems multiple times. In the future, we expect that other SPRP-SS-specific solution approaches to emerge. Promising candidates include Lagrangian relaxation, combinatorial branch-and-bound algorithms, and Benders decomposition approaches, to name just a few.

Acknowledgement

Parts of this research were supported by Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/13-1 of project 555303283.

Bibliography

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2003). Concorde-03.12.19. Website. `https://www.math.uwaterloo.ca/tsp/concorde/index.html`.

Bartholdi, III, J. J. and Hackman, S. T. (2019). *Warehouse & Distribution Science*. Atlanta, GA 30332-0205 USA, release 0.98.1 edition. Online only, `https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf`.

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.

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

Daniels, R. L., Rummel, J. L., and Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, **105**(1), 1–17.

Fischetti, M., Salazar-Gonzalez, J.-J., and Toth, P. (2002). The generalized traveling salesman and orienteering problems. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, chapter 13, pages 609–662. Kluwer, Dordrecht.

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

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

Haouassi, M., Kergosien, Y., Mendoza, J. E., and Rousseau, L.-M. (2025). The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, **63**(4), 1304–1325.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, **126**(1), 106–130.

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*, **36**(6), 1417–1435.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer.

Khan, I., Maurer, O., Pätzold, J., Pszona, P., and Salchow, J.-D. (2024). Joint order selection, allocation, batching and picking for large scale warehouses. arXiv 2401.04563. `https://arxiv.org/abs/2401.04563`.

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.

Masae, M., Glock, C. H., and Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, **224**, 107564.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problems. *Journal of Association for Computing Machinery*, **7**, 326–329.

Öztürkoğlu, Ö., Gue, K. R., and Meller, R. D. (2012). Optimal unit-load warehouse designs for single-command operations. *IIE Transactions*, **44**(6), 459–475.

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

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

Pugliese, L. D. P. and Guerriero, F. (2013). A reference point approach for the resource constrained shortest path problems. *Transportation Science*, **47**(2), 247–265.

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.

Revenant, P., Cambazard, H., and Catusse, N. (2025). A note about a transition of Ratliff and Rosenthal's order picking algorithm for rectangular warehouses. *Operations Research Letters*, **62**, 107325.

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.

Singh, K. N. and van Oudheusden, D. L. (1997). A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, **97**(3), 571–579.

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., and Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, **305**(1), 222–239.

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, **95**, 139–150.

Weidinger, F., Boysen, N., and Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, **274**(2), 501–515.

Wildt, C., Weidinger, F., and Boysen, N. (2024). Picker routing in scattered storage warehouses: an evaluation of solution methods based on TSP transformations. *OR Spectrum*, **47**, 35–66.

Appendix

Problems and problem description:	
DCC	demand covering constraint
EG	subnetwork replacing parallel edges that allows to expand gaps
GTSP	generalized traveling salesman problem
HI	MIP model of Heßler and Irnich for the SPRP-SS
NEG	non-extensible gap: a `gap` action that cannot be extended to either side
O	articles that appear in more than one aisle in the warehouse
PTS	partial tour subgraph
RG	subnetwork replacing parallel edges that allows to reduces gaps
SKU	stock keeping unit
SPPRC	shortest path problem with resource constraints
SPRP	single picker routing problem
SPRP-SS	single picker routing problem with scattered storage
TSP	traveling salesman problem
UA	articles that only appear in one aisle in the warehouse but in more than one position
UP	articles that only appear in one unique position in the warehouse
VF	vertex factor
Methods and Algorithms:	
DP	dynamic program/dynamic programming
LKH	Lin-Kernighan-Helsgaun
LNS	large neighborhood search
LP	linear programming
MIP	mixed-integer programming
MTZ	Miller, Tucker & Zemlin

Chapter 4

The Single Picker Routing Problem with Scattered Storage in Parallel-Aisle Warehouses with Multiple Blocks

Stefan Irnich, Laura Lüke

Abstract

This paper investigates the b-block single picker routing problem with scattered storage (b-SPRP-SS). For a parallel-aisle warehouse comprising b blocks, the b-SPRP-SS asks for the determination of a picker tour that minimizes travel distance while collecting all articles from a given pick list. Scattered storage, where articles can be stored at multiple locations, substantially increases the problem's complexity by coupling the selection of collection points with routing decisions. Existing research on the b-SPRP-SS has predominantly focused on single-block and two-block warehouse layouts. To address this gap, we propose a novel formulation for warehouses with more than two blocks. The formulation is inspired by the dynamic-programming state spaces that Ratliff and Rosenthal introduced for the single-block case and Roodbergen and de Koster for the two-block case. The new state space is a relaxed one that omits connectivity information, thereby aggregating multiple original states into a single relaxed state. This relaxation significantly reduces computational complexity, although it may lead to disconnected tour fragments. To ensure route connectivity, the problem is solved using a branch-and-cut algorithm which dynamically adds subtour-elimination constraints. Extensive computational experiments demonstrate that the proposed approach is effective and outperforms the only other competitive exact approach from the literature that relies on a transformation into a generalized traveling salesman problem.

4.1 Introduction

Warehouse activities include receiving, storing, picking, packing, and shipping operations (Gu *et al.*, 2007). Excellent surveys introduce warehouse operations planning, including topics such as storage assignment, warehouse layout planning, zoning, routing, and batching (van Gils *et al.*, 2018; Boysen *et al.*, 2019). In this work, we address picker routing in manual (non-automated) warehouses where pickers move through the warehouse in order to collect articles from the storage locations called pick positions (picker-to-parts). De Koster *et al.* (2007) highlight that more than 80% of all order-picking systems in Western Europe are low-level picker-to-parts picking systems. Order picking denotes the process of retrieving inventory items from their storage locations in response to specific customer requests (de Koster *et al.*, 2007; Masae *et al.*, 2020b). Manual order picking is certainly very labor-intensive, and the literature gives different estimations for the effort: Typically, 60% of all labor activities in the warehouse result from order picking, and its cost can be estimated to be as much as 55% of the total warehouse operating expense (Drury, 1988; Tompkins *et al.*, 2003). Frazelle (2002) estimates that order picking contributes to up to 50% of the total warehouse operating costs. Given the cost breakdown of picking operations, the time spent moving personnel and equipment between storage locations can make up as much as 50% of a picker's working time (Richards, 2017). These figures explain why research on order picking operations, especially picker routing, is extensive and of high practical relevance.

In its basic form, the *single picker routing problem* (SPRP) seeks a minimum-length picker tour given a single-block parallel-aisle warehouse with the pick positions from where articles must be collected. The SPRP can be considered solved: On the one hand, the seminal work of Ratliff and Rosenthal (1983) shows that a minimum-length picker tour can be computed with dynamic programming (DP) in linear time (Heßler and Irnich, 2022b). On the other hand, the SPRP is practically well-solved with routing policies that are rule-based heuristics such as `traversal` (a.k.a. S-shape), `midpoint`, `largest gap` (Hall, 1993), `return`, `composite` (Petersen, 1997), and `combined` (Roodbergen and de Koster, 2001a). The application of heuristic routing policies is well justified in settings where pickers cannot perform all types of optimal tours or when optimal tours are complicated, counterintuitive, or difficult to memorize. Instead, pickers perform tours defined by some simple rules.

In this work, we consider the exact solution of the picker routing problem defined over a parallel-aisle warehouse with *multiple blocks* and *scattered storage*, which means an article may be stored at more than one pick position. The scattered storage (or mixed shelves, Weidinger, 2018) strategy adds a decision level to the problem, since it must be decided from which pick position an article is collected.

We denote this problem as the *b-block single picker routing problem with scattered storage* (b-SPRP-SS), where $b \geq 1$ describes the number of blocks.

Concerning the classical picker routing problem without scattered storage in multi-block warehouses, Prunet *et al.* (2025) showed that it is strongly NP-hard. Several approaches have been presented in the last decade for this problem: For the case of $b = 2$, the DP algorithms of Roodbergen and de Koster (2001b) can be used to determine an optimal picker tour. The solution principle is a direct extension of the DP algorithm of Ratliff and Rosenthal. The former requires the distinction of 25 instead of only seven different states per stage used in the latter. This is, however, irrelevant to the worst-case complexity: as shown by Heßler and Irnich (2022b), the DPs can be constructed and solved in linear time $\mathcal{O}(m + n)$, where m is the number of aisles and n the number of pick positions.

For $b = 1$ and $b = 2$ blocks, an approach based on *mixed-integer programming* (MIP) has been recently presented by Saylam *et al.* (2024). The SPRP is formulated as a variant of the arc routing problem, where the traditional subtour-elimination constraints are omitted and replaced by problem-specific disconnectivity constraints. To speed up the MIP solution, the disconnectivity constraints are dynamically added when violated so that the MIP solver implements a branch-and-cut algorithm.

For the multi-block case without scattered storage, i.e., when the number b of blocks can exceed two and is considered as part of the problem instance's input data, Çelik and Süral (2018) provided an excellent overview of the problem complexity, including a detailed overview of related problems (in addition, they present and analyze a graph theory-based heuristic). Straightforwardly, the b-SPRP can be modeled and solved as a (Steiner vertex) *traveling salesman problem* (TSP). However, a TSP-based approach does *not* exploit the rectilinear layout of the warehouse.

In contrast, Cambazard and Catusse (2018) considered the Steiner TSP over a rectilinear graph and provided an extension of Ratliff and Rosenthal's DP. For n points lying on h different horizontal lines, their DP can be solved in $\mathcal{O}(nh5^h)$ time. As a consequence, Pansart *et al.* (2018) showed that a similar DP can be defined for the b-SPRP. Although being better than the other MIP-based solution approaches that they tailor to b-SPRP by using Steiner TSP formulations, the DP fails for instances with more than $b = 10$ blocks (more precisely, the experiments are performed for $b \in \{3, 6, 11\}$, so that the behavior for $7 \leq b \leq 10$ is unclear). We will compare the exponential growth of their state space with the relaxed state space of our approach in Section 4.2.3.

A generic exact algorithm for picker routing in multi-block warehouses that outperforms the one by Pansart *et al.* (2018) has been presented by Schiffer *et al.* (2022). They use a layered graph algorithm that can also be used within a frame-

work, which can include additional attributes such as multiple drop-off points, dynamic batching policies, and cartless subtours.

Valle *et al.* (2017) proposed a branch-and-cut algorithm originally designed to address the *joint order batching and picker routing problem*, wherein customer orders are grouped into capacity-constrained batches to minimize the total travel distance required for picking within a warehouse. This approach can also be adapted to solve the standalone b-SPRP. Using this approach, instances involving up to five cross-aisles have been solved to optimality.

Apart from the exact solution methods, there are also some heuristic routing strategies for multi-block warehouses. Traversal, largest gap, and combined are heuristics originally designed for single-block warehouses which were adapted by Roodbergen and de Koster (2001a) for the multi-block case. Routing heuristics exclusively for multi-block warehouses are aisle-by-aisle (Vaughan, 1999) and no-reversal (Valle *et al.*, 2017). Theys *et al.* (2010) formulated the SPRP in multi-block warehouses as TSP and solved it heuristically. To do this, they used heuristics originally designed for the TSP. Although these do not use the advantages of the warehouse layout, they can be used for an arbitrary number of blocks without adjustment.

So far, picker routing with scattered storage has been primarily considered for $b = 1$ in the literature. Therefore, the problem is referred to as the *single picker routing problem with scattered storage* (SPRP-SS) in single-block warehouses, or rather, when it does not explicitly describe the multi-block case. Already Singh and van Oudheusden (1997) demonstrated that the SPRP-SS is a special case of the *traveling purchaser problem* (TPP). One year later, Daniels *et al.* (1998) formulated the problem based on the TSP and proposed a tabu search heuristic, representing one of the first algorithmic contributions specifically to the SPRP-SS. Despite its practical significance, research on picker routing with scattered storage remained limited for many years. Gu *et al.* (2007) explicitly noted this gap, emphasizing the research potential of the problem. It was only in the following decade that more systematic investigations emerged. Weidinger (2018) proposed a two-stage heuristic: in the first stage, pick positions are selected based on specific alternative rules, followed by the application of the Ratliff and Rosenthal (1983) DP algorithm for the resulting routing problem in the second stage. To analyze the gap to the optimal solution, they also proposed an MIP, which is solved by a solver. Weidinger *et al.* (2019) extended the heuristic to incorporate multiple depots. Recently, Wildt and Weidinger (2026) introduced a branch-and-cut algorithm capable of handling non-rectangular aisle layouts as well as multiple depots. Another MIP approach for single-block parallel-aisle warehouses was introduced by Goeke and Schneider (2021); it outperforms that of Weidinger (2018).

The current state-of-the-art exact method is that of Heßler and Irnich (2024),

who presented a generic approach to extend DP state space to incorporate scattered storage (for an overview, see Table 1 in Heßler and Irnich, 2024). Their computational study included the extension of the state spaces of Ratliff and Rosenthal and Roodbergen and de Koster, i.e., it covers the single-block and two-block cases. The final algorithm formulates an MIP model that can be solved using any type of IP solver. Recently, this approach has been refined by Lüke *et al.* (2026) who present a linear-size model that reduced the number of parallel arcs compared to the model of Heßler and Irnich (2024). The benefit of this model however degrades when the number of blocks increases. Lüke *et al.* (2024) explored the application of rule-based routing strategies, including traversal, midpoint, largest gap (Hall, 1993), return, and composite strategies (Petersen, 1997)—which are traditionally used for the SPRP. Their findings confirm that even when the routing subproblem is simplified via rules, the overall SPRP-SS remains NP-hard.

Concerning picker routing in multi-block warehouses with scattered storage, to the best of our knowledge, only Haouassi *et al.* (2025) and Su *et al.* (2023) propose exact approaches to solve the b-SPRP-SS. Haouassi *et al.* (2025) use a logic-based Benders decomposition method to solve the SPRP-SS in warehouses with two and three blocks and pick lists with not more than 30 articles. In addition, they adapt the approach of Schiffer *et al.* (2022) to warehouses with scattered storage. The two approaches solve two and five instances of a total of 16 instances for the 3-block setting within a time limit of 300 seconds. Su *et al.* (2023) propose a MIP-based approach, which is not compared to any other solution approach for the SPRP-SS. They solve small and medium-scale instances with two and five blocks. Wildt *et al.* (2025) solve the b-SPRP-SS heuristically. With the help of known transformation schemes, they convert the problem into a TSP and solve it with standard solvers for the TSP. Since the number of blocks does not affect the transformation of the b-SPRP-SS into a TSP, they also solve multi-block instances.

The b-SPRP-SS with unit demand, i.e., when only one unit of every article is requested, is a special case of the *generalized traveling salesman problem* (GTSP), as already mentioned by Daniels *et al.* (1998). The use of an GTSP algorithm is convenient, but any information about the warehouse layout is not algorithmically exploited. Heßler and Irnich (2024) used a re-implementation of the branch-and-cut algorithm of Fischetti *et al.* (2002) to compare their approach for the 2-SPRP-SS.

4.1.1 Definition of the b-SPRP-SS

We assume that the warehouse layout is given and is that of a multi-block warehouse with parallel aisles. We denote the set of aisles by $J = \{1, 2, \ldots, m\}$ and number the aisles $j \in J$ from left to right. The blocks are numbered from back (=top) to front (=bottom) by $k \in B = \{1, 2, \ldots, b\}$. Each block $k \in B$ in each

aisle $j \in J$ allows picking from the *sub-aisle* (j, k), which is separated by the two cross-aisles k and $k+1 \in K = \{1, 2, \ldots, b, b+1\}$ that lie perpendicular to the aisles, see Figure 4.1. The warehouse is equipped with a depot (I/O point) located at an intersection point of an aisle $j \in J$ and cross-aisle $k \in K$. Articles are stored at the pick positions denoted by $p \in P$ throughout the warehouse. We assume that a *pick list* with articles S is given. As scattered storage is applied, each article may be stored at several pick positions. The pick list specifies the demanded number q_s for each article $s \in S$. We distinguish two cases: If the demand for each article is one, i.e., $q_s = 1$ for all $s \in S$, this is referred to as the *unit-demand* case. In particular, only a single pick position storing the demanded article needs to be visited. In contrast, if the demand for some or all articles of the pick list is greater than one, this is known as the *general-demand* case. In the general-demand case, several pick positions may need to be visited to fulfill the demand of an article. On the one hand, this may be because the demand for an article exceeds the maximum quantity of that article stored at a single pick position. On the other hand, it may also be more efficient for the length of the picker's route to collect the required quantity of an article from several well-located pick positions, rather than taking a long detour to a single pick position with sufficient stock. The b-SPRP-SS seeks a minimum-length picker tour that starts and ends at the depot and collects all articles of the pick list in the demanded quantities. Figure 4.1 shows a small instance with three blocks, four cross-aisles, five aisles, and $10 = |S|$ different articles to be collected. The ten articles $1, 2, \ldots, 10$ are displayed with different colors, whereby one article may be stored at multiple pick positions, e.g., article 1 is stored in aisle $j = 2$ and $j = 4$ of block $k = 2$. An optimal solution for the unit-demand case is also shown.

4.1.2 Contributions

The contributions of this paper can be summarized as follows:

- We present a new network-flow formulation of the b-SPRP-SS. This formulation enforces three key properties of a tour graph, namely that all intersection points between aisles and cross-aisles must have an even vertex degree, that the tour visits sufficiently many pick positions to fulfill demand, and that the tour graph is connected.
- This formulation is solved with a branch-and-cut algorithm, where subtour-elimination constraints are added dynamically to guarantee that the resulting picker tour is connected.
- We compare our new branch-and-cut algorithm with a GTSP solver that is also based on branch-and-cut. We outline the limitations of our approach and also show that it is a superior exact algorithm when warehouses are relatively small, but many scattered articles need to be collected.

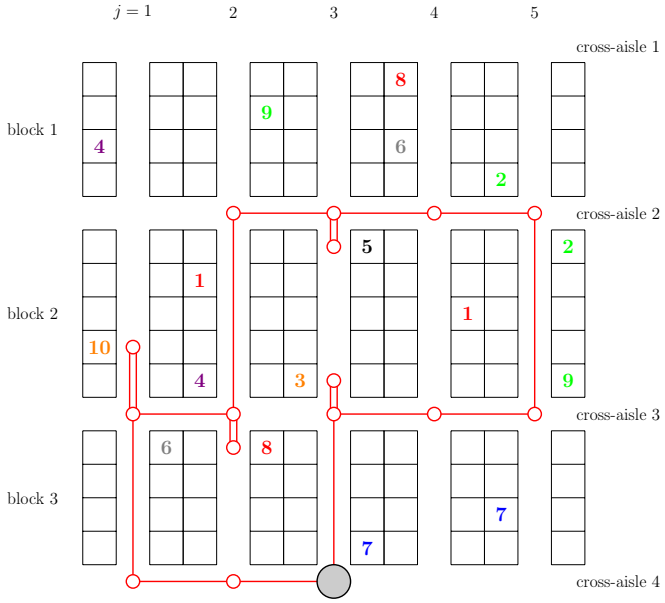


Figure 4.1: Solution of a 3-SPRP-SS instance with four cross-aisles and five aisles.

4.1.3 Structure

Section 4.2 defines the new relaxed state space for the b-SPRP-SS, starting from classical state spaces for SPRP variants, introducing parallel edges to capture options resulting from scattered storage, and highlighting the need for a relaxation when the number of blocks exceeds two. The formulation of the b-SPRP-SS with an exponential class of subtour-elimination constraints is presented in Section 4.3. Section 4.4 presents the associated branch-and-cut algorithm. Computational results are presented and discussed in Section 4.5. The paper closes with final conclusions in Section 4.6.

4.2 The Relaxed State Space for the b-SPRP-SS

Our MIP-based solution approach relies on the construction of a state space, over which a shortest-path problem with additional demand-covering constraints and connectivity constraints is solved. The state space is an extension of Ratliff and Rosenthal or Roodbergen and de Koster (2001b)'s state space, incorporating additional transitions needed to account for scattered storage. In order to make this paper at hand self-contained, we start in Section 4.2.1 with the description of the classical state spaces of Ratliff and Rosenthal and Roodbergen and de Koster (2001b). Subsequently, the additional transitions as introduced by Heßler and Ir-

nich (2024) are presented in Section 4.2.2. The relaxed state space that we use for the b-SPRP-SS is presented in Section 4.2.3.

4.2.1 Classical State Spaces

The DPs of Ratliff and Rosenthal (1983) and Roodbergen and de Koster (2001b) solve the SPRP in warehouses without scattered storage. They construct partial solutions, so-called *partial tour subgraphs* (PTSs), from left to right. We describe the state space as a directed graph (V, E) consisting of a set of vertices V and directed edges E. Each vertex $v \in V$ is a combination of a *stage* and a *state*. The stage denotes the furthest (sub-)aisle (aisles ordered from left to right, blocks from top to bottom) included in the PTS. The state encodes structural information about the PTS.

As the states are of great relevance in our approach, we will discuss them in more detail. The notation of a state consists of two parts. The first part describes the parity of each right-most intersection point (between cross-aisle and sub-aisle) of the PTS. The distinction is made between odd (=uneven) degree, even degree, and a degree of zero, denoted as `U`, `E`, and `0`, respectively. More precisely, the intersection points of an aisle are described one after the other from cross-aisle 1 to cross-aisle $b+1$. This means the first part of a state has always as many symbols as cross-aisles exist.

The second part of the state describes the connected components of the PTS. The state shows how many components it consists of, and (if more than one component exists) also how the components are divided between the blocks, if this information cannot be derived from the parity of the intersection points. Therefore, the cross-aisle $k = 1$ is denoted with `a`, $k = 2$ with `b`, and so forth. If two cross-aisles (or blocks) are disconnected, this is represented with a '$-$' in the second part of the state.

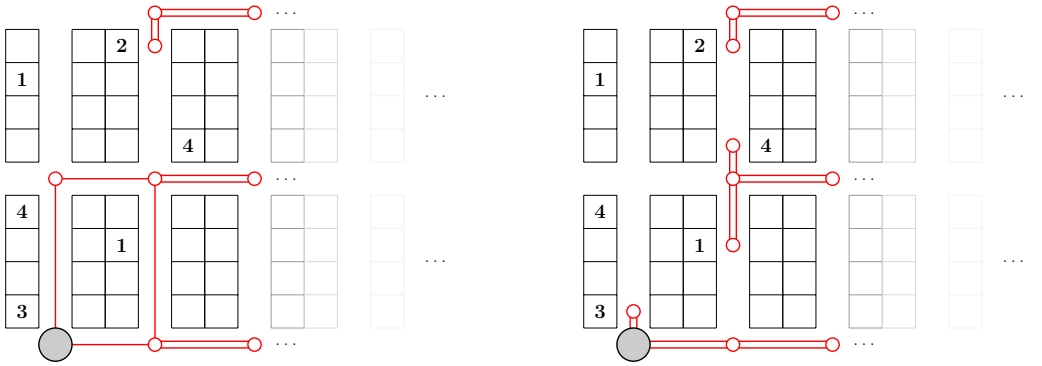
For example, the set of states for a single-block warehouse is defined as

$$\mathscr{S}_1 = \{\texttt{UU.1}, \texttt{0E.1}, \texttt{E0.1}, \texttt{EE.1}, \texttt{EE.2}, \texttt{00.0}, \texttt{00.1}\},$$

(see Ratliff and Rosenthal, 1983), and for a two-block warehouse as

$$\begin{aligned}\mathscr{S}_2 = \{&\texttt{000.0}, \texttt{000.1}, \texttt{E00.1}, \texttt{0E0.1}, \texttt{00E.1}, \texttt{EE0.1}, \texttt{E0E.1}, \texttt{0EE.1}, \texttt{EEE.1}, \texttt{UU0.1}, \texttt{U0U.1},\\ &\texttt{0UU.1}, \texttt{EUU.1}, \texttt{UEU.1}, \texttt{UUE.1}, \texttt{EE0.2}, \texttt{E0E.2}, \texttt{0EE.2}, \texttt{EEE.2a-bc}, \texttt{EEE.2b-ac},\\ &\texttt{EEE.2c-ab}, \texttt{EUU.2}, \texttt{UEU.2}, \texttt{UUE.2}, \texttt{EEE.3}\}\end{aligned}$$

(see Roodbergen and de Koster, 2001b). For more than two blocks, a similar representation is possible, but the number of states increases drastically with the number of blocks in the warehouse.



(a) One component continues through cross-aisle 1 (denoted by `a` in (Roodbergen and de Koster, 2001b)), the other component continues through cross-aisles 2 and 3 (`b` and `c`). This results in the state `EEE.2a-bc` in aisle 3.

(b) Through each cross-aisle leads an unconnected component. The corresponding state is `EEE.3`, respectively.

Figure 4.2: Two PTSs, both with an even degree at all intersection points in aisle 3, but different unconnected components.

Figure 4.2 shows two PTSs constructed up to the third aisle. Neither of the PTSs yet represents a complete tour. All rightmost intersection points have an even degree, leading to `EEE` for the first part of the state. However, there exist different possibilities of (dis)connected cross-aisles or blocks, so the state includes this information. In Figure 4.2a, the state is `EEE.2a-bc`, because the PTS has two unconnected components, where the first component continues through cross-aisle $k = 1$ (`a`) and the second component continues through cross-aisle 2 and 3 (`b` and `c`). Here, it is necessary to provide the information about which cross-aisles are connected, as components cannot be inferred just from the parity of the intersection points. In Figure 4.2b, the state is `EEE.3`, because the PTS has three unconnected components that must be finally connected using later sub-aisle actions.

The state space includes a designated origin vertex o and a destination vertex d. In the DP of Ratliff and Rosenthal (1983), these are $o =$ `00.0` at the first stage 1 and $d =$ `00.1` at the (artificial) last stage $2m + 1$. Likewise, in the DP of Roodbergen and de Koster (2001b), these are $o =$ `000.0` at the first stage 1 and $d =$ `000.1` at the last stage $3m + 1$.

The set of edges is defined as

$$E = \bigcup_{j \in J} E_j^{cross} \cup \bigcup_{(j,k) \in J \times B} E_{jk}^{aisle}.$$

The sets contain all possible traversing options (=actions) of cross-aisles between aisle j and $j+1$, as well as of (sub-)aisles $(j,k) \in J \times B$, that can lead to the optimal solution. We describe these sets in more detail now.

Regarding the sets E_j^{cross} of cross-aisle actions, it is important to see that they depend on the number of blocks. For the one-block case, Ratliff and Rosenthal (1983) define these actions as `00`, `11`, `02`, `20`, and `22` where the first (second) symbol indicates how often the front (back) cross-aisle is traversed. For the two-block case, Roodbergen and de Koster (2001b) define cross-aisle actions `000`, `110`, `101`, `011`, `200`, ..., `211`, ..., `222` where the first (second, third) symbol indicates how often the front (middle, back) cross-aisle is traversed. For more than two blocks, cross-aisle actions can either be modeled as $b+1$-dimensional vectors with an even number of entries `1` (referring to the cross-aisles from front to back; as we do in the next section) or cross-aisle actions can be broken down into decisions per sub-aisle (as in Pansart *et al.*, 2018). Finally, for the rightmost aisle $j = m$, the set of cross-aisle actions E_m^{cross} comprises only `0` movements, since the respective vertices of the state space are hereby connected to the unique sink vertex.

Regarding the sets E_{jk}^{aisle}, aisle actions can be `1pass`, `2pass`, `top`, `bottom`, `gap`, and `void`. If no pick position is located in the sub-aisle (j,k), then $E_{jk}^{aisle} = \{\texttt{1pass}, \texttt{2pass}, \texttt{void}\}$ where the actions `1pass` and `2pass` describe the complete traversal of a sub-aisle once or twice, respectively. (Revenant *et al.* (2025) have shown `2pass` is redundant in the single-block case, but otherwise indispensable to ensure optimality.) If the picker enters a sub-aisle from the top, collects all demanded articles in the sub-aisle, and then leaves the sub-aisle again at the top, this is called a `top` action. When the action is mirrored and the sub-aisle is entered from the bottom, it is called `bottom`. The action `gap` is the combination of `top` and `bottom` in the same sub-aisle, where the largest part of the sub-aisle with no demanded articles stored is not visited. Accordingly, sub-aisles (j,k) with a pick position have the complete set of aisle actions given by $E_{jk}^{aisle} = \{\texttt{1pass}, \texttt{2pass}, \texttt{top}, \texttt{bottom}, \texttt{gap}, \texttt{void}\}$.

For a visualization of the state spaces of Ratliff and Rosenthal (1983) and Roodbergen and de Koster (2001b), we refer to (Heßler and Irnich, 2024, Figures 2 and EC.2).

4.2.2 Additional Transitions for the SPRP-SS

If the articles may be stored at more than one pick position, the just described classical state space (Section 4.2.1) needs to be adapted as described by Heßler and Irnich (2024). The set V of vertices of the extended state space for the SPRP-SS remains identical to the vertex set of the classical state space for the SPRP. In contrast, the set E of edges of the extended state space is significantly expanded. Specifically, the cross-aisle actions remain unchanged compared to the classical state space, while the number of aisle actions increases considerably.

In the SPRP, each demanded article is stored at a unique pick position in the warehouse. Consequently, the turning points of the actions `top`, `bottom`, and `gap` are clearly determined for each aisle. This is no longer the case for the SPRP-SS, as an article may be picked from alternative pick positions in the warehouse. More precisely, the turning point of an aisle action is no longer unique for the SPRP-SS. In the unit-demand case, it depends on the decision from which specific pick position a certain article is collected, as this also determines which pick positions of the article are irrelevant for the tour. In the general-demand case, it is also possible that the demanded quantity of an article is collected from different pick positions. This means that a subset of all the article's pick positions needs to be visited during the picker tour. Therefore, Heßler and Irnich (2024) introduced parallel aisle actions of the same type in one aisle, differing only in their respective turning points and the resulting cost of the action.

4.2.3 Relaxed State Space for the b-SPRP-SS

Independent of the number b of blocks, a feasible picker tour in the SPRP-SS can be described by the following properties: The tour

(A) imposes an even vertex degree at all intersection points between aisles and cross-aisles,
(B) visits one or sufficiently many pick position(s) for each article $s \in S$, and
(C) makes the tour graph connected.

By construction, each feasible o-d-path over the state space ensures (A). Property (B), i.e., demand coverage, has been guaranteed by Heßler and Irnich (2024) through the introduction of demand-covering constraints in the MIP. We will use the same idea in Section 4.3.

The overarching idea of the paper at hand is to relax property (C) by constructing a *relaxed state space*. In this relaxed state space, we omit the connectivity information of the states, i.e., we simply drop the second part of each state that describes the components of the PTS. Hence, the states only specify the parity of each intersection point of the considered aisle. Accordingly, all states with a common first part of the original state space merge to a single state in the

relaxed state space. For example, in a two-block warehouse, the states `EEE.1c`, `EEE.2a-bc`, `EEE.2b-ac`, `EEE.2c-ab`, and `EEE.3` merge into the new state `EEE` in the relaxed state space. Any state is given by a vector of $b+1$ entries of $\{\texttt{0}, \texttt{U}, \texttt{E}\}$ with an even number of `U` entries. It is straightforward to see that the relaxed state space has only $N(b)$ different states, where the function N follows the recursion $N(b) = 2N(b-1) + bN(b-2)$ with the initialization $N(1) = 5$ and $N(2) = 14$.

The advantage of the relaxed state space is that the number of states per stage grows moderately. Table 4.1 compares the number of states of the relaxed state space with the original DP of Ratliff and Rosenthal (1983) for single-block warehouses, the DP for two-block warehouses by Roodbergen and de Koster (2001a), and the DP of Pansart *et al.* (2018) for multi-block warehouses. For the SPRP-SS in warehouses with only one or two blocks ($b \leq 2$), $N(b)$ does not differ strongly from the number of states of the above mentioned DPs. Thus, the known approach of Heßler and Irnich (2022b), improved by (Lüke *et al.*, 2026), can be expected to be fast and effective. Therefore, we will focus on true multi-block warehouses with $b \geq 3$ in the following. For $b \geq 3$, the difference between $N(b)$ and the number of states in the DP of Pansart *et al.* (2018) reaches a factor of four. However, even if the difference becomes more pronounced for $b \geq 5$ (factor six and larger), the absolute number of states $N(b)$ becomes prohibitively large so that we cannot solve SPRP-SS instances of these block sizes either. Note that the state spaces have approximately $(b+1)m \cdot N(b)$ vertices.

Number b of blocks	1	2	3	4	5	6	7	8	9	10
Number of states per stage										
in DP of Ratliff and Rosenthal	7									
in DP of Roodbergen and de Koster		25								
in DP of Pansart *et al.*	6	24	112	568	3032	16,768	95,200	551,616	3,248,704	19,389,824
in our relaxed DP	5	14	43	142	499	1850	7193	29,186	123,109	538,078

Table 4.1: Comparison of the number of states in warehouses with b blocks of different DPs.

In the relaxed state space, two vertices $v, w \in V$ refer to consecutive stages and an action (cross-aisle action or sub-aisle action) translates v into w. The rules to describe feasible actions are identical to those that define the DPs of Ratliff and Rosenthal; Roodbergen and de Koster. For the sake of brevity, we do not formalize the rules here but we give two examples: For a warehouse with $b = 5$ blocks, the sub-aisle action `gap` in sub-aisle $(j, 2)$ translates the state `EUOUOE` into `EUEUOE` (positions referring to the second block are underlined). The cross-aisle action `210100` translates the same state `EUOUOE` into `EUOUOO`.

On the downside, Property (C) (connectivity) is no longer fulfilled for o-d-path over the relaxed state space. We will finally ensure connectivity by adding subtour-

elimination constraints when the MIP formulation is solved with a *branch-and-cut* (B&C) algorithm (see Sections 4.3 and 4.4).

4.3 Network-Flow Formulation of the b-SPRP-SS

Recall that (V, E) describes the relaxed state space introduced in Section 4.2.3. For each edge $e \in E$, let c_e be the length of the part of the tour related to the action e. Additionally, let b_{se} be the number of articles $s \in S$ that can be collected when performing action e. In particular, $b_{se} = 0$ holds for all $s \in S$ and $e \in E_j^{cross}$ as well as $e \in E^{aisle}$ describing the action `void`. We denote by $e \in E_s$ all edges that correspond to actions collecting article $s \in S$. In a feasible instance, $\sum_{e \in E_s} b_{se} \geq q_s$ holds for all $s \in S$. Let $e \in \delta^+(v)$ and $e \in \delta^-(v)$ denote the sets of outgoing and incoming edges of vertex $v \in V$, respectively. The *network-flow formulation* (NF) of the b-SPRP-SS that we will use has binary variables x_e for all $e \in E$ and reads as follows:

$$z_{b\text{-SPRP-SS}} = \min \sum_{e \in E} c_e x_e \tag{4.1a}$$

$$\text{subject to} \quad \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = \begin{cases} +1, & \text{if } v = o \\ -1, & \text{if } v = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in V \tag{4.1b}$$

$$\sum_{e \in E_s} b_{se} x_e \geq q_s \qquad \forall s \in S \tag{4.1c}$$

$$\{\text{subtour-elimination constraints}\} \tag{4.1d}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4.1e}$$

The objective function (5.1a) minimizes the total length of the picker tour. Flow conservation through the relaxed state space is enforced by constraints (5.1b). Constraints (5.1c) guarantee that the demand for each article on the pick list is fully met. The *subtour-elimination constraints* (SECs) (4.1d) ensure that the resulting picker tour is connected. Note that we intentionally do not use the term *connectivity constraints*, since the support graph that we will introduce in Section 4.4.1 does not need to be connected. However, subtours are clearly infeasible. Finally, the domain restrictions of the decision variables are specified in (5.1d).

Some remarks about the structure of formulation (5.1) are due: For the SPRP (no scattering), the equivalence of DP and linear programming implies that the SPRP can be solved solely with (5.1a), (5.1b), and (5.1d) relaxed to $x_e \geq 0$ for all $e \in E$. The approach of Heßler and Irnich (2024) for the SPRP-SS in parallel-aisle warehouses with a single or two blocks uses NF without SECs, i.e., (5.1a), (5.1b),

(5.1c), and (5.1d).

We can also estimate the size of NF. Since the number of vertices of the relaxed state space is proportional to $N(b) \cdot mb$, the same is true for the number of edges. Therefore, formulation (5.1) has $\mathcal{O}(N(b) \cdot mb)$ variables. There are $|V|$ flow-conservation constraints (5.1b) and $|S|$ demand-covering constraints (5.1c). Since the number of connectivity constraints (4.1d) is exponential in the warehouse size, we will use a B&C algorithm described next to solve the model.

4.4 Branch-and-Cut Algorithm for the b-SPRP-SS

In this section, we formally introduce the SECs to be used in formulation (5.1). A major complication stems from the fact that already testing whether a tentative solution computed over the relaxed state space imposes a connected picker tour (or not) is not trivial. To this end, we transform tentative solutions into flow values in a support graph. This support graph reflects the movements of the picker in the warehouse, and it is much smaller than the relaxed state space. Therefore, feasibility checking and the separation of violated SECs can be performed efficiently. Indeed, all separation procedures reduce to max flow/min cut computation on the support graph.

We assume that a partial formulation of (5.1) is solved, i.e., a model consisting of (5.1a), (5.1b), (5.1c), and (5.1d) with no or just a subset of the SECs (4.1d). Let $\bar{x} = (\bar{x}_e)_{e \in E}$ be the optimal solution of this partial formulation.

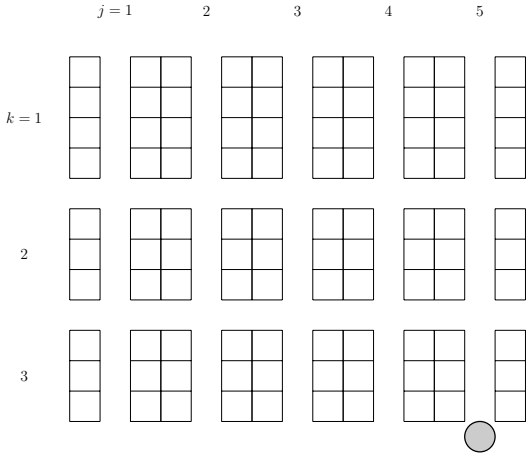
4.4.1 Warehouse Graph and Support Graph

We motivate the warehouse graph and support graph with the help of an example. Figure 4.3a shows the layout of the underlying warehouse of a small 3-SPRP-SS instance with five aisles, four cross-aisles, and three blocks. Moreover, Figure 4.3c shows an instance of the 3-SPRP-SS for this warehouse with a picker tour that is disconnected and consists of five subtours. We want the warehouse graph to represent the warehouse structure with the possible movements of a picker and the support graph to capture the picker tour with possible subtours. Hence, the set W of vertices of the warehouse graph represents all intersection points between aisles and cross-aisles, i.e.,

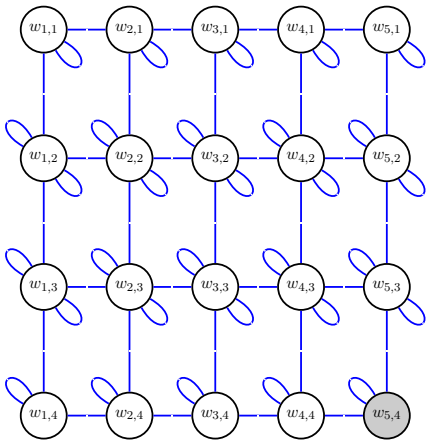
$$W = \{w_{jk} : j \in J, k \in K\}$$

(recall that $b+1 \in K$ but $b+1 \notin B$).

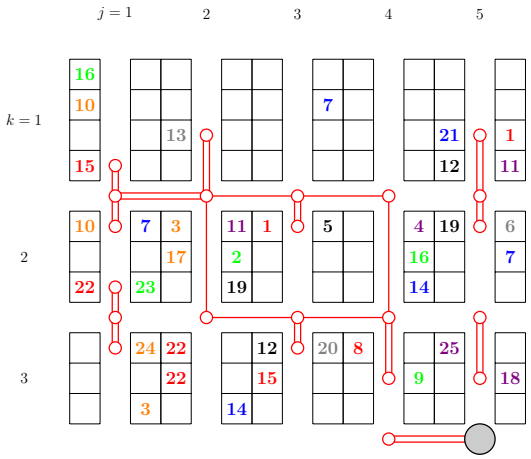
The warehouse graph $(W, \mathcal{E}, \bar{f})$ is an undirected, edge-weighted multi-graph with loops. It includes four types of edges, defining the set $\mathcal{E}$ as visualized in Figure 4.3b.



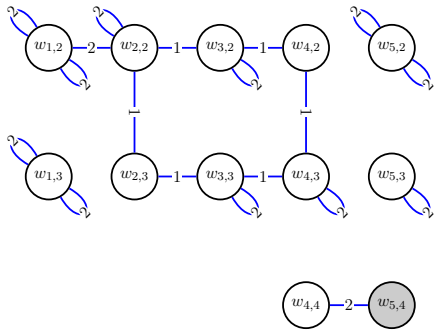
(a) Warehouse layout with $b = 3$ blocks and $m = 5$ aisles.



(b) Warehouse graph $(W, \mathcal{E})$ without flow values.



(c) An infeasible integer solution of a unit-demand 3-SPRP-SS instance during the B&C procedure.



(d) Support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$ of the solution with flow values.

Figure 4.3: Infeasible tentative, integer solution of a 3-SPRP-SS instance.

The first two types describe all possible 'horizontal' and 'vertical' connections between the intersection points, which are defined by

$$\mathcal{E}^{\text{hor}} = \{(w_{jk}, w_{j+1,k}) : j \in J, j \neq m, k \in K\}$$

and

$$\mathcal{E}^{\text{vert}} = \{(w_{jk}, w_{j,k+1}) : j \in J, k \in B\}.$$

For both types of edges, we define *flow values*. The flow values on the horizontal connections between the vertices $w_{j,k}$ and $w_{j+1,k} \in W$ indicate the number of the traversals of the cross-aisle k between the aisles j and $j+1$. We assume that $cross_{ke}$ indicates how often cross-aisle k between aisle j and $j+1$ is traversed by edge $e \in E_j^{cross}$ of the relaxed state space. Hence, the flow value on edge $(w_{jk}, w_{j+1,k}) \in \mathcal{E}^{\text{hor}}$ is defined as

$$\bar{f}_{w_{jk},w_{j+1,k}} = \sum_{\substack{e \in E_j^{cross}: \\ cross_{ke}=1}} \bar{x}_e + \sum_{\substack{e \in E_j^{cross}: \\ cross_{ke}=2}} 2\bar{x}_e. \tag{4.2a}$$

The flow values on the vertical connections consider all actions that *completely* traverse an aisle, which includes only aisle actions `1pass` and `2pass`. We refer to this subset of actions as *traversal*-aisle actions and denote by $E_{jk}^{\texttt{1pass}}$ ($E_{jk}^{\texttt{2pass}}$) the set of edges of aisle j and block k that represents the action `1pass` (`2pass`). The flow value on edge $(w_{jk}, w_{j,k+1}) \in \mathcal{E}^{\text{vert}}$ is given by

$$\bar{f}_{w_{jk},w_{j,k+1}} = \sum_{e \in E_{jk}^{\texttt{1pass}}} \bar{x}_e + \sum_{e \in E_{jk}^{\texttt{2pass}}} 2\bar{x}_e. \tag{4.2b}$$

The solution shown in Figure 4.3c includes a small subtour that connects the depot in aisle $j = 5$ with aisle $j = 4$, where the corresponding segment of the cross-aisle is traversed twice. In the warehouse graph in Figure 4.3d, this subtour is represented by the edge $(w_{4,4}, w_{5,4})$ with the flow value 2.

However, up to now, with only these first two types of edges, we are not able to detect the two subtours in aisle $j = 5$ and the short subtour in aisle $j = 1$ depicted in Figure 4.3c. The reason is that aisle actions with movements in an aisle not traversing a sub-aisle completely are not yet transferred to the warehouse graph. To this end, we consider the set of *return*-aisle actions, which includes the actions `top`, `bottom`, and `gap`. These aisle actions are represented in the warehouse graph as possibly parallel loops, i.e., edges with both endpoints referring to the same vertex. We further distinguish whether a sub-aisle is entered from above or from below. Action `top` enters the sub-aisle of aisle j and block k from the above using cross-aisle k. However, action `bottom` enters the same sub-aisle from below using cross-aisle $k+1$. Action `gap` enters a sub-aisle both from above and from below.

Accordingly, the third and fourth types of edges are defined as follows: For each vertex $w_{jk} \in W$ with $j \in J$ and $k \in B$, we introduce a loop indicating that sub-aisle (j,k) is entered (*not* traversed completely) via cross-aisle k from above. Likewise, for each vertex $w_{jk} \in W$ with $j \in J$ and $k \in K, k > 1$, we introduce a loop indicating that sub-aisle $(j, k-1)$ is entered by cross-aisle k from below, but not traversed completely.

To lighten the notation, we refer to the loops of w_{jk} as $(w_{jk})^{\downarrow}$ and $(w_{jk})^{\uparrow}$ (instead of $(w_{jk}, w_{jk})^{\uparrow}$ and $(w_{jk}, w_{jk})^{\downarrow}$), respectively. Accordingly, we define

$$\mathcal{E}^{\downarrow} = \{(w_{jk})^{\downarrow} : j \in J, k \in B\} \qquad \text{and} \qquad \mathcal{E}^{\uparrow} = \{(w_{jk})^{\uparrow} : j \in J, k \in K, k \neq 1\}.$$

Thus, the complete set of edges is $\mathcal{E} = \mathcal{E}^{\text{hor}} \cup \mathcal{E}^{\text{vert}} \cup \mathcal{E}^{\downarrow} \cup \mathcal{E}^{\uparrow}$. The flow values of the above loops are given by

$$\bar{f}_{w_{jk}^{\downarrow}} = \sum_{e \in E_{jk}^{\texttt{top}}} 2\bar{x}_e + \sum_{e \in E_{jk}^{\texttt{gap}}} 2\bar{x}_e \qquad \text{and} \qquad \bar{f}_{w_{jk}^{\uparrow}} = \sum_{e \in E_{j,k-1}^{\texttt{bottom}}} 2\bar{x}_e + \sum_{e \in E_{j,k-1}^{\texttt{gap}}} 2\bar{x}_e, \quad (4.2c)$$

where $E_{jk}^{\texttt{top}}$, $E_{jk}^{\texttt{bottom}}$, and $E_{jk}^{\texttt{gap}}$ denote the sets of edges of aisle j and block k that represent the return-aisle actions `top`, `bottom`, and `gap`, respectively. For convenience, we also define the set of all collection-aisle actions as $E_{jk}^{\texttt{collect}} = E_{jk}^{\texttt{1pass}} \cup E_{jk}^{\texttt{2pass}} \cup E_{jk}^{\texttt{top}} \cup E_{jk}^{\texttt{bottom}} \cup E_{jk}^{\texttt{gap}}$.

In the example in Figure 4.3c, the first block of the last aisle $j = 5$ is entered from below, while the second and the third block are entered from above by two different subtours. The flow values of the corresponding three loops emanating from the vertices $w_{5,2}$ and $w_{5,3} \in W$ are shown with a value of 2 in Figure 4.3d.

Finally, let a possibly fractional solution $\bar{x} = (\bar{x}_e)_{e \in E}$ with flow values $\bar{f} = (\bar{f}_\epsilon)_{\epsilon \in \mathcal{E}}$ defined by (4.2) be given. We define the support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$ as the graph of the warehouse graph spanned by the edges with positive flow. Formally, the edge set is $\mathcal{E}_{\bar{f}} = \{\epsilon \in \mathcal{E} : \bar{f}_\epsilon > 0\}$ and the vertex set is the span $W_{\bar{f}} = W(\mathcal{E}_{\bar{f}})$. In Figure 4.3d, the support graph contains only the twelve vertices w_{jk} of the warehouse graph that are incident to edges with a positive flow value.

4.4.2 Subtour-Elimination Constraints

Different types of SECs are needed to exclude all possible subtours. Figure 4.4 provides an overview of the different SEC types and their requirements. All SECs have in common that they are defined for a subset $\mathcal{S} \subset W$ of the vertices of the warehouse graph. For any subset $\mathcal{S}$, the set $\delta(\mathcal{S})$ consists of all edges of $\mathcal{E}$ with one endpoint in $\mathcal{S}$ and the other endpoint in $\bar{\mathcal{S}} := W \setminus \mathcal{S}$. We refer to $\delta(\mathcal{S}) = \delta(\bar{\mathcal{S}})$ as the *cut set* of $\mathcal{S}$. By definition, loops are never contained in a cut set. We use equations (4.2a), (4.2b), and (4.2c) to state the SECs in terms of flows on the edges of the support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$. Note that we can transform a condition on the flow of the support graph back into the relaxed state space using these equations. Formally, we replace $\bar{f}$ by f and $\bar{x}$ by x.

For each article $s \in S$, we define the set $\mathcal{S}_s$ as

$$\mathcal{S}_s = \bigcup_{\substack{(j,k)\in J\times B:\\ \text{sub-aisle } (j,k)\\ \text{contains article } s}} \{w_{jk}, w_{j,k+1}\}.$$

This set includes all intersection points at the end of a sub-aisle that allows the collection of the article s.

Next, we further characterize subsets of the warehouse graph.

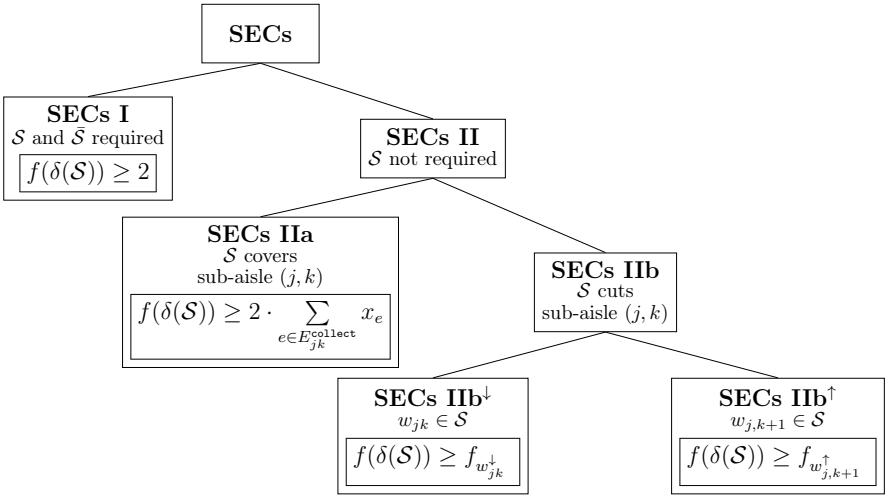


Figure 4.4: Overview of the different types of SECs.

Definition 1. *Let $\mathcal{S} \subseteq W$ be an arbitrary subset of vertices of the support graph. Then,*

(i) $\mathcal{S}$ includes the depot*, if $w_{j_0,k_0} \in \mathcal{S}$ and the depot 0 is located at the intersection (j_0, k_0);*

(ii) $\mathcal{S}$ cuts *a sub-aisle $(j,k) \in J \times B$, if $(w_{jk}, w_{j,k+1}) \in \delta(\mathcal{S})$;*

(iii) $\mathcal{S}$ covers *a sub-aisle $(j,k) \in J \times B$, if $w_{jk} \in \mathcal{S}$ and $w_{j,k+1} \in \mathcal{S}$;*

(iv) In the unit-demand case, $\mathcal{S}$ is required*, if $\mathcal{S}_s \subseteq \mathcal{S}$ holds for at least one article $s \in S$, or $\mathcal{S}$ includes the depot;*

(v) In the general-demand case, $\mathcal{S}$ is required*, if for at least one article $s \in S$, all sub-aisles (j,k)* not *covered by $\mathcal{S}$ have an accumulated supply strictly smaller than the demand q_s, or $\mathcal{S}$ includes the depot.*

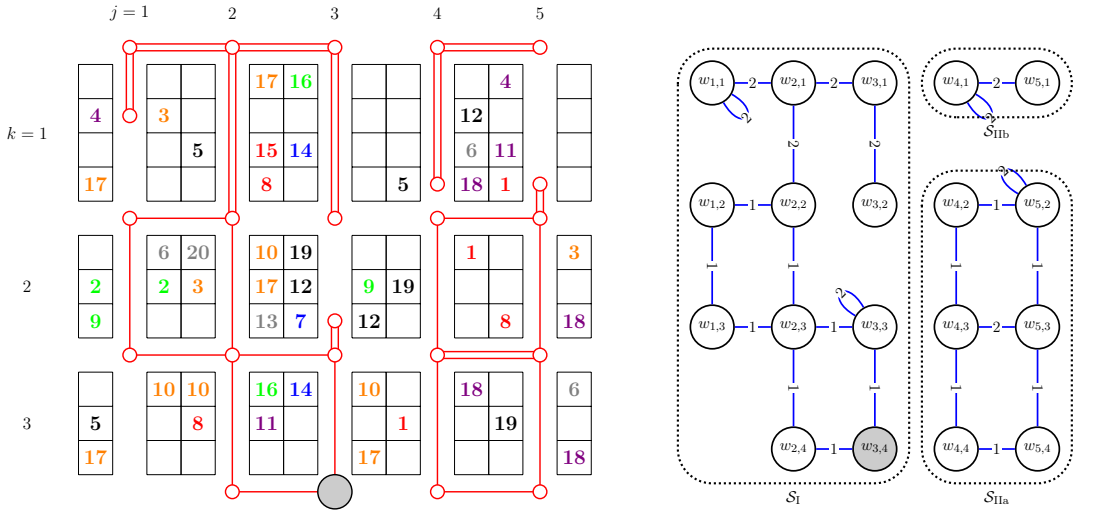
Note that either $\mathcal{S}$ or its complement $\bar{\mathcal{S}} = W \setminus \mathcal{S}$ or both are required because of the depot condition. Any non-required subset $\mathcal{S}$ must have a cut set $\delta(\mathcal{S})$ that necessarily separates the depot from the vertices in $\mathcal{S}$. This shows that the scheme

presented in Figure 4.4 applies to all proper, non-empty subsets $\mathcal{S}$ or $\bar{\mathcal{S}}$, taking into account that $f(\delta(\mathcal{S})) = f(\delta(\bar{\mathcal{S}}))$ holds (we use the shorthand notation $f(\mathcal{E}')$ for the sum $\sum_{(w,w')\in\mathcal{E}'} f_{ww'}$ of an arbitrary subset $\mathcal{E}' \subseteq \mathcal{E}$).

SECs of Type I If both $\mathcal{S}$ and $\bar{\mathcal{S}}$ are required, the picker tour must necessarily traverse the cut set $\delta(\mathcal{S})$ twice. This implies

$$f(\delta(\mathcal{S})) \geq 2. \tag{4.3a}$$

Figure 4.5 shows an example where the subset $\mathcal{S}_{\mathrm{I}} = \{w_{jk} : 1 \leq j \leq 3, 1 \leq k \leq 4\}$ includes the depot located at $w_{3,4}$. Hence, $\mathcal{S}_{\mathrm{I}}$ is required. Its complement $\bar{\mathcal{S}}_{\mathrm{I}} = \{w_{jk} : 4 \leq j \leq 5, 1 \leq k \leq 4\}$ is also required, since article $s = 18$ is only available in sub-aisles covered by $\bar{\mathcal{S}}_{\mathrm{I}}$. Therefore, the SEC (4.3a) of Type I is valid and violated for the subset $\mathcal{S}_{\mathrm{I}}$ (equivalently for $\bar{\mathcal{S}}_{\mathrm{I}}$).



(a) An infeasible, tentative 3-SPRP-SS solution.

(b) The corresponding support graph $(W, \mathcal{E})$ with flow values and subsets indicating violated SECs.

Figure 4.5: Different types of violated SECs for an infeasible, tentative solution of a unit-demand 3-SPRP-SS instance.

SECs of Type IIa If $\mathcal{S}$ is non-required and covers a sub-aisle (j, k), the cut set $\delta(\mathcal{S})$ separates the depot (in $\bar{\mathcal{S}}$) from that sub-aisle. A positive value of $f_{w_{jk},w_{j,k+1}}$ indicates that the sub-aisle is completely traversed. This implies

$f(\delta(\mathcal{S})) \geq f_{w_{jk},w_{j,k+1}}$. Note that the right-hand side $f_{w_{jk},w_{j,k+1}}$ can be a value between 0 and 2 due to the definition of the flow values in (4.2b). Similarly, positive values of $f^{\downarrow}_{w_{j,k}}$ or $f^{\uparrow}_{w_{j,k+1}}$ indicate that the sub-aisle (j,k) is at least entered from the upper or lower end, implying $f(\delta(\mathcal{S})) \geq f^{\downarrow}_{w_{j,k}}$ and $f(\delta(\mathcal{S})) \geq f^{\uparrow}_{w_{j,k+1}}$.

The three inequalities can be aggregated into

$$f(\delta(\mathcal{S})) \geq 2 \cdot \sum_{e \in E^{\texttt{collect}}_{jk}} x_e. \tag{4.3b}$$

Inequality (4.3b) has the advantage over the three inequalities with a f-value on the right hand side that the factor of 2 can be added. This results from the fact that the sum on the right-hand side assumes values between 0 and 1 compared to $f_{w_{jk},w_{j,k+1}}$, $f^{\downarrow}_{w_{j,k}}$, and $f^{\uparrow}_{w_{j,k+1}}$ which can assume values between 0 and 2. Indeed, inequality (4.3b) dominates the three above inequalities.

Figure 4.5 shows another example for the subset $\mathcal{S}_{\text{IIa}} = \{w_{jk} : 4 \leq j \leq 5, 2 \leq k \leq 4\}$. This subset $\mathcal{S}_{\text{IIa}}$ does not include the depot. Moreover, the four sub-aisles included in $\mathcal{S}_{\text{IIa}}$ allow to collect the articles $S' = \{1, 3, 6, 8, 18, 19\}$. All these $s \in S'$ are also collectable from other sub-aisles that are not included in $\mathcal{S}_{\text{IIa}}$. As a result, $\mathcal{S}_{\text{IIa}}$ is not required. Since the sub-aisles $(4,2)$, $(4,3)$, $(5,2)$, and $(5,3)$ are covered by $\mathcal{S}_{\text{IIa}}$, the SEC (4.3b) of Type IIa is valid and also violated for these sub-aisles (j,k).

SECs of Type IIb If $\mathcal{S}$ is not required and $\mathcal{S}$ cuts a sub-aisle (j,k), the cut set $\delta(\mathcal{S})$ separates the depot (in $\bar{\mathcal{S}}$) either from the upper part of that sub-aisle or from the lower part of that sub-aisle. If it separates the upper part, this implies

$$f(\delta(\mathcal{S})) \geq f^{\downarrow}_{w_{jk}} \tag{4.3c}$$

which is a SEC of Type IIb$^{\downarrow}$. If it separates the lower part, this implies

$$f(\delta(\mathcal{S})) \geq f^{\uparrow}_{w_{j,k+1}} \tag{4.3d}$$

which is a SEC of Type IIb$^{\uparrow}$.

The subset $\mathcal{S}_{\text{IIb}} = \{w_{4,1}, w_{5,1}\}$ highlighted in Figure 4.5b does not contain the depot and does not cover any sub-aisle. Therefore, it is not required. For $(j,k) = (4,1)$ or $(j,k) = (5,1)$, the respective sub-aisle is cut by $\mathcal{S}$. Only for $(j,k) = (4,1)$, the SEC (4.3c) of Type IIb$^{\downarrow}$ is violated for $\mathcal{S}_{\text{IIb}}$. It is important to mention that a subset $\mathcal{S}$ of Type IIb with a positive flow inside the subset without a violated SEC (4.3c) of Type IIb$^{\downarrow}$ and SEC (4.3d) of Type IIb$^{\uparrow}$ is possible. For example, any subtour exclusively traversing cross aisles without any connected aisle action is feasible. However, these solutions do not occur in feasible LP solutions, since

they are dominated.

Note that the subset $\mathcal{S}_{\text{IIa}}$ can serve as an example of a violated SEC (4.3d) of Type IIb$^{\uparrow}$ with $(j, k) = (5, 2)$.

4.4.3 Separation of Subtour-Elimination Constraints

The separation of violated SECs in the B&C algorithm distinguishes between integer (but possibly infeasible) solutions and fractional solutions. We describe these cases in separate paragraphs now.

Integer Solutions Let an integer solution be given by $\bar{x} = (\bar{x}_e)_{e \in E}$ with corresponding flow values $\bar{f} = (\bar{f}_\epsilon)_{\epsilon \in \mathcal{E}}$ computed via equations (4.2a)–(4.2c). It suffices to inspect the components of the support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$. A union-find algorithm can be used to efficiently determine components, we use the one by Tarjan (1975). If the support graph has two or more components, subtours exist. For a component given by $\mathcal{S} \subset W_{\bar{f}}$, we have $\bar{f}(\delta(\mathcal{S})) = 0$ such that either the SEC (4.3a) of Type I is violated (if both $\mathcal{S}$ and $\bar{\mathcal{S}}$ are required), or the SEC (4.3b) of Type IIa is violated (if $\mathcal{S}$ covers a sub-aisle), or the SEC (4.3c) or (4.3d) is violated (if $\mathcal{S}$ does not cover any sub-aisle).

Fractional Solutions Let a fractional solution be given by $\bar{x} = (\bar{x}_e)_{e \in E}$ with flow values $\bar{f} = (\bar{f}_\epsilon)_{\epsilon \in \mathcal{E}}$. If the support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$ has two or more components, we use the same procedure as described for integer solutions. Otherwise, the support graph consists of a single component, and each type of SEC requires its own separation algorithm. However, they all exploit that the left-hand side $f(\delta(\mathcal{S}))$ describes the flow over a cut set $\delta(\mathcal{S})$, so that a max flow/min cut algorithm is applicable (Ahuja *et al.*, 1993, Chapters. 6–8). Depending on the type of SEC, we will determine minimum cuts in which the subsets $\mathcal{S}$ and/or $\bar{\mathcal{S}}$ must include certain subsets $X \subset W$ or $Y \subset W$ of vertices. These problems can be solved efficiently by using a standard s-t-max flow/min cut algorithm, either by merging the vertices of X and of Y, respectively, creating a new, slightly smaller support graph, or by inserting additional internal edges connecting vertices of X and Y, respectively, with a high cost in a preparatory step. (We apply the latter technique.) We will use the term *X-Y-max flow/min cut* for such a problem. For all types of SECs, the maximum flow/minimum cut value must be compared with the right-hand side of (4.3a)–(4.3d) depending on the type.

SECs of Type I We start with the simple case of unit demands. Recall from Definition 1(iv) that here the subsets $\mathcal{S}$ and $\bar{\mathcal{S}}$ must be required, i.e., contain the depot or a subset $\mathcal{S}_s$ of at least one article $s \in S$. Assuming that the depot 0

is located at the intersection $w_0 = (j_0, k_0)$, we solve a sequence of w_0-$\mathcal{S}_s$-max flow/min cut problems, one for each $s \in S$. Such a minimum cut has the property that the depot w_0 is contained in $\mathcal{S}$ and $\mathcal{S}_s \subseteq \bar{\mathcal{S}}$. This implies that both $\mathcal{S}$ and $\bar{\mathcal{S}}$ are required. If the flow value of the cut is smaller than 2, a violated SEC (4.3a) of Type I is found. The procedure can be made (slightly) more efficient by preprocessing the subsets $\mathcal{S}_s, s \in S$, so that only inclusion-minimal subsets are taken into account.

The case of general demand is more involved. Here, a subset $\mathcal{S}$ that does not include the depot is certainly required, if $\mathcal{S}_s \subseteq \mathcal{S}$ holds for an article $s \in S$. However, recall from Definition 1(v) that smaller subsets of $\mathcal{S}_s$ may suffice to obtain a required subset. Such smaller subsets contain some but not necessarily all sub-aisles that allow the collection of the article $s \in S$. An exact procedure would have to construct exactly all these smaller subsets and solve max-flow/min cut problems for these. To reduce the computational burden in the general-demand case, we use a heuristic and only test cuts that result from the w_0-$\mathcal{S}_s$-max flow/min cut problems for all $s \in S$, and cuts separating the depot from in from a sub-aisle (for all sub-aisles $(j, k) \in J \times B$; here the resulting sets $\mathcal{S}$ and $\bar{\mathcal{S}}$ do not necessarily give SECs of Type I). Note that for the correctness of the overall B&C algorithm, it is not necessary to identify violated SECs in fractional solutions. The use of separation heuristics as the one outlined above trades computational effort against lower bound improvement.

SECs of Type IIa For SECs (4.3b) of Type IIa, the subset $\mathcal{S}$ must not be required. However, it must contain a sub-aisle (j, k), i.e., $\{w_{jk}, w_{j,k+1}\} \subseteq \mathcal{S}$. Therefore, the separation routine considers all sub-aisles $(j, k) \in J \times B$, and solves for each of them the $\{w_{jk}, w_{j,k+1}\}$-w_0-max flow/min cut problems on the support graph. The sets $\mathcal{S}$ and $\bar{\mathcal{S}}$ have the properties that the depot w_0 is contained in $\bar{\mathcal{S}}$ and that $\mathcal{S}$ contains the sub-aisle (j, k). The flow value is then compared against $2 \cdot \sum_{e \in E^{\texttt{collect}}_{jk}} \bar{x}_e$. If the flow value is smaller, a violated SEC (4.3b) of Type IIa is found for $\mathcal{S}$ and sub-aisle (j, k).

SECs of Type IIb For SECs (4.3d) and (4.3c) of Type IIb, the subset $\mathcal{S}$ must not be required and must cut a sub-aisle (j, k), i.e., $(w_{jk}, w_{j,k+1}) \in \delta(\mathcal{S})$. Therefore, the separation routine considers all sub-aisles $(j, k) \in J \times B$, and solves for each of them the $\{w_{jk}, w_{j,k+1}\}$-w_0-max flow/min cut problem. The sets $\mathcal{S}$ and $\bar{\mathcal{S}}$) separate both ends w_{jk} and $w_{j,k+1}$ of the sub-aisle. If the flow value is smaller than $\bar{f}_{w^{\downarrow}_{jk}}$ or $\bar{f}_{w^{\uparrow}_{j,k+1}}$ (or both), then a violated SEC (4.3c) or (4.3d) is found for sub-aisle (j, k), respectively.

4.5 Computational Results

This section on computational experiments and results discusses the computational setup in Section 4.5.1, describes the generation of b-SPRP-SS instances used in the experiments in Section 4.5.2, analyzes the lower bounds at the root node of the B&C in Section 4.5.2, and compares the number of optimally solved instances and computation times between a B&C solving an equivalent GTSP and our new B&C in Section 4.5.4.

4.5.1 Computational Setup

Our new B&C algorithm is implemented in `C++` using the callable library of `CPLEX` 22.1.2 with Concert Technology and compiled in release mode with `GCC` 13.3.0. `CPLEX` built-in cuts have been used in all experiments. In all calls to `CPLEX`, default values of all parameters are kept except for setting the number of available threads to one. The computational study was performed on the high-performance computing cluster `MOGON KI` of the Johannes Gutenberg University Mainz. The cluster consists of several `AMD EPYC 7713` processors running at 2.0 GHz (the performance of a single thread is slightly lower than that of a standard personal computer).

For solving the unit-demand b-SPRP-SS as GTSP instances, we had access to the reimplementation in `C++` of the B&C algorithm of Fischetti *et al.* (2002) provided by Heßler and Irnich (2024), where the callable library of `CPLEX` is also used. In (Heßler and Irnich, 2024, e-companion, Appendix E), details of the implementation are discussed, including the GTSP model and valid inequalities used in the B&C algorithm.

The B&C algorithm for NF uses a straightforward separation strategy. All separated valid inequalities are added if violated by at least $\varepsilon = 0.1$. Additionally, we skip the separation of valid inequalities for fractional solutions at the point when the branch-and-bound search tree contains more than $n^{B\&B} = 10$ nodes. No other filtering or heuristic separation strategy is used.

B&C algorithms often benefit from tight upper bounds, in particular when solving difficult instances. We therefore use the *iterated local search* (ILS) heuristic for the GTSP by Schmidt and Irnich (2022), also written in `C++`, to compute upper bounds used in Sections 4.5.3 and 4.5.4.

4.5.2 Instances

Goeke and Schneider (2021) introduced a generation procedure for SPRP-SS instances. Identical or very similar procedures have been used later by Lüke *et al.*

(2024) and Heßler and Irnich (2024). We use the same generation scheme to create instances with the following characteristics:

- the number of aisles is $m \in \{5, 10\}$;
- the number of cells per aisle and per side is $C \in \{20, 30\}$;
- the number of blocks is $b \in \{3, 4\}$;
- The number of different articles to collect is $a \in \{20, 30, 40, 50\}$;
- The scatter factor is $\alpha \in \{1.2, 1.8, 2.5, 3.0\}$.

We refer to a specific pick position on one side of the aisle as a cell. Consequently, a pick position consists of two cells, one on either side of the aisle (see Figure 4.1). When possible, all blocks have the same number of cells; otherwise they differ by no more than one cell. As illustrated in Figure 4.1, the distance between two adjacent pick positions is one unit, as is the distance from the first or last position of an aisle to the nearest cross-aisle. Two adjacent aisles have a distance of three units, and each cross-aisle has a height of one unit. The scattering procedure distributes articles randomly throughout the warehouse. More precisely, in the first step, each article $s \in S$ ($a = |S|$) is assigned to a random cell, ensuring that no cell contains more than one article. In the second step, $(\alpha - 1) \cdot a$ uniformly randomly chosen articles $s \in S$ are assigned to random cells, again ensuring that no more than one article per cell. Consequently, on average, each article $s \in S$ can be found α times in the warehouse. However, the demand q_s for article s can be smaller than α (1 is the lower bound) or larger. We repeat the random procedure ten times per setting. Overall, we obtain a set of $1280 = 10 \cdot 2 \cdot 2 \cdot 2 \cdot 4 \cdot 4$ instances.

Instances with general-demand are generated in the same way, except that, for each article $s \in S$, the number of available units at each pick position p is uniformly randomly selected from $b_{sp} \in \{1, 2, 3\}$. Subsequently, the demands q_s are randomly drawn from $\{1, \ldots, \min(6, \sum_p b_{sp})\}$. As there is no competitive solution algorithm available for the general-demand instances, we will not discuss the results here. However, all solutions are available at `https://logistik.bwl.uni-mainz.de/research/benchmarks/`.

4.5.3 Comparison of Lower Bounds

In a first step, we analyze the strength of the dual bounds of the two formulations, i.e., the GTSP model used by Fischetti *et al.* (2002) versus the NF for the unit-demand b-SPRP-SS. More precisely, we compare the root node lower bounds in the B&C implementations in the MIP solver`CPLEX`. The *lower bounds* (LBs) that we compare result from solving the LP relaxation, adding GTSP-specific valid inequalities and SECs of Type I, IIa, IIb$^{\downarrow}$, and IIb$^{\uparrow}$ (see Section 4.4.2), respectively. Additionally, we allow `CPLEX` to add general-purpose cuts. The implementation is rather straightforward by limiting the number of nodes in the B&C to one. We denote by $LB_{GTSP}(I)$ and $LB_{NF}(I)$ the respective lower bounds for a unit-demand

Number of		Scatter	Number of articles $a = 20$		$a = 30$		$a = 40$		$a = 50$	
blocks	aisles	factor	GTSP	NF	GTSP	NF	GTSP	NF	GTSP	NF
$b = 3$	$m = 5$	$\alpha = 1.2$	4.82	1.89	7.28	1.55	10.69	1.49	13.15	1.13
		1.8	16.97	3.38	26.67	1.98	30.69	2.59	34.18	2.16
		2.5	35.51	3.56	45.16	3.63	47.62	3.67	50.84	3.44
		3.0	44.75	4.75	53.86	3.83	55.36	5.24	58.62	4.52
		Subtotal	25.51	3.40	33.24	2.75	36.09	3.25	39.20	2.82
	$m = 10$	$\alpha = 1.2$	3.64	3.02	5.42	2.57	6.88	3.14	8.04	2.92
		1.8	13.11	2.73	18.64	2.50	18.43	2.22	22.89	3.28
		2.5	25.13	3.34	30.84	3.18	32.91	3.30	38.04	3.73
		3.0	31.83	4.84	38.16	3.90	42.22	4.30	47.64	4.68
		Subtotal	18.42	3.48	23.27	3.04	25.11	3.24	29.15	3.65
$b = 4$	$m = 5$	$\alpha = 1.2$	5.55	3.16	6.61	2.92	9.02	2.43	12.06	1.93
		1.8	16.14	3.01	22.23	2.80	28.20	3.12	30.62	3.14
		2.5	32.24	3.08	40.72	3.89	45.14	4.36	49.77	4.92
		3.0	42.60	3.65	50.38	4.43	53.40	5.46	57.20	5.15
		Subtotal	24.13	3.22	29.99	3.51	33.94	3.84	37.41	3.79
	$m = 10$	$\alpha = 1.2$	3.82	4.36	4.81	3.20	6.93	3.59	6.66	3.52
		1.8	13.46	2.74	16.30	3.35	16.90	3.62	20.15	4.20
		2.5	25.08	3.42	28.51	4.20	30.00	4.59	37.31	6.40
		3.0	31.61	3.77	36.12	4.60	40.51	5.56	47.66	10.46
		Subtotal	18.49	3.57	21.44	3.84	23.59	4.34	27.95	6.15

Table 4.2: Average integrality gap $(opt(I) - LB_X(I))/opt(I)$ in percent for the GTSP and the NF model, $X \in \{GTSP, NF\}$. When $opt(I)$ is unknown, we used $UB(I)$ from the ILS heuristic by Schmidt and Irnich (2022).

instance of the b-SPRP-SS.

To make the lower bounds comparable, we compute the relative integrality gap as $100 \cdot (opt(I) - LB(I))/opt(I)$ (in percent), where $opt(I)$ is the optimal objective value and $LB(I)$ is the lower bound at the root node. For 57 out of 1280 instances, we are not able to prove optimality with either B&C algorithm. In these cases, we replace $opt(I)$ by the upper bound $UB(I)$ computed with the heuristic of Schmidt and Irnich (2022). The latter was run for 600 seconds (10 minutes) per instance I. Note that no time limit was given to the GTSP and NF algorithms, because computation times are negligible.

Table 4.2 shows the average relative integrality gaps for 64 groups of instances with an identical number of aisles m, number of blocks b, scatter factor α, and

number of articles to collect a. The 20 instances in each group only differ in the number of cells per aisle (20 or 30), for which we found that relaxation results do not differ significantly. Comparing GTSP and NF bounds, the general trend is very clear: $LB_{GTSP}(I)$ is weaker than $LB_{NF}(I)$. The only group in which GTSP lower bounds are tighter than NF lower bounds is for $(b, m, \alpha, a) = (4, 10, 1.2, 20)$ with average values of 3.82 and 4.36 percent, respectively. While the GTSP model produces average relative gaps that differ between 3.82 and 58.62 percent, those of the NF model are relatively smaller and vary between 1.13 and 10.46 percent. For the GTSP model, the largest relative gap occurs for $(b, m, \alpha, a) = (3, 5, 3.0, 50)$, i.e., for instances with approximately 150 pick positions in five aisles and three blocks. For the NF model, the largest relative gap occurs for $(b, m, \alpha, a) = (4, 10, 3.0, 50)$, i.e., also for 150 pick positions but in combination with ten aisles and four blocks. We can therefore expect that instances with many pick positions are difficult for both types of models and that a larger number of aisles and blocks makes instances less (more) difficult for the GTSP (NF) model.

4.5.4 Integer Results

In this section, we report the results of the complete runs of both B&C algorithms. We set the maximum computation time to 300 seconds (5 minutes) for each run and also use the same grouping of instances as in the previous section. In pretests, we observed that the performance of the B&C algorithm using NF can significantly depend on whether a good feasible solution and herewith a tight primal bound and is found early. Therefore, we test both GTSP and NF in two B&C configurations:

- w/o: without providing an initial *upper bound* (UB)
- UB: providing the upper bound $UB(I) = opt(I) + 1$ (using the `CPLEX` parameter `UpperCutoff`)

where $opt(I)$ is the optimal objective value of an instance I or the best known upper bound that we found in all experiments (for 57 of 1280 instances). Note that these values result from the heuristic GTSP solver and runs of the B&C algorithms. Accordingly, Table 4.3 lists for GTSP and NF, either without or with providing the tight UB, the number of instances for which the B&C algorithms terminate with a proven optimum within the given time limit.

In total, the B&C algorithm using GTSP without UB provides 781 proven optimal solutions, GTSP with UB provides 747, NF without UB provides 1116, and NF with UB provides 1186. The superiority of the NF model can be attributed to the much better dual bound (see Section 4.5.3). However, it is surprising that providing an excellent UB to the B&C algorithm using the GTSP formulation does not have the expected positive effect. One possible explanation is that, for instances with a rather large integrality gap, which is often seen with the GTSP formulation, the quality of the upper bound is not decisive. The MIP solver finds

			Number of articles															
			$a = 20$				$a = 30$				$a = 40$				$a = 50$			
Number of		Scatter	GTSP		NF		GTSP		NF		GTSP		NF		GTSP		NF	
blocks	aisles	factor	w/o	UB	w/o	UB	w/o	UB	w/o	UB	w/o	UB	w/o	UB	w/o	UB	w/o	UB
$b = 3$	$m = 5$	$\alpha = 1.2$	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
		1.8	20	20	20	20	20	20	20	20	11	7	20	20	0	0	20	20
		2.5	20	20	20	20	8	6	20	20	0	0	20	20	0	0	20	20
		3.0	18	15	20	20	0	0	20	20	0	0	20	20	0	0	20	20
		Subtotal	78	75	80	80	48	46	80	80	31	7	80	80	20	20	80	80
	$m = 10$	$\alpha = 1.2$	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
		1.8	20	20	20	20	20	20	20	20	19	18	20	20	8	7	20	20
		2.5	20	20	20	20	9	9	20	20	4	2	20	20	0	0	20	20
		3.0	20	20	20	20	3	1	20	20	0	0	19	20	0	0	20	20
		Subtotal	80	80	80	80	52	50	80	80	43	40	79	80	28	27	80	80
$b = 4$	$m = 5$	$\alpha = 1.2$	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
		1.8	20	20	20	20	20	20	20	20	16	14	20	20	3	1	20	20
		2.5	20	20	20	20	13	9	20	20	0	0	20	20	0	0	20	20
		3.0	17	16	20	20	1	0	20	20	0	0	20	20	0	0	20	20
		Subtotal	77	76	80	80	54	49	80	80	36	34	80	80	23	21	80	80
	$m = 10$	$\alpha = 1.2$	20	20	14	17	20	20	15	17	20	20	10	16	20	20	11	15
		1.8	20	20	17	18	20	20	16	19	20	19	8	12	10	6	3	12
		2.5	20	20	17	18	13	11	9	15	6	4	7	11	0	0	2	5
		3.0	20	20	13	19	1	1	9	15	1	1	6	13	0	0	0	4
		Subtotal	80	80	61	72	54	52	49	66	47	44	31	52	30	26	16	36

Table 4.3: Number of instances solved to proven optimality.

some reasonable bounds independently of the presence of an initial upper bound, resulting in small relative differences in the heuristic gaps.

In what follows, we use the best configuration in each case, i.e., GTSP 'w/o' and NF with 'UB'. For the sake of brevity, we refer to them as GTSP and NF, respectively. Next, we compare computation times.

Table 4.4 shows the average computation time per group. These results show that the B&C algorithm using NF is not always the fastest method. Indeed, for the smallest scatter factor of $\alpha = 1.2$, the GTSP approach is the method of choice (in these instances, on average, only one out of every five article is present twice). In particular, when the scatter factor α is greater than two and at least 30 articles must be picked, the B&C algorithm using NF is faster than the GTSP solver for almost all groups of instances. The B&C algorithm using NF solves instances with $b = 3$ blocks faster than instances with $b = 4$ blocks. Likewise, it is faster for $m = 5$ aisles than for $m = 10$ aisles. This is intuitive because $b \cdot m$ is the number of sub-aisles over which SECs are defined. The best case, $(b, m, \alpha, a) = (3, 5, 1.8, 50)$, has a speedup of more than 300 compared to the GTSP solver.

Cut Statistics For the B&C algorithm using NF with UB, the average number of cuts of Type I, IIa, and IIb differs depending on whether they are computed as `CPLEX`'s user cuts (for fractional solutions) or as lazy cuts (for integer solutions),

Number of		Scatter	Number of articles $a = 20$		$a = 30$		$a = 40$		$a = 50$	
blocks	aisles	factor	GTSP	NF	GTSP	NF	GTSP	NF	GTSP	NF
$b = 3$	$m = 5$	$\alpha = 1.2$	0.03	0.36	0.09	0.38	0.74	0.42	5.86	0.32
		1.8	0.63	0.85	16.78	0.74	199.35	0.91	300.00	0.77
		2.5	14.15	1.09	233.49	1.17	300.00	1.50	300.00	2.09
		3.0	91.19	1.30	300.00	1.35	300.00	3.85	300.00	3.30
		Subtotal	26.50	0.90	137.85	0.91	201.17	1.67	229.60	1.62
	$m = 10$	$\alpha = 1.2$	0.04	2.56	0.09	3.70	0.49	5.61	1.42	5.63
		1.8	0.38	2.72	5.78	4.37	48.82	5.47	207.65	15.34
		2.5	4.05	5.16	183.18	7.95	276.24	15.76	300.00	19.44
		3.0	37.73	9.18	283.92	16.68	300.00	40.19	300.00	47.01
		Subtotal	10.55	4.90	118.24	8.17	157.20	16.76	205.44	21.85
$b = 4$	$m = 5$	$\alpha = 1.2$	0.03	6.14	0.10	5.14	0.37	2.82	2.23	2.05
		1.8	0.44	6.94	4.61	8.82	102.94	10.56	264.63	9.08
		2.5	13.98	9.75	170.93	11.47	300.00	21.97	300.00	32.31
		3.0	77.86	9.44	296.88	13.39	300.00	28.08	300.00	40.24
		Subtotal	23.08	8.07	118.13	9.70	177.00	15.86	219.43	20.92
	$m = 10$	$\alpha = 1.2$	0.04	101.88	0.09	101.82	0.31	121.27	0.85	162.82
		1.8	0.43	63.77	2.63	84.67	29.85	171.66	185.39	225.69
		2.5	4.56	66.38	152.32	125.34	256.34	196.40	300.00	255.48
		3.0	21.33	87.95	289.32	153.60	292.72	171.80	300.00	272.57
		Subtotal	6.59	79.99	111.09	116.36	144.80	165.28	199.50	229.14

Table 4.4: Average computation time (in seconds) of the B&C algorithm using GTSP model without UB and NF with UB. Instances not solved to optimality within the 300-second time limit are considered with 300s.

and depending on the type of instance. Accordingly, we use the identical grouping scheme as in the above tables to differentiate between instances. Tables 4.5a and 4.5b show the average number of separated user and lazy cuts of each type, respectively.

Since the user cut callback of `CPLEX` is invoked much more often than the lazy callback, the numbers in Table 4.5a tend to be higher than those in Table 4.5b. Moreover, it is intuitive that groups of instances that require more computation time (see Table 4.4) also have a higher number of separated cuts. For fractional solutions, the majority of the separated cuts are of Type I and IIa, while for integer solutions, the majority of the separated cuts are of Type IIb. An unexpected outcome is that almost no cuts of Type IIa are separated in the lazy cut callback.

Finally, we analyzed how much of the total computation time of the B&C algorithm is spent on separation. The outcome was unexpected but very consistent across all types of instances and all types of cuts: The portion of time spent with separation is negligible.

Number of blocks	Number of aisles	Scatter factor	Number of articles $a = 20$ type I	$a = 20$ IIa	$a = 20$ IIb	$a = 30$ type I	$a = 30$ IIa	$a = 30$ IIb	$a = 40$ type I	$a = 40$ IIa	$a = 40$ IIb	$a = 50$ type I	$a = 50$ IIa	$a = 50$ IIb
$b = 3$	$m = 5$	$\alpha = 1.2$	39.20	20.15	2.65	31.90	18.75	2.75	32.15	19.10	1.70	28.70	17.10	1.60
		1.8	49.95	21.15	3.55	56.55	23.75	3.50	65.25	24.45	3.00	65.05	21.35	2.50
		2.5	51.10	18.70	4.75	68.85	20.30	3.75	84.50	22.40	3.85	112.75	28.30	4.05
		3.0	46.30	16.05	3.45	70.90	18.40	5.15	87.30	19.00	6.05	113.20	22.95	5.75
		Subtotal	46.64	19.01	3.60	57.05	20.30	3.79	67.30	21.24	3.65	79.93	22.43	3.48
	$m = 10$	$\alpha = 1.2$	109.80	55.60	5.90	145.45	79.70	8.10	161.95	98.20	9.50	185.00	105.70	7.05
		1.8	115.55	48.15	8.00	172.90	72.85	9.45	196.15	80.05	8.85	205.00	84.05	9.35
		2.5	111.50	41.80	9.20	161.75	62.50	12.35	198.15	68.50	10.55	239.95	78.50	11.35
		3.0	100.75	39.55	10.05	161.55	53.55	12.30	192.05	57.85	13.55	254.25	67.10	13.00
		Subtotal	109.40	46.28	8.29	160.41	67.15	10.55	187.08	76.15	10.61	221.05	83.84	10.19
$b = 4$	$m = 5$	$\alpha = 1.2$	81.55	37.90	4.65	80.85	41.65	4.75	69.25	39.25	4.00	77.45	44.10	3.65
		1.8	90.55	35.75	5.15	90.90	31.50	5.35	120.55	45.15	5.65	130.40	44.75	4.95
		2.5	80.95	28.00	6.20	109.25	30.10	7.75	127.50	34.20	6.15	175.45	41.15	7.45
		3.0	66.60	20.70	6.25	94.85	26.80	8.50	145.40	32.05	7.10	180.20	37.65	8.10
		Subtotal	79.91	30.59	5.56	93.96	32.51	6.59	115.68	37.66	5.73	140.88	41.91	6.04
	$m = 10$	$\alpha = 1.2$	194.50	99.70	8.55	260.60	135.20	10.25	282.90	151.20	10.95	338.95	185.35	12.90
		1.8	188.60	85.20	11.15	266.95	115.70	13.15	286.15	111.65	14.05	368.10	148.40	14.05
		2.5	171.65	64.15	11.90	234.45	92.70	16.55	306.75	102.25	16.40	392.70	123.40	19.65
		3.0	151.50	57.60	13.70	221.30	77.45	18.65	278.10	86.30	20.10	375.20	102.90	20.00
		Subtotal	176.56	76.66	11.33	245.83	105.26	14.65	288.48	112.85	15.38	368.74	140.01	16.65

(a) `CPLEX`'s user cuts, fractional solutions.

Number of blocks	Number of aisles	Scatter factor	Number of articles $a = 20$ type I	$a = 20$ IIa	$a = 20$ IIb	$a = 30$ type I	$a = 30$ IIa	$a = 30$ IIb	$a = 40$ type I	$a = 40$ IIa	$a = 40$ IIb	$a = 50$ type I	$a = 50$ IIa	$a = 50$ IIb
$b = 3$	$m = 5$	$\alpha = 1.2$	2.00	0.00	8.00	2.95	0.00	8.75	2.70	0.00	7.70	2.75	0.00	6.75
		1.8	2.00	0.00	8.85	2.40	0.00	7.20	2.50	0.00	6.85	2.50	0.00	6.05
		2.5	1.40	0.15	7.50	2.20	0.00	7.55	1.85	0.05	5.60	1.90	0.00	6.25
		3.0	1.75	0.05	8.20	1.35	0.05	5.30	1.30	0.00	5.55	1.45	0.00	4.40
		Subtotal	1.79	0.05	8.14	2.23	0.01	7.20	2.09	0.01	6.43	2.15	0.00	5.86
	$m = 10$	$\alpha = 1.2$	2.00	0.00	12.20	2.50	0.00	12.45	2.75	0.00	11.30	3.85	0.00	12.75
		1.8	1.75	0.00	11.55	2.45	0.05	12.40	2.85	0.00	11.00	3.25	0.00	10.80
		2.5	1.60	0.15	10.95	1.75	0.10	10.15	2.35	0.05	11.35	2.70	0.00	10.70
		3.0	1.75	0.00	11.05	1.60	0.20	8.20	1.30	0.05	7.35	1.70	0.05	9.05
		Subtotal	1.78	0.04	11.44	2.08	0.09	10.80	2.31	0.03	10.25	2.88	0.01	10.83
$b = 4$	$m = 5$	$\alpha = 1.2$	1.65	0.00	9.55	2.40	0.00	10.10	2.60	0.00	8.85	2.95	0.00	8.40
		1.8	1.90	0.05	8.80	2.20	0.00	9.60	2.55	0.00	8.55	2.90	0.00	8.35
		2.5	1.90	0.00	9.10	1.60	0.05	7.10	1.75	0.00	6.95	1.65	0.00	6.55
		3.0	1.40	0.15	9.75	1.15	0.00	5.85	1.15	0.00	5.90	0.85	0.05	4.20
		Subtotal	1.71	0.05	9.30	1.84	0.01	8.16	2.01	0.00	7.56	2.09	0.01	6.88
	$m = 10$	$\alpha = 1.2$	1.60	0.00	13.25	2.30	0.00	14.65	2.85	0.00	16.45	2.50	0.00	14.95
		1.8	1.50	0.05	11.75	2.00	0.10	14.40	2.55	0.00	15.05	2.65	0.05	13.45
		2.5	1.50	0.00	11.90	1.65	0.15	11.50	1.90	0.10	11.70	1.65	0.05	9.70
		3.0	1.40	0.00	12.40	1.30	0.25	9.75	1.25	0.25	8.10	1.10	0.00	6.90
		Subtotal	1.50	0.01	12.33	1.81	0.13	12.58	2.14	0.09	12.83	1.98	0.03	11.25

(b) `CPLEX`'s lazy cuts, integer solutions.

Table 4.5: Average number of cuts added by type using NF with UB.

For integer solutions, the average time spent with separation is below 0.2 percent of the total time, with a maximum of 2.1 percent for a b-SPRP-SS instance. This can be explained by the few calls to the lazy cut callback and the highly efficient union-find algorithm for component identification, which is applied to a relatively small support graph.

For fractional solutions, the same procedure determines whether the support graph is connected or not. Less than 0.1 percent of the total B&C time is consumed for this task, with a maximum of 0.9 percent found in some instances. The consecutive max flow/min cut procedures (in case of a single component) require, on average, 0.24 percent for separating SECs of Type I, 0.18 percent for Type IIa, and 0.05 percent for Type IIb of the total B&C time. The respective maximum percentages are 1.68, 1.41, and 0.5 percent for the Types I, IIa, and IIb, respectively.

We interpret this outcome as strong support for the overarching idea of our approach. It is much more effective to translate the solution values $\bar{x}$ from the huge relaxed state space into the small support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$. Separation can be performed fast here. The result is that the main bottleneck of the B&C algorithm is the LP re-optimization and management of the branch-and-bound search tree, including pre-solving and model reduction, cut management, heuristics trying to identify good integer solutions, etc.

Impact of the Number of Cells per Aisle We have not yet analyzed the impact of the number C of cells per aisle on the computational performance. Recall that all of the above tables have presented aggregated results regarding C. Since the GTSP approach did not reveal any significant differences, we only present results for our new B&C algorithm using NF. Table 4.6 compares computation times for $C = 20$ and $C = 30$ cells per aisle. With a few exceptions, the average computation time is shorter for $C = 20$ than for $C = 30$. The only group in which not all instances are optimally solved using B&C with NF is the one with $b = 4$ blocks and $m = 10$ aisles. In this group, 40 instances (of 160) with 20 cells and 54 instances (of 160) with 30 cells remain unsolved within the 300-second time limit.

4.6 Conclusions and Outlook

In this paper, we have presented the first problem-tailored exact algorithm for the b-SPRP-SS in parallel-aisle warehouses with more than two blocks. Currently, only one exact solution approach is known for these variants of the SPRP-SS, but it is only applicable in the unit-demand case. This approach relies on the fact that these b-SPRP-SS instances can be solved as GTSP instances. We have proposed a B&C algorithm that outperforms the GTSP solver on instances of the b-SPRP-SS

Number of		Scatter	Number of articles $a = 20$		$a = 30$		$a = 40$		$a = 50$	
blocks	aisles	factor	$C = 20$	30	$C = 20$	30	$C = 20$	30	$C = 20$	30
$b = 3$	$m = 5$	$\alpha = 1.2$	0.33	0.38	0.40	0.37	0.43	0.41	0.28	0.36
		1.8	0.84	0.85	0.67	0.81	0.76	1.05	0.75	0.79
		2.5	0.81	1.37	0.96	1.38	1.22	1.78	2.09	2.09
		3.0	1.42	1.18	1.03	1.67	3.15	4.56	3.75	2.85
		Subtotal	0.85	0.95	0.76	1.06	1.39	1.95	1.72	1.52
	$m = 10$	$\alpha = 1.2$	2.44	2.68	3.76	3.63	4.35	6.87	6.51	4.76
		1.8	2.43	3.01	3.86	4.88	2.87	8.07	17.34	13.33
		2.5	5.24	5.08	5.32	10.58	22.40	9.13	22.28	16.59
		3.0	7.29	11.06	15.60	17.76	42.46	37.93	50.67	43.36
		Subtotal	4.35	5.46	7.13	9.21	18.02	15.50	24.20	19.51
$b = 4$	$m = 5$	$\alpha = 1.2$	4.58	7.70	4.51	5.78	2.05	3.59	1.62	2.48
		1.8	4.55	9.32	5.68	11.96	8.31	12.81	5.97	12.18
		2.5	4.62	14.88	4.78	18.15	11.90	32.04	25.21	39.41
		3.0	6.52	12.36	6.18	20.60	21.63	34.52	25.39	55.09
		Subtotal	5.07	11.06	5.29	14.12	10.97	20.74	14.55	27.29
	$m = 10$	$\alpha = 1.2$	74.05	129.71	65.30	138.35	77.68	164.85	150.12	175.51
		1.8	37.17	90.36	40.99	128.35	168.26	175.07	193.66	257.72
		2.5	55.05	77.71	126.20	124.49	195.81	196.99	250.38	260.58
		3.0	53.19	122.70	115.40	191.79	144.99	198.60	256.41	288.72
		Subtotal	54.86	105.12	86.97	145.74	146.68	183.88	212.64	245.64

Table 4.6: Average computation time (in seconds) of the B&C algorithm using NF with UB for $C = 20$ and 30. Instances not solved to optimality within the 300-second time limit are considered with 300s.

in parallel-aisle warehouses with three or four blocks, including both unit-demand and general-demand cases.

From a methodological point of view, we introduced a hierarchy of SECs sufficient to produce integer solutions. We showed that the separation problems for all these types of SECs can be efficiently solved using a so-called warehouse graph, which resembles the aisles and cross-aisles of a warehouse. Solving the separation requires the solving of a series of max flow/min cut problems defined over that warehouse graph.

Extensive computational experiments yielded the following key findings: The linear programming relaxation of the new NF usually has a smaller integrality gap than the GTSP model used in the competitor's B&C algorithm. Therefore, within a 5-minute computation time limit, our new B&C algorithm can compute many more provably optimal solutions than the GTSP solver. However, the computational study also shows that results are not clear-cut. Our B&C algorithm only outperforms the B&C algorithm for the GTSP when there are many pick positions, i.e., when the scatter factors is greater than 2 and more than 20 articles to be picked

on the picker tour. Otherwise, our proposed B&C can be considerably slower than the B&C algorithm for the GTSP. In summary, our new B&C algorithm exploits the fact that the corresponding GTSP instances often have comparatively more vertices (one for each relevant pick position) than the warehouse graph of that instance.

We think that future research on effective exact algorithms for b-SPRP-SS, particularly for warehouses with more aisles or blocks, should probably not rely on the type of dynamic-programming state spaces that were used here. Promising B&C-based approaches could combine inequalities for GTSP with cuts that exploit the warehouse structure.

Acknowledgement

Parts of this research were supported by Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/13-1 of project 555303283.

The authors gratefully acknowledge the computing time granted on the high performance computing cluster MOGON NHR Süd-West at Johannes Gutenberg University Mainz (hpc.uni-mainz.de).

Bibliography

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey.

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.

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. (2018). Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research*, **57**(3), 888–906.

Daniels, R. L., Rummel, J. L., and Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, **105**(1), 1–17.

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.

Drury, J. (1988). Towards more efficient order picking. Technical report, The Institute of Materials Management, Cranfield, UK.

Fischetti, M., Salazar-Gonzalez, J.-J., and Toth, P. (2002). The generalized traveling salesman and orienteering problems. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, chapter 13, pages 609–662. Kluwer, Dordrecht.

Frazelle, E. (2002). *World-Class Warehousing and Material Handling*. McGraw-Hill, New York.

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

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.

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

Haouassi, M., Kergosien, Y., Mendoza, J. E., and Rousseau, L.-M. (2025). The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, **63**(4), 1304–1325.

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*, **36**(6), 1417–1435.

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**(3), 909–951.

Lüke, L., Hessenius, A., and Irnich, S. (2025). A linear-size model for the single picker routing problem with scattered storage. *European Journal of Operational Research*, **332**(1), 132–143.

Masae, M., Glock, C. H., and Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, **224**, 107564.

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

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

Prunet, T., Absi, N., and Cattaruzza, D. (2025). A note on the complexity of the picker routing problem in multi-block warehouses and related problems. *Annals of Operations Research*, **347**, 1595–1605.

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.

Revenant, P., Cambazard, H., and Catusse, N. (2025). A note about a transition of ratliff and rosenthal's order picking algorithm for rectangular warehouses. *Operations Research Letters*, **62**, 107325.

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

Saylam, S., Çelik, M., and Süral, H. (2024). Arc routing based compact formulations for picker routing in single and two block parallel aisle warehouses. *European Journal of Operational Research*, **313**(1), 225–240.

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.

Schmidt, J. and Irnich, S. (2022). New neighborhoods and an iterated local search algorithm for the generalized traveling salesman problem. *EURO Journal on Computational Optimization*, **10**, 100029.

Singh, K. N. and van Oudheusden, D. L. (1997). A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, **97**(3), 571–579.

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., and Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, **305**(1), 222–239.

Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, **22**(2), 215–225.

Theys, C., Bräysy, O., Dullaert, W., and Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, **200**(3), 755–763.

Tompkins, J., White, J., Bozer, Y., Frazelle, E., and Tanchoco, J. (2003). *Facilities Planning*. John Wiley & Sons, Hoboken, NJ.

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., Ramaekers, K., Caris, A., and de Koster, R. B. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, **267**(1), 1–15.

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

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers and Operations Research*, **95**, 139–150.

Weidinger, F., Boysen, N., and Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, **274**(2), 501–515.

Wildt, C. and Weidinger, F. (2026). One fits all: a flexible branch-and-cut algorithm for picker routing in scattered storage warehouses. *OR Spectrum*.

Wildt, C., Weidinger, F., and Boysen, N. (2025). Picker routing in scattered storage warehouses: an evaluation of solution methods based on TSP transformations. *OR Spectrum*, **47**, 35–66.

Chapter 5

Exact Solution of Picker Routing Problems in Zoned Warehouses with Scattered Storage

Laura Lüke

Abstract

We present two picker routing problems in a warehouse split into multiple disjoint zones with a picker assigned to each zone. Scattered storage is applied, meaning an article can be stored at several pick positions in one or more zones, and capacity restrictions limit the number of collected articles for each picker. Both problems seek picker tours that collect all requested articles, with each tour operating within one zone. While the multi-zone picker routing problem (MZPRP) minimizes the total length of all picker tours, the balanced multi-zone picker routing problem (BMZPRP) balances the tour lengths by minimizing the length of the longest picker tour. Both problems constitute a three-level optimization problem. While on the higher levels, zone assignment decisions and the selection of pick positions must be made, on the lower level, picker tours for each zone are determined.

To solve both problems, we use a network-flow model with covering constraints and varying objectives. This type of model was recently presented for the single picker routing problem with scattered storage, building on an extended state space of the dynamic-programming approach by Ratliff and Rosenthal. Our model contains a network for each zone, and demand-covering constraints across all zones ensure that the requested articles are collected. Computational experiments, including large instances with up to 200 articles, show that our solution approach efficiently solves the two problems within (milli-)seconds. An analysis of the number of zones demonstrates that in larger warehouse layouts, zoning reduces costs compared to single-zone picker routing.

5.1 Introduction

Although, especially in e-commerce warehouses, automation is increasing, *picker routing* is still an indispensable component of warehouse operations and is likely to remain so (Schiffer *et al.*, 2022). Picker routing determines the order in which required articles are collected from their storage locations in the warehouse, so that the length of the resulting picker tour, starting and ending at a fixed depot, is minimized. There are multiple reasons why picker routing is still highly relevant and will remain important in the future. On the one hand, like human pickers, autonomous mobile robots (AMRs) also need a predefined, efficient route to collect the required articles (Bock *et al.*, 2025). Therefore, picker routing problems are also employed to optimize picking routes for robots. This also includes, for instance, scenarios in which human pickers are assisted by automated guided vehicles (AGVs) that accompany them during the picking process and subsequently transport the collected articles back to the depot. Meanwhile, the picker can proceed with the next picking list, accompanied by a different AGV (Löffler *et al.*, 2022). On the other hand, manual order picking is advantageous for some of the typical features of e-commerce orders. Highly volatile demand and tight delivery schedules are characteristics of online retail (Boysen *et al.*, 2019). Discount campaigns or seasonal sales lead to short-term increases in both the number of orders and the volume of individual orders, and a fast-delivery standard requires fast processing of incoming orders. Especially the combination of those two features leads to highly variable workloads, most automated warehouse systems have difficulties to adapt to (Löffler *et al.*, 2022). However, manually operated warehouses are flexible and can more easily handle peaks in demand by scheduling additional pickers. Despite the feasibility of automating order picking, many companies opt for manual processes. Their rationale is that humans can respond more flexibly to unexpected changes, particularly when logical reasoning is required — something machines cannot yet do effectively (Grosse *et al.*, 2015).

In this paper, we examine picker routing in *zoned* warehouses with *scattered storage*, as the combination of zoning and scattered storage is widespread in warehouses for online retail (e.g., Weidinger, 2018; Boysen *et al.*, 2019; Schiffer *et al.*, 2022). This warehouse configuration is adopted by major global retailers, including Amazon Europe and the fashion retailer Zalando (Boysen *et al.*, 2019), underscoring its practical relevance. To the best of our knowledge, the existing literature does not provide an exact solution approach that simultaneously addresses picker routing under both zoning and scattered storage. This research aims to close this gap by providing an exact method for a problem setting of significant industrial importance.

In warehouses with scattered storage, also called *mixed-shelves storage* (Weidinger, 2018), some or all articles can be picked from more than one pick position

(=storage location) in the warehouse. The main advantage of a scattered storage strategy is "that items of demanded SKUs are found close by irrespective of the position within the warehouse [so that] the distance to be covered for order picking is reduced this way" (Weidinger, 2018, p. 139). However, a picker may need to visit multiple pick positions to collect a large quantity of a single article (Weidinger, 2018).

Zoning is a layout design that divides the warehouse into disjoint zones. A pick zone comprises pick positions typically arranged closely together, where a subset of all *stock keeping units* (SKUs) in the warehouse is stored. At least one picker is assigned to each zone to pick the demanded articles exclusively within this zone. In the literature, a distinction is usually made between one (Jane, 2000) and several pickers per zone (Chen *et al.*, 2013). We examine the case with one picker per zone, eliminating the need to consider congestion effects. *Picker congestion* or *picker blocking* occurs when a picker is blocked by another picker from passing or accessing a pick position (Parikh and Meller, 2009). The benefits of zoned warehouses are numerous: The picker attains a high ratio of time spent extracting SKUs compared to time spent traveling between pick positions, due to the restricted size of the zone. This also results in enhanced familiarity with the SKUs within the zone, and the picking time of a specific order can be significantly reduced (Gu *et al.*, 2007). Compared to order picking systems where several pickers move through the whole warehouse, zone picking reduces traffic congestion (de Koster *et al.*, 2007). Nonetheless, there also come along disadvantages with zoning: If the required articles from different customer orders are picked simultaneously in different zones, they must then be consolidated again according to the original orders. This usually requires a separate downstream process. However, if picking takes place one after the other in the different zones, containers with the partially picked articles are transferred from zone to zone. This can lead to unnecessary waiting times for pickers who are waiting for their bin for the next picker tour (Gu *et al.*, 2007).

For both picker routing problems presented in this work, a zoned single-block warehouse with parallel aisles and a depot located in each zone is given, where each zone is assigned a picker with a predefined capacity limiting the number of articles to be picked in that zone. A set of articles must be collected from pick positions distributed throughout the warehouse, assuming each article is stored at one or more pick positions. Both problems seek simultaneous picker tours within the zones that start and end at the depots in each zone, such that all demanded articles are collected and the capacities of the pickers are respected. While the *multi-zone picker routing problem* (MZPRP) focuses on minimizing the total distance traveled across all picker tours, the *balanced multi-zone picker routing problem* (BMZPRP) aims to achieve a more equitable distribution by minimizing the length of the longest picker tour. To keep the problem description clear and the problem

universally applicable, we assume the set of demanded articles is given. Thus, the set of articles may describe either a single customer order or multiple orders batched in a preprocessing step. Additionally, we limit the problem to one picker per zone. This prevents pickers from blocking each other. A picker is assigned to each zone in every working shift, as certain articles may be stored exclusively in specific zones and can therefore only be picked there. Therefore, the pickers' fixed costs are constant and can be neglected.

Figure 5.1 shows an optimal solution for the MZPRP for an instance with ten articles, three zones, and a picker capacity of five articles. The depots are marked in gray, and encircled SKUs indicate that the articles are collected from these pick positions. According to the problem description, both problem variants can be seen as a three-level optimization problem. The decision at the highest level concerns the zone assignment and determines which required articles are picked in each zone. The second level deals with the scattered storage aspect by defining the subset of SKUs to be picked per zone. Last but not least, the picker tours are determined at the third level. Since the decisions at all three levels are interdependent, they must be made simultaneously to solve the problem optimally.

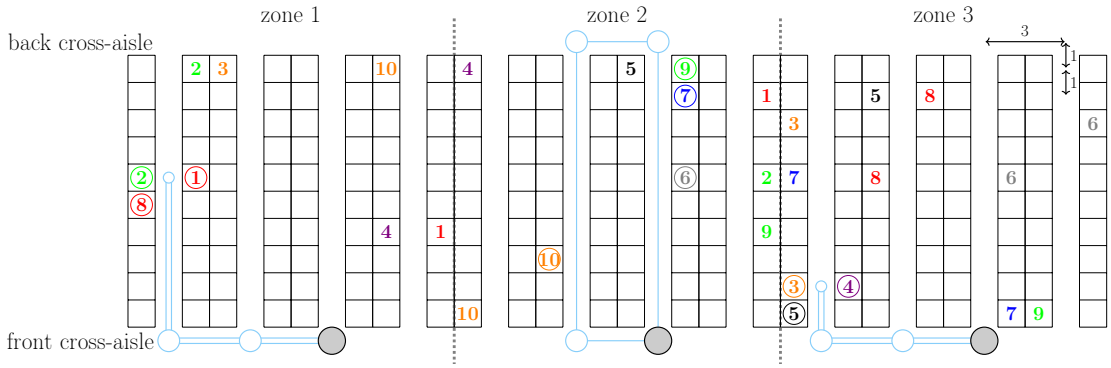


Figure 5.1: An optimal solution for an MZPRP instance with ten articles, three zones, and a capacity of $Q = 5$ for all pickers.

5.1.1 Contributions

This work contributes to further investigation of the two problem areas of zoning and picker routing in scattered warehouses, both of which have received little scientific attention to date. Despite their common occurrence in the industry, the combination of these challenges remains unexplored in the literature to the best of our knowledge. We define the two problems MZPRP and BMZPRP to address this research gap. The key contributions of this work are:

- We introduce two picker routing problems in zoned warehouses with scattered storage. While the MZPRP minimizes the total length of all picker tours, the BMZPRP minimizes the length of the largest picker tour to balance the picker tour lengths of all zones and, thus, the workload of all pickers. Both problems involve three levels of optimization, contributing to their computational complexity.
- We present an effective solution algorithm to both problems consisting of a network-flow model with different types of covering constraints that can be solved with the help of a *mixed-integer programming* (MIP) solver.
- We show that our generic model can also be applied universally to various routing problem variants of the (B)MZPRP that involve different warehouse layouts, multiple depots per zone, or routing strategies.
- Our extensive computational study, including very large instances with up to 200 required articles, shows that our solution approach efficiently solves both problems. Concerning the MZPRP, the vast majority of instances can be solved optimally within milliseconds. Only some of the largest instances with 200 articles require a few seconds. For the BMZPRP, the computation times are considerably longer for the largest instances with up to 60 aisles and 200 articles, and a few of these instances cannot be solved to optimality within the time limit.
- An analysis of the number of zones demonstrates that in larger warehouse layouts, implementing zoning can reduce costs by up to 12.6% compared to using a single-zone picker routing approach.

5.1.2 Structure

The remainder of this paper is structured as follows. In Section 5.2 we present a literature review on the topics of zoning and picker routing in scattered warehouses. Section 5.3 describes the single picker routing problem in warehouses with scattered storage and the extended state space of Heßler and Irnich (2024), which is the basis for our solution approach. Section 5.4 addresses the MZPRP and the BMZPRP and presents the new exact solution approach. Section 5.5 reports and analyzes the computational results. Final conclusions are drawn in Section 5.6.

5.2 Literature Review

The literature on zoning and picker routing in warehouses with scattered storage is very limited. However, zoning has a substantial impact on the picking performance

(de Koster *et al.*, 2007) and scattered storage is widely used in practice (Weidinger, 2018). While only 6% of the surveyed literature by Gu *et al.* (2007) considered zoning, only 7% of the routing algorithms reviewed by Masae *et al.* (2020b) referred to scattered storage. To the best of our knowledge, the combination of zoning and picker routing in warehouses with scattered storage has not yet been considered in the literature. Therefore, we will first focus on zoning and then on scattered storage.

Extensive surveys that take into account zoning are conducted by Gu *et al.* (2007), de Koster *et al.* (2007), Gu *et al.* (2010), van Gils *et al.* (2018), and Boysen *et al.* (2019). In the literature, a distinction is usually made between two different types of zone picking. While simultaneous zone picking involves picking all SKUs in parallel in different zones, resulting in a high pick-rate of the pickers, sequential zone picking leads to a lower pick-rate, as the articles of an order are picked in one zone at a time (Parikh and Meller, 2008). However, simultaneous zone picking requires an additional process to sort the picked articles according to the original orders. This can either be done directly by the picker while picking (sort-while-pick), if the picking cart (or equivalent tool) has a sorting system, or in an additional downstream sorting process (pick-and-sort) (Choe, 1991). There are many different names for these two types of picking in the literature. Synonyms for simultaneous zone picking are concurrent (Gu *et al.*, 2007), parallel (Gu *et al.*, 2010), or synchronized (de Koster *et al.*, 2007) zone picking. Sequential zone picking is also called progressive zoning or pick-and-pass (de Koster *et al.*, 2007).

A batch is a set of orders where all the demanded articles are picked together, and batching describes the process of grouping customer orders into batches. Already Mellema and Smith (1988) showed that picking batched orders in zones is more productive than full-range picking of single orders. Batching is often combined with zoning for the picking process, as both operations are similar. For example Yu and De Koster (2009) described an approximation model that analyzes the effect of dynamic batching and zoning on the order throughput time for sequential zone picking. However, Parikh and Meller (2008) examined the batch versus zone problem where either batching or zoning is applied. Gray *et al.* (1992) used simulations to determine the optimal number of zones (as part of an integrated problem) in a fixed-size warehouse. The authors assumed that each zone consists of only one aisle and that several pickers can work in parallel in each zone, using simultaneous zone picking with a subsequent sorting process. de Koster *et al.* (2012) considered an almost identical warehouse setting, except that a zone can also consist of multiple aisles, and determined the optimal number of zones as an isolated problem. Petersen (2002) considered the effect of different batch sizes and shapes of the zones on the picker tour lengths. For a given number of zones and a given number of pick positions per zone, zone configurations differing in the number of

aisles and pick positions per aisle are examined. The storage location assignment in a warehouse with sequential zone picking was examined heuristically by Jane (2000) with the goal of balancing the workload in the different zones. Jane and Laih (2005) concentrated on simultaneous zone picking and proposed a heuristic approach to the same problem, incorporating the co-appearance of distinct articles in the same order. For the objective function of minimizing the longest picker tour, an exact approach was proposed by Saylam *et al.* (2023) for simultaneous picking in warehouses with dynamic zones, i.e., zones can be reconfigured after each picking wave. Table 5.1 provides an overview of the various attributes of zone picking problems. Both MZPRP and BMZPRP use a given pick list, but it is also possible to solve a batching problem during preprocessing.

Reference / Problem	Picking Strategy	Batching	Workload Balancing	Scattered Storage
de Koster *et al.* (2012)	simultaneous	yes	no	no
Gray *et al.* (1992)	simultaneous	yes	yes	no
Jane (2000)	sequential	no	yes	no
Jane and Laih (2005)	simultaneous	no	yes	no
Mellema and Smith (1988)	simultaneous	yes	no	no
Parikh and Meller (2008)	simultaneous	yes	yes	no
Petersen (2002)	simultaneous, sequential	no	no	no
Saylam *et al.* (2023)	simultaneous	no	yes	no
Yu and De Koster (2009)	sequential	yes	no	no
MZPRP & BMZPRP	simultaneous	no	yes	yes

Table 5.1: Attributes of zone picking problems.

The *single picker routing problem with scattered storage* (SPRP-SS) aims to determine the shortest possible picker tour in a warehouse where each SKU to be collected by the picker may be stored at more than one pick position. If each SKU is stored only at a single pick position, the decision from which pick position a particular SKU is to be collected is no longer necessary. The resulting problem is called *single picker routing problem* (SPRP). Ratliff and Rosenthal (1983) demonstrated that the SPRP is a well-solvable case of the NP-hard *traveling salesman problem* (TSP) (Gutin and Punnen, 2002). Using the dynamic-programming algorithm of Ratliff and Rosenthal (1983), this classic order picking problem can even be solved in linear time (Heßler and Irnich, 2022b). However, the SPRP-SS is proven to be NP-hard in the strong sense by Weidinger (2018). Singh and van Oudheusden (1997) showed that the SPRP-SS is a special case of the traveling purchaser problem. Daniels *et al.* (1998) proposed a formulation based on the TSP and an efficient heuristic based on the tabu search metaheuristic for the SPRP-SS. Gu *et al.* (2007) asserted that picker routing with scattered storage is a barely investigated problem with a lot of research potential. Nevertheless, it took more than another ten years for the problem to be researched further. Weidinger (2018) proposed heuristics that select the pick positions for the required SKUs according

to specific rules in a first step, before the dynamic program of Ratliff and Rosenthal (1983) for the resulting SPRP is then applied in a second step. Weidinger *et al.* (2019) added multiple depots to the SPRP-SS and provided heuristic solution methods. MIP-based solution approaches for parallel-aisle warehouses have been proposed by Goeke and Schneider (2021) and Su *et al.* (2023). While the former approach solves the SPRP-SS in single-block warehouses, the latter is designed for multi-block warehouses. Also for multi-block warehouses, Haouassi *et al.* (2025) provided an exact logic-based Benders decomposition method. The state-of-the-art exact approach is the one by Heßler and Irnich (2024). They extended the state space of Ratliff and Rosenthal (1983) which is the basis for a MIP model solved by a commercial solver. A more recent heuristic was presented by Wildt *et al.* (2025), who use transformation schemes for TSP variants to reduce the SPRP-SS to a TSP, which is then solved close to optimality by generic TSP solvers. The advantage of this method is that it is not limited to a specific warehouse layout. Instead of exact routing, Lüke *et al.* (2024) applied different rule-based routing strategies designed for the SPRP (traversal, midpoint, largest gap (Hall, 1993) and return, composite (Petersen, 1997)) to solve the SPRP-SS and showed that the NP-hardness of the problem remains in this case.

5.3 Single Picker Routing Problem with Scattered Storage

In this section, we define the SPRP-SS and briefly introduce the extended state space and modeling approach of Heßler and Irnich (2024). The MZPRP and the BMZPRP are then considered in the following Section 5.4.

5.3.1 Problem Definition

Given is a single-block warehouse with parallel aisles and a set of articles S. Each article is stored at one or more pick positions with a supply of ≥ 1 per pick position. The demand for each article $s \in S$ is denoted by $q_s \in \mathbb{N}$, whereby a distinction is made between two different cases: In the so-called *unit-demand* case, we assume that the articles of S always have a demand of one, so that only a single pick position per required article needs to be visited. In the *general-demand* case, the demanded quantity of an article may be greater than the supply at a specific pick position, meaning that several pick positions must be visited to satisfy the demand of one article. There is a single depot in the warehouse, which serves as the picker's start and end point. The picker aims to collect all the required articles in the warehouse in such a way that the resulting picker tour is as short as possible.

We assume that the process of picking the articles from the pick positions does not require any additional distance.

5.3.2 Extended State Space and Modeling Approach of Heßler and Irnich (2024)

To be able to use Ratliff and Rosenthal's state space also for warehouse layouts with scattered storage, Heßler and Irnich (2024) extended the state space originally designed for the SPRP. For the sake of scope, we do not discuss the state space of Ratliff and Rosenthal (1983) in detail and refer the interested reader to their seminal paper. Instead, we describe the basic adaptions of the extended state space for the SPRP-SS.

The state space consists of nodes and (directed) edges. The nodes V of the extended state space remain the same compared to the nodes of the state space for the SPRP. Each node represents a combination of a specific *stage* and *state*. While a stage describes up to which aisle (numbered from left to right) the (still incomplete) picker tour has been constructed at this node, a state provides very limited information about the structure of the (incomplete) picker tour. Moreover, there is a designated origin node o and a destination node d in the state space.

With regard to the edges, we distinguish between the two edge types E^{aisle} and E^{cross}, which describe all possible traversing options (=actions) of the aisles and cross-aisles in a warehouse that can potentially result in an optimal solution. We assume $J = \{1, 2, \ldots, m\}$ denotes the set of aisles. Then the complete set of directed edges is defined as

$$E = \bigcup_{j=1}^{m} \left(E_j^{aisle} \cup E_j^{cross} \right).$$

The cross-aisle actions E^{cross} are identical in the state spaces for the SPRP and the SPRP-SS and are not discussed in detail in this work. In contrast, the aisle actions E^{aisle} need some adaption, which we describe in the following.

Concerning the actions within an aisle, the six options

$$E^{aisle} = \{\texttt{1pass}, \texttt{2pass}, \texttt{top}, \texttt{bottom}, \texttt{gap}, \texttt{void}\}$$

only exist. Figure 5.2 shows the optimal picker route of an SPRP-SS instance that includes all aisle action types except `2pass`. `2pass` is shown to be redundant in the single-block case by Revenant *et al.* (2025), but is otherwise essential for ensuring optimality. `1pass` (used in aisle 3 and 5) and `2pass` describe that an aisle is completely passed through once or twice, respectively. `top` (`bottom`) means that an aisle is entered from the back (front) cross-aisle and left again at the same cross-

aisle after all required articles in the aisle have been collected. While `top` is used in aisle 6, `bottom` is applied in the first aisle. The action `gap` combines `top` and `bottom`, so an aisle is entered and exited from both cross-aisles while the largest part inside the aisle without any required articles (the so-called gap) is not visited. It is used in aisle 4, where the gap includes all pick positions between the SKUs 4 and 6. `void` indicates that an aisle is not visited at all, which is the case in aisle 2 of the example.

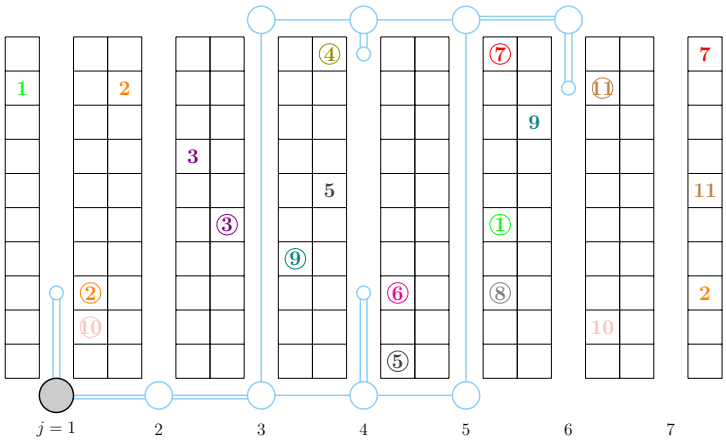
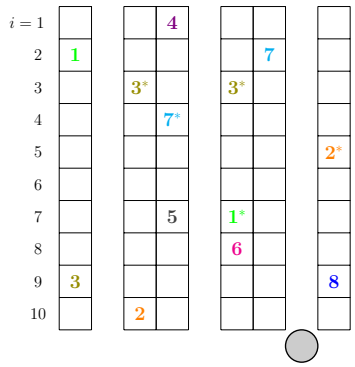


Figure 5.2: Example of an SPRP-SS instance, showing all possible aisle action types except `2pass`.

For the SPRP, each demanded article is collected from its assigned pick position. For the SPRP-SS, however, only a subset of the pick positions where the demanded articles are stored needs to be visited, as an article may be stored at multiple pick positions. As a result, for the SPRP-SS the actions `top`, `bottom`, and `gap` can be performed with different turning points in an aisle and are therefore no longer clearly defined. For this reason, an additional specification is required for each of these actions in order to indicate the turning point(s).

Figure 5.3 demonstrates the differences between the aisle actions for the state space of the SPRP and the SPRP-SS using a small instance. The instances for both problem types are illustrated in Figure 5.3a. While the colored numbers without asterisks indicate the unique pick positions of the demanded articles for the SPRP, the numbers with asterisks represent additionally scattered pick positions with the same articles available only for the SPRP-SS. Figure 5.3b lists all actions for the first aisle for both problems. In the traditional state space for the SPRP exists exactly one edge for each possible action type. Thus, there is a maximum of six edges for each aisle. In the given SPRP instance, articles $1, 2$, and 3 need to be collected in aisle 1, so an edge of type `void` is not feasible in the first aisle, and we



(a) An SPRP instance with extension into an SPRP-SS instance where the asterisks mark additional pick positions of the instance with scattered storage.

Type	E_1^{aisle} of SPRP	E_1^{aisle} of SPRP-SS
`1pass`	✓	✓
`2pass`	✓	✓
`top`(i)	Cell $i = 10$	$i \in \{2, 3, 9, 10\}$
`bottom`(i)	Cell $i = 2$	$i \in \{2, 3, 9, 10\}$
`gap`(h, i)	Cells $(h, i) = (2, 9)$	$(h, i) \in \{(2, 9), (2, 10), (3, 9), (3, 10)\}$
`void`	✗	✓

(b) Aisle actions of aisle 1 for the SPRP and the SPRP-SS in comparison. Actions marked in gray are dominated by cheaper actions.

Figure 5.3: Comparison of the aisle actions of a traditional state space (for SPRP) with the aisle actions of an extended state space resulting from the scattering of articles (for SPRP-SS).

have a total of five edges to traverse aisle 1. Concerning the SPRP-SS, there are several pick positions where the demanded articles can be collected. We assume the unit-demand case, where the demand for each article is always one. This means that only a single pick position needs to be visited for each article. For example, article 4 needs to be collected from a specific pick position, while articles 1 and 3 can be picked from two and three different pick positions, respectively. As a result, the extended state space for the SPRP-SS has multiple edges of the traditional state space for the SPRP, and there can be several edges with different turning points of the same action type `top`, `bottom`, and `gap` within an aisle. Looking at Figure 5.3b, these edges are listed in the rightmost column for the first aisle. All three articles $1, 2$, and 3 in the first aisle can also be collected from different pick positions in other aisles, so it is not mandatory to pick any of these articles in aisle 1. This results, for example, in three `top` edges with turning points at pick positions $2, 3$, and 10, denoted `top`(2), `top`(3), and `top`(10), respectively, which allow picking a subset of all articles in aisle 1. Note that in the unit-demand case, an additional edge with turning point 9 is dominated by the cheaper action `top`(3). The graph for the SPRP, in contrast, has only one `top` action with a turning point at position 10. Parallel edges and domination also occur with the

actions `bottom` and `gap`. Furthermore, for the SPRP-SS it is also feasible to collect none of the articles in aisle 1, which is realized with the action `void`. With general-demand instances, the state space often has fewer parallel edges. If the demand for an article can only be fulfilled if a specific pick position is visited (possibly in combination with other pick positions), only edges that represent aisle actions that visit this pick position are added to the state space for the corresponding aisle.

For the SPRP, an optimal picker tour is a shortest path in the state space of Ratliff and Rosenthal (1983) that can be computed with dynamic programming (DP) in linear time (Heßler and Irnich, 2022b). However, the SPRP-SS cannot be solved by dynamic programming over the extended state space. The reason for this is the parallel edges of the same aisle action within an aisle (see Figure 5.3b), which no longer ensure that the demand for each required article is met. Therefore, a shortest-path problem is modeled based on the extended space with additional constraints that guarantee demand covering. The model is solved with a MIP solver.

Set E_s includes all edges $e \in E$ that contain pick positions of article $s \in S$. The supply of article $s \in S$ associated with edge $e \in E$ is denoted by b_{se}. Besides, the cost c_e represents the length of the (cross-)aisle action that is described by edge $e \in E$. Heßler and Irnich (2024) use binary variables x_e for all $e \in E$ that indicate whether edge e is part of the solution. They propose the following formulation to model the SPRP-SS:

$$z_{SPRP-SS} = \min \sum_{e \in E} c_e x_e \tag{5.1a}$$

$$\text{subject to} \quad \sum_{e \in \delta^+(\sigma)} x_e - \sum_{e \in \delta^-(\sigma)} x_e = \begin{cases} +1, & \text{if } \sigma = o \\ -1, & \text{if } \sigma = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall \sigma \in V \tag{5.1b}$$

$$\sum_{e \in E_s} b_{se} x_e \geq q_s \qquad \forall s \in S \tag{5.1c}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{5.1d}$$

The objective function (5.1a) minimizes the picker tour length. Flow conservation is guaranteed by constraints (5.1b). For each node $\sigma \in V$ the set of outgoing edges is denoted by $\delta^+(\sigma)$ and the set of ingoing edges is described with $\delta^-(\sigma)$. The flow-conservation constraints can also be written in a compact form as $\mathcal{N}\mathbf{x} = \mathbf{u}_o - \mathbf{u}_d$. In this case, $\mathcal{N}$ describes the incidence matrix of the state space (V, E), $\mathbf{x} = (x_e)_{e \in E}$ the vector of the x-variables, and $\mathbf{u}_o$, $\mathbf{u}_d \in \{0,1\}^V$ the unit vectors of the origin and destination nodes of the state space. Constraints (5.1c) ensure that the demand is covered for all articles of the pick list. The domain of

the x-variables is given by (5.1d).

5.4 Picker Routing in Zoned Warehouses with Scattered Storage

In this section, we first formally define the MZPRP and introduce some notation before we present our modeling approach for this problem. We then introduce the BMZPRP including necessary model adaptions. Finally, we point out the limitations of our approach and show that the presented model can also be used to solve problem variants of the (B)MZPRP without extensive adaptions.

5.4.1 Definition of the Multi-Zone Picker Routing Problem

We consider a single-block warehouse with parallel aisles, divided into a set of disjoint zones $\mathcal{Z}$ such that each aisle always belongs to exactly one zone (see Figure 5.1). Each zone has an assigned picker and is equipped with a depot located at the intersection of a cross-aisle and an aisle. The order pickers begin and end their picker tours at the depot of their zone. A given a set of different articles S with a demand q_s for each article $s \in S$ is to be collected in the warehouse. The articles are distributed throughout the warehouse, each stored at one or more pick positions $p \in P$. The set P^z contains all pick positions of zone $z \in \mathcal{Z}$ and P_s^z is a subset of P^z that only includes pick positions where article $s \in S$ is stored. b_{sp}^z indicates the supply of article $s \in S$ at position $p \in P_s^z$ in zone $z \in \mathcal{Z}$. Furthermore, each picker has a capacity Q of a maximum number of different articles that can be picked and transported during a tour. The objective of the MZPRP is to find picker tours within the zones with a minimum total length such that the demand for all articles is satisfied and the picker capacities are respected. We assume that each picker is equipped with the same type of picking cart, so that the capacities of the pickers are identical. In the case of individual picker capacities, the subsequent model can be easily adapted.

The special case of the MZPRP with only one zone is identical to the SPRP-SS. Since the SPRP-SS is proven to be NP-hard by Weidinger (2018), and the MZPRP is a generalization of the SPRP-SS, the MZPRP is therefore also NP-hard.

5.4.2 Network-Flow Formulation

We use the extended state space for the SPRP-SS (Heßler and Irnich, 2024) and combine it with the new aspect of zoning. More precisely, for each zone $z \in \mathcal{Z}$ a distinct extended state space $(V, E)^z$ is constructed in order to identify an optimal path within each state space that minimizes the total length across all paths.

Each $o - d$ path in a state space represents a feasible picker routing tour in the corresponding zone in the sense that the tour starts and ends at the depot and only includes feasible (cross-)aisle actions. In addition, it must be ensured that the demand for all articles is satisfied and that the capacity of the pickers is not exceeded.

We denote by E_p^z the subset of edges E^z of zone $z \in \mathcal{Z}$ that includes pick position $p \in P^z$. Compared to the SPRP-SS, the cost of an edge is extended by the dimension of zones, i.e., c_e^z describes the length of the (cross-)aisle action that is depicted by edge $e \in E^z$ in zone $z \in \mathcal{Z}$. Accordingly, the binary variable x_e^z takes the value one if edge $e \in E^z$ of zone $z \in \mathcal{Z}$ is part of the picker tour, otherwise the value zero. Additional integer variables y_s^z for all $s \in S$ and $z \in \mathcal{Z}$ indicate the collected quantity of article s in zone z. In the case of unit demand, these y-variables are only binary. In this section, we formulate the model for the MZPRP. In the subsequent Section 5.4.3, the BMZPRP is described, and necessary adjustments to the formulation of the MZPRP are discussed. We propose the following network-flow formulation:

$$z_{MZPRP} = \min \sum_{z \in \mathcal{Z}} \sum_{e \in E^z} c_e^z x_e^z \tag{5.2a}$$

$$\text{subject to} \quad \mathcal{N}^z \mathbf{x}^z = \mathbf{u}_o^z - \mathbf{u}_d^z \qquad \forall z \in \mathcal{Z} \tag{5.2b}$$

$$\sum_{p \in P_s^z} \sum_{e \in E_p^z} b_{sp}^z x_e^z \geq y_s^z \qquad \forall s \in S, \forall z \in \mathcal{Z} \tag{5.2c}$$

$$\sum_{z \in \mathcal{Z}} y_s^z \geq q_s \qquad \forall s \in S \tag{5.2d}$$

$$\sum_{s \in S} y_s^z \leq Q \qquad \forall z \in \mathcal{Z} \tag{5.2e}$$

$$x_e^z \in \{0, 1\} \qquad \forall e \in E^z, \forall z \in \mathcal{Z} \tag{5.2f}$$

$$y_s^z \leq \sum_{p \in P_s^z} b_{sp}^z \qquad \forall s \in S, \forall z \in \mathcal{Z} \tag{5.2g}$$

$$y_s^z \in \mathbb{N}_0 \qquad \forall s \in S, \forall z \in \mathcal{Z} \tag{5.2h}$$

The objective function (5.2a) minimizes the total length of all picker tours within the zones. The flow-conservation constraints (5.2b) are built on those of the SPRP-SS (formulated in the compact form), except that they are extended by the zone dimension, as a separate state space is constructed for each zone $z \in \mathcal{Z}$. The collected quantity of an article within a zone must not exceed the available supply of this article in the zone, which is guaranteed by constraints (5.2c). Demand constraints (5.2d) make sure that the demand for each article is satisfied. Constraints (5.2e) ensure that the capacity of the picker is respected in each zone.

The domains of the variables are defined by (5.2f) - (5.2h). Note that (5.2g) is optional and can be omitted.

The size of formulation (5.2) is summarized below: Let us denote the number of different articles by $a = |S|$ and the total number of pick positions where those articles are stored by n. The number of variables is bounded by $\mathcal{O}(m+n^2+a\cdot|\mathcal{Z}|)$, which is composed as follows. For the SPRP-SS, the number of x-variables is bounded by $\mathcal{O}(m+n^2)$ (Heßler and Irnich, 2024), which remains the same for the MZPRP. Each state space has stages for the aisles of the zone where edges start and end, which together result in the summand m. The second summand n^2 relates to the parallel edges that emerge from scattered storage. In the worst case, all relevant pick positions are in a single aisle, which can lead to $\mathcal{O}(n^2)$ edges of type $\texttt{gap}(h,i)$. This also remains independent of the zoning aspect. The number of y-variables in a zone coincides with the number of different articles stored in this zone. Thus, there is a total of $a\cdot|\mathcal{Z}|$ y-variables.

The number of constraints is bounded by $\mathcal{O}((m+a+1)\cdot|\mathcal{Z}|+a)$. The flow-conservation constraints (5.2b) account for the addend $m\cdot|\mathcal{Z}|$, constraints (5.2c) and (5.2g) are constructed for each demanded article in each zone, contributing the summand $a\cdot|\mathcal{Z}|$. Moreover, constraints (5.2d) are required for each article s and constraints (5.2e) are built for each zone, i.e., a and $|\mathcal{Z}|$ constraints.

5.4.3 The Balanced Multi-Zone Picker Routing Problem

Minimizing the total length of all picker tours is the most common objective in the literature about warehousing (Weidinger *et al.*, 2019) and, thus, a reasoned objective for the MZPRP. As order picking costs can be responsible for more than half of the total warehouse operating expense (Tompkins *et al.*, 2010), shorter picker tours result in major savings. However, the route plan with minimal total length can lead to imbalanced picker tour lengths (Gu *et al.*, 2007). If the picker in one zone is allocated shorter tours on average than the pickers in the other zones, this leads to a lower work volume with unnecessary waiting times for that picker. This imbalance grows with the number of zones and reduces as the number of required articles increases (Gray *et al.*, 1992). Thus, we introduce the BMZPRP, which balances the picker tour lengths of all zones. Instead of minimizing the total picker tour length, we use a min-max objective that minimizes the length of the longest tour. This is known as the system throughput time for simultaneous zone picking (de Koster *et al.*, 2012). In the context of scheduling problems (Pinedo, 2022) and vehicle-routing problems (Irnich *et al.*, 2014) this kind of objective is also common.

To adapt the network-flow formulation, the objective function that minimizes the total length of all picker tours (5.2a) needs to be replaced by the following

min-max objective function:

$$z_{BMZPRP} = \min \quad T \tag{5.3a}$$

The continuous variable T represents the length of the longest picker tour which is defined in additional constraints of the network-flow formulation as

$$\sum_{e \in E^z} c_e^z x_e^z \leq T \qquad \forall z \in \mathcal{Z} \tag{5.3b}$$

$$T \geq 0. \tag{5.3c}$$

The resulting model consists of (5.3a) - (5.3c) together with (5.2b) - (5.2h), describing the BZPRP with a workload-balancing objective. The number of variables is virtually identical to the MZPRP, only the new variable T is added. The number of constraints increases by additional $|\mathcal{Z}|$ constraints and is bounded by $\mathcal{O}\left((m + a + 2) \cdot |\mathcal{Z}| + a\right)$.

5.4.4 Limitations and Possible Extensions of the Approach

Our approach has limitations regarding certain modeling assumptions. However, it is universally applicable and can handle problem variants without requiring substantial adjustments.

Our approach is limited to one picker per zone, as each state space determines one specific picker route. This is also a realistic assumption to prevent pickers from blocking each other. Besides, the capacity of the pickers is fixed to a certain quantity of collected items. However, adjustments to the model allow for flexibility in capacity formulation, such as formulating the capacity depending on the number of different SKUs, neglecting the collected quantity of each SKU. Moreover, as an alternative to a uniform capacity, capacities can easily be specified for each individual picker.

Although the model itself is fairly simple with demand, supply, and capacity constraints, the heart of the formulation is rather subtle with the state spaces for each zone. The fact that a separate state space is created for each zone reduces the problem to an SPRP-SS on the level of the zones with demand constraints on the global level of the complete warehouse including all zones. The main advantage of this approach is that the extended state space can be easily substituted by other state space concepts or adapted to other problem variants of the SPRP-SS. Therefore, our modeling approach can serve as a basis to describe other problem variants of the (B)MZPRP without the need for extensive adaptations.

Possible problem variants relate to the warehouse layout, the depot, or the routing (Heßler and Irnich, 2024, Table 1). Examples include multi-block warehouses

or warehouse layouts with non-parallel aisles, modification of the depot(s) resulting in picker tours with different start and end points, and routing strategies other than exact routing. Notably, our modeling approach allows the (B)MZPRP to accommodate such alternative routing strategies by generating a thinned-out state space (Lüke *et al.*, 2024) for each zone.

Instead of instances with uniformly distributed SKUs, storage assignment strategies can also be taken into account when generating instances. In addition to the skewed distribution of orders and demand discussed in Section 5.5.4, another option is to place articles that are likely to be ordered together in the same zone.

5.5 Computational Study

In this section, we first address the instances of the computational study. Therefore, we describe the warehouse layout and the parameters used for the instances, and explain the generation of the instances. We then present and analyze the results for the two problems. First, we focus on the MZPRP and, in addition, analyze the impact of the number of zones. Next, we examine the BMZPRP and end with a comparison of the MZPRP and the BMZPRP.

The algorithms are implemented in C++ with the callable library of CPLEX 20.1.0 and compiled into 64-bit single-thread release code using Microsoft Visual Studio 2022. The MIP solver uses all CPLEX default parameters except the time limit, the relative MIP gap tolerance (10^{-6}), and the setting to restrict the solver to a single thread. The computational experiments are conducted on a 64-bit machine running Microsoft Windows 11, equipped with an Intel® Core™ i7-5930K CPU at 3.5 GHz and 64 GB of RAM.

5.5.1 Instances

Since, to the best of our knowledge, neither the MZPRP nor the BMZPRP has yet been considered in the literature, there are no benchmark instances to date. Therefore, we have created a comprehensive set of instances for the unit-demand and general-demand cases which are used for both problems.

As already mentioned, we assume a one-block warehouse with parallel aisles and two cross-aisles at the back and the front of the aisles. The warehouse is divided into disjoint zones so that each aisle is assigned to exactly one zone, and each zone consists of adjacent aisles. Besides, all zones should have the same number of aisles, i.e., be the same size. If exact uniformity is not feasible, the zone sizes differ by no more than one aisle. The warehouse layout including the relevant distances is illustrated in Figure 5.1. The distance between two adjacent pick positions is one unit, as is the distance from the first or last position of an aisle

to the nearest cross-aisle. Two adjacent aisles have a distance of three units. The number of pick positions per aisle is fixed to 80, whereby the pick positions are divided between two racks with 40 positions each on the left and right side of the aisle. We assume the articles stored at the pick positions to be rather small, so it is possible to store different articles at the same pick position. The aisles are wide enough to allow the picker to change direction during a traversal of the aisle (for the aisle actions `top`, `bottom`, or `gap`), but the picker can reach the pick positions on both sides of the aisle without an increase in travel distance. Roodbergen and Vis (2006) showed that a depot located in the middle of a cross-aisle in a single-block warehouse minimizes the average picker tour lengths. Therefore, in each zone, we place the depot at the intersection of the middle aisle and the front cross-aisle. If the zone has an even number of aisles, the depot is located in front of the aisle immediately to the right of the center aisle. We define the picker capacity as a function of the number of different required articles and the number of zones, namely as $Q = \lceil a/|\mathcal{Z}| \times 1.5 \rceil$. On the one hand, this approach ensures distributing the workload among multiple pickers. On the other hand, the capacity is sufficiently large to avoid implicitly constraining the solution by predefining the pick list size for each picker. The scatter factor is set to $\alpha = 2$, meaning each article is stored on average at two different pick positions in the warehouse. The articles are assigned randomly to the pick positions throughout the warehouse. In unit-demand instances, the demand for each article is always one. In general-demand instances, the demand for each article is between one and five, with a decreasing distribution such that the demand is one (five) with a probability of 60% (5%). The supply of each SKU available at a specific pick position is uniformly distributed between one and three.

We vary several parameters to get different instance settings to allow detailed analysis of the results. As shown in Table 5.2, we generated instances with 10 to 60 aisles, one to five zones, and 10 to 200 articles to be picked. For each combination of parameter values, 50 instances were created. This results in a total of 5 000 different instances for both the unit-demand and the general-demand cases. The instances are available online at https://logistik.bwl.uni-mainz.de/research/benchmarks/.

Due to the picker capacity in each zone, infeasible instances may arise, specifically when the number of articles to be collected exceeds the available capacity in a zone, and the articles are not sufficiently stored in other zones. To ensure the feasibility of the instances, we solved a *constraint satisfaction problem* in advance for each generated instance using the MIP solver. This problem consists of the demand constraints (5.2d), the capacity constraints (5.2e), and the constraints (5.2g) and (5.2h) to define the domain of the y-variables. If an instance was found that did not comply with these constraints, it was discarded and not considered

Parameter	Value(s)
Number of aisles m	$10, 20, 35, 60$
Number of pick positions per aisle	80 (2×40)
Number of zones $\lvert\mathcal{Z}\rvert$	$1, 2, 3, 4, 5$
Number of articles a	$10, 25, 50, 100, 200$
Picker capacity Q	$\lceil a/\lvert\mathcal{Z}\rvert \times 1.5\rceil$
Scatter factor α	2
Number of instances per setting	50

Table 5.2: Parameters used for the generation of the instances.

in our study. However, this only occurred a few times, with ten different articles to collect.

5.5.2 Results for the Multi-Zone Picker Routing Problem

We first discuss the results of the MZPRP. Table 5.3 shows the average computation times in seconds for the MZPRP for the different parameter values of the number of aisles m in the warehouse, the number of zones (synonymous with the number of pickers) $|\mathcal{Z}|$, and the number of required articles a. Furthermore, we distinguish between the unit-demand case and the general-demand case. The time measurement includes the generation of the MIP model and its solution by the MIP solver. Values are rounded to two decimal places.

Note that each of the 10 000 instances can be solved optimally within few (milli-)seconds, which highlights the effectiveness of our MIP-based approach for solving the MZPRP. The number of zones does not seem to have a noticeable impact on the computation times. It is striking that the computation times for unit demand are many times longer than for general demand. The difference in computation time grows with an increasing number of articles. This is because, in the unit-demand case, only one pick position has to be frequented for each article. In contrast, in the general-demand case, several pick positions may need to be visited to satisfy the demand for an article. The necessity of visiting several pick positions for an article reduces the number of feasible aisle actions. This is the case if some pick positions of an article must be visited and are not optional to fulfill the demand of the article. As a result, fewer parallel edges correspond to aisle actions in the state space for the general-demand case than in the unit-demand case. The effect becomes particularly clear the more articles there are to collect. Since each edge in the state space corresponds to an x-variable in the MIP model, this relationship can also be observed when evaluating the number of variables of the models generated from the instances of the computational study. Table 5.4

shows the ratio of the average number of the x-variables of unit demand to general demand. Consistent with the reasoning just given, all values are ≥ 1, since unit demand results in at least as many parallel edges as general demand. It can also be noted that the values increase the more articles need to be collected. For the unit-demand instances with 200 articles, on average 2.41 times as many x-variables are generated as for the general-demand instances, which in turn explains the different computation times for these two cases. The number of the y-variables is identical in the unit-demand case and general-demand case and is therefore not considered.

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	0.02	0.05	0.16	0.65	2.41	0.01	0.02	0.05	0.09	0.17
	2	0.02	0.06	0.17	0.83	2.45	0.02	0.03	0.05	0.09	0.20
	3	0.01	0.05	0.16	0.48	2.12	0.01	0.02	0.04	0.07	0.16
	4	0.01	0.04	0.09	0.23	0.55	0.01	0.02	0.03	0.06	0.11
	5	0.01	0.03	0.05	0.10	0.24	0.01	0.01	0.02	0.04	0.07
20	1	0.02	0.04	0.17	0.73	12.77	0.02	0.03	0.06	0.12	0.20
	2	0.03	0.06	0.17	0.93	13.74	0.02	0.03	0.06	0.12	0.23
	3	0.02	0.07	0.19	0.97	9.03	0.02	0.03	0.07	0.11	0.21
	4	0.02	0.08	0.20	0.86	11.86	0.02	0.03	0.06	0.11	0.20
	5	0.02	0.07	0.21	0.86	6.94	0.02	0.03	0.06	0.09	0.18
35	1	0.02	0.04	0.09	0.54	10.96	0.02	0.03	0.05	0.13	0.25
	2	0.03	0.06	0.12	0.54	11.04	0.03	0.04	0.05	0.11	0.24
	3	0.03	0.07	0.15	0.64	15.63	0.03	0.04	0.06	0.12	0.26
	4	0.04	0.07	0.17	0.75	15.36	0.02	0.04	0.06	0.12	0.24
	5	0.04	0.12	0.18	0.78	16.98	0.03	0.04	0.08	0.11	0.21
60	1	0.03	0.06	0.10	0.24	4.46	0.03	0.04	0.06	0.09	0.27
	2	0.05	0.07	0.12	0.28	5.62	0.03	0.05	0.06	0.10	0.23
	3	0.05	0.08	0.11	0.35	6.43	0.04	0.04	0.06	0.10	0.24
	4	0.06	0.07	0.13	0.43	11.21	0.03	0.06	0.06	0.10	0.23
	5	0.06	0.10	0.18	0.55	10.36	0.05	0.05	0.07	0.10	0.22

Table 5.3: Average computation time in seconds for the MZPRP.

Another observation that can be made from Table 5.3 is that for instances with many required articles, especially in the unit-demand case, the computation times first increase with an augmenting number of aisles but then decrease for warehouse layouts with 60 or already 35 aisles. A possible explanation for this phenomenon relates to the fact that the total number of edges is bounded by $\mathcal{O}(m+n^2)$. In the

m	$\lvert\mathcal{Z}\rvert$	$a = 10$	$a = 25$	$a = 50$	$a = 100$	$a = 200$
10	1	1.11	1.36	1.67	2.46	3.56
	2	1.11	1.35	1.64	2.48	3.55
	3	1.11	1.34	1.69	2.42	3.80
	4	1.13	1.36	1.65	2.29	3.73
	5	1.09	1.36	1.59	2.43	3.73
20	1	1.05	1.15	1.31	1.74	2.53
	2	1.06	1.16	1.31	1.76	2.55
	3	1.06	1.14	1.30	1.70	2.63
	4	1.05	1.16	1.31	1.75	2.58
	5	1.04	1.16	1.30	1.74	2.59
35	1	1.03	1.08	1.16	1.40	1.88
	2	1.03	1.08	1.17	1.39	1.88
	3	1.03	1.08	1.15	1.38	1.90
	4	1.03	1.09	1.18	1.40	1.91
	5	1.03	1.08	1.16	1.40	1.92
60	1	1.02	1.04	1.08	1.21	1.50
	2	1.02	1.05	1.09	1.22	1.50
	3	1.02	1.04	1.08	1.21	1.50
	4	1.02	1.04	1.09	1.20	1.49
	5	1.02	1.04	1.08	1.20	1.50
Total		1.05	1.16	1.30	1.69	2.41

Table 5.4: Number of the x-variables generated from the MZPRP instances of the unit-demand case in relation to the general-demand case.

worst case, all articles are stored in only one aisle of the warehouse. As a result, the aisle actions of the type `gap` can lead to a high number of edges in the state space, which is quadratic in the number of relevant pick positions ($\mathcal{O}(n^2)$) (Heßler and Irnich, 2024). However, if the articles are distributed over many aisles, the effect becomes insignificant. In the unit-demand case, instances with up to 20 or 35 aisles and 200 articles have a high density of relevant pick positions per aisle and, therefore, a very high number of edges in the state spaces, leading to longer computation times. However, instances with only 10 aisles are solved relatively quickly within 2.5 seconds or less. We observed that the routes of instances with 10 aisles and 200 articles mainly consist of the aisle action `1pass` as many pick positions in the few aisles need to be visited. Perhaps these types of routes can be found easier by the solver than more complex routes. In the case of general demand, the effect is much weaker due to the smaller number of parallel edges. As the number of aisles increases, the relevant pick positions are distributed over more aisles, and the number of parallel edges in the state spaces for unit demand and

general demand becomes increasingly similar. Again, this effect can also be seen in Table 5.4, where the number of variables of the unit-demand case in relation to the general-demand case is highest for 10 aisles and decreases with an augmenting number of aisles.

In the following, we analyze the average cost of the MZPRP. Table 5.5 has the same structure as Table 5.3 and presents the average optimal objective function values since all instances were solved optimally. Values are rounded to one decimal place. In general, the average cost rises as the number of articles or aisles increases. This is because more pick positions must be visited during the route, or the pick positions to be visited are spread over a larger area. It is also noticeable that the costs for the unit-demand case are, on average, lower than the costs for the general-demand case. This can be logically justified, as in the unit-demand case the demand for an article is always one, and the picker, therefore, only has to visit a single pick position for each article. In the case of general demand, on the contrary, the supply at a pick position may not be sufficient to fully satisfy the demand for an article, which might require the picker to visit several pick positions to satisfy the demand for an article.

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	156.2	235.2	301.9	357.3	403.3	189.0	289.3	364.0	429.7	460.8
	2	165.5	239.1	299.8	352.9	398.9	205.0	291.8	363.6	426.4	466.4
	3	181.5	254.7	302.6	353.7	399.4	217.7	310.2	368.2	428.2	469.8
	4	191.4	266.5	315.7	364.7	398.5	236.0	325.0	386.0	425.1	465.6
	5	205.4	285.5	329.1	376.4	412.8	237.8	338.1	403.0	435.3	440.0
20	1	218.3	344.3	468.2	596.1	711.5	260.6	421.9	574.9	732.1	853.6
	2	219.4	341.6	462.9	589.7	704.9	265.4	417.7	568.6	728.2	857.1
	3	225.7	354.4	467.9	584.4	694.6	276.9	429.5	563.0	722.6	852.7
	4	230.7	354.3	466.8	576.9	694.2	283.0	445.2	570.2	719.8	846.7
	5	241.6	375.5	474.1	587.0	695.6	284.4	448.1	581.4	726.9	854.6
35	1	284.6	463.3	649.5	862.6	1083.0	339.8	558.0	788.5	1061.7	1326.6
	2	278.1	445.8	643.3	848.6	1066.2	324.8	558.2	767.1	1049.2	1339.2
	3	278.2	449.7	633.3	850.8	1070.6	342.9	547.0	777.0	1039.6	1317.0
	4	281.2	450.1	635.0	840.7	1059.0	328.6	563.4	769.3	1044.0	1314.5
	5	283.7	450.1	636.0	840.4	1059.2	328.7	564.9	765.0	1042.3	1324.8
60	1	389.0	619.4	875.2	1192.4	1571.2	451.6	726.6	1025.5	1449.4	1924.4
	2	366.6	590.7	849.3	1184.7	1555.1	420.8	712.2	1011.0	1429.6	1916.6
	3	344.4	580.1	840.5	1176.6	1548.1	414.6	707.0	1018.0	1447.6	1922.9
	4	340.0	574.0	832.2	1162.6	1542.5	406.4	708.2	1002.7	1430.6	1923.4
	5	342.4	578.8	827.3	1151.6	1536.3	399.2	702.0	1005.6	1424.9	1930.9

Table 5.5: Average cost of the MZPRP.

We use the data from Table 5.6 to analyze the impact of the number of zones

on the total length of the picker routes. The table shows the percentage difference in average cost between the SPRP-SS (=MZPRP with one zone) and the MZPRP with multiple zones. For each parameter combination of the number of aisles and articles, the average percentage cost difference of the instances with two to five zones is shown compared to the single-zone instances. Thus, negative values mean that the MZPRP has lower costs than the SPRP-SS or that zoning reduces costs. Note that the instance setting resulting in the lowest total length is written in bold. Table 5.6 reveals that in most settings, cost savings in the single-digit percentage range can be achieved by using two or more zones. This represents great potential, as order picking accounts for the majority of warehouse operating costs (Bartholdi and Hackman, 2019), and therefore, even small savings for individual routes add up to significant savings overall. Small instances with no more than 25 articles and 20 aisles usually have the lowest costs with only one zone. On the contrary, zoning can achieve by far the greatest savings for instances with 60 aisles and 10 articles. More precisely, in the unit-demand case (general-demand case), picking in four (five) zones can save 12.6% (11.6%) of the cost compared to the case in which the warehouse is a single zone.

Overall, in both the unit-demand case and the general-demand case, a higher number of zones is beneficial as the number of aisles increases. The same is true for an increasing number of articles, although the effect is diminished in the case of general demand for instance settings with a large number of aisles because of the very small percentage cost difference. In general, the average percentage cost differences due to the number of zones are slightly lower in the general-demand case than in the unit-demand case. In the general-demand case, visiting multiple pick positions per article may be necessary, limiting the options for collecting sufficient quantities. In contrast, the unit-demand case always requires only one pick position per article, providing more picking options and allowing greater cost savings by introducing more zones.

5.5.3 Results for the Balanced Multi-Zone Picker Routing Problem

In this section, we consider the BMZPRP with the objective of minimizing the length of the largest picker tour, which was introduced in Section 5.4.3. Table 5.7 shows the average computation time in seconds for the BMZPRP. The structure of the table is the same as that of the previous tables. The time measurement again includes the generation of the MIP model and its solution by the MIP solver. In contrast to the MZPRP, some instances with 200 articles cannot be solved optimally within the time limit of 600s. The number of affected instances is noted in the column *tl* (=time limit), and these instances are included in the average

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	**0.0%**	**0.0%**	0.0%	0.0%	0.0%	**0.0%**	**0.0%**	0.0%	0.0%	0.0%
	2	6.0%	1.7%	**-0.7%**	**-1.2%**	-1.1%	8.4%	0.9%	**-0.1%**	-0.8%	1.2%
	3	16.2%	8.3%	0.2%	-1.0%	-1.0%	15.2%	7.2%	1.2%	-0.4%	2.0%
	4	22.6%	13.3%	4.6%	2.1%	**-1.2%**	24.9%	12.3%	6.1%	**-1.1%**	1.1%
	5	31.5%	21.4%	9.0%	5.3%	2.4%	25.8%	16.9%	10.7%	1.3%	**-4.5%**
20	1	**0.0%**	0.0%	0.0%	0.0%	0.0%	**0.0%**	0.0%	0.0%	0.0%	0.0%
	2	0.5%	**-0.8%**	**-1.1%**	-1.1%	-0.9%	1.9%	**-1.0%**	-1.1%	-0.5%	0.4%
	3	3.4%	2.9%	-0.1%	-2.0%	-2.4%	6.3%	1.8%	**-2.1%**	-1.3%	-0.1%
	4	5.7%	2.9%	-0.3%	**-3.2%**	**-2.4%**	8.6%	5.5%	-0.8%	**-1.7%**	**-0.8%**
	5	10.7%	9.1%	1.3%	-1.5%	-2.2%	9.2%	6.2%	1.1%	-0.7%	0.1%
35	1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	2	**-2.3%**	**-3.8%**	-1.0%	-1.6%	-1.5%	**-4.4%**	0.0%	-2.7%	-1.2%	0.9%
	3	-2.3%	-2.9%	**-2.5%**	-1.4%	-1.1%	0.9%	**-2.0%**	-1.5%	**-2.1%**	-0.7%
	4	-1.2%	-2.9%	-2.2%	-2.5%	**-2.2%**	-3.3%	1.0%	-2.4%	-1.7%	**-0.9%**
	5	-0.3%	-2.9%	-2.1%	**-2.6%**	-2.2%	-3.3%	1.2%	**-3.0%**	-1.8%	-0.1%
60	1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	2	-5.8%	-4.6%	-3.0%	-0.6%	-1.0%	-6.8%	-2.0%	-1.4%	-1.4%	**-0.4%**
	3	-11.5%	-6.4%	-4.0%	-1.3%	-1.5%	-8.2%	-2.7%	-0.7%	-0.1%	-0.1%
	4	**-12.6%**	**-7.3%**	-4.9%	-2.5%	-1.8%	-10.0%	-2.5%	**-2.2%**	-1.3%	-0.1%
	5	-12.0%	-6.6%	**-5.5%**	**-3.4%**	**-2.2%**	**-11.6%**	**-3.4%**	-1.9%	**-1.7%**	0.3%

Table 5.6: Percentage cost difference between the SPRP-SS and the MZPRP.

computation time with 600s.

To some extent, the required computation time to solve the BMZPRP behaves the same as the MZPRP. This includes the fact that general-demand instances can be solved significantly faster than unit-demand instances and that the computation times increase with the number of articles. But there are also differences compared to the MZPRP concerning the computation time. In particular, for instance settings with many articles and aisles, the computation time heavily increases with a growing number of zones, which is not the case for the MZPRP. As mentioned earlier, minimizing the length of the largest route does not change the number of variables and constraints in the MIP model much compared to minimizing the total length. Therefore, it seems that the objective function of the BMZPRP (5.3a) in combination with the additional constraints (5.3b) and (5.3c) makes the MIP significantly more difficult to solve than the MIP of the MZPRP.

Table 5.8 shows the average length of the largest picker route. Instances that were not solved to optimality within the time limit of 600s are listed in the column *tl* and are not included in the calculation of the average cost. As expected, the route lengths increase as both the number of articles and the number of aisles increase. In addition, the average length of the largest picker route decreases the more zones the warehouse has. For the same reasons as for the MZPRP, the routes

		unit demand						general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	tl	$a = 10$	25	50	100	200
10	1	0.02	0.05	0.17	0.70	2.44		0.02	0.03	0.05	0.09	0.16
	2	0.04	0.12	0.43	1.76	3.87		0.03	0.04	0.07	0.12	0.26
	3	0.03	0.07	0.22	0.61	1.78		0.02	0.03	0.05	0.09	0.20
	4	0.02	0.06	0.09	0.23	0.81		0.02	0.02	0.04	0.07	0.14
	5	0.02	0.04	0.07	0.14	0.29		0.02	0.02	0.03	0.04	0.08
20	1	0.02	0.04	0.16	0.75	12.97		0.02	0.03	0.06	0.12	0.19
	2	0.10	0.18	0.47	2.55	44.75		0.04	0.07	0.14	0.24	0.31
	3	0.12	0.24	0.54	2.91	27.55		0.05	0.08	0.16	0.21	0.31
	4	0.09	0.25	0.53	4.77	123.27	4	0.04	0.10	0.12	0.25	0.30
	5	0.09	0.17	0.46	1.88	111.89	3	0.04	0.06	0.11	0.21	0.24
35	1	0.02	0.04	0.09	0.49	12.62		0.02	0.03	0.05	0.13	0.25
	2	0.15	0.23	0.47	1.78	40.20		0.07	0.10	0.16	0.29	0.57
	3	0.23	0.32	0.68	4.04	70.63		0.11	0.13	0.27	0.55	0.68
	4	0.27	0.41	0.90	9.61	128.38	1	0.10	0.19	0.32	0.76	0.88
	5	0.22	0.39	1.40	10.63	280.07	9	0.11	0.20	0.35	0.75	0.95
60	1	0.03	0.05	0.09	0.22	4.78		0.03	0.04	0.06	0.10	0.27
	2	0.16	0.30	0.41	1.18	22.73		0.10	0.13	0.22	0.34	0.67
	3	0.19	0.43	0.89	3.20	60.48		0.10	0.25	0.44	0.83	1.80
	4	0.26	0.64	1.91	9.51	151.87	2	0.13	0.29	0.68	1.75	4.24
	5	0.29	0.77	4.10	22.15	285.11	4	0.14	0.31	0.87	2.67	4.37

Table 5.7: Average computation time in seconds for the BMZPRP.

in the general-demand case are at least as long as in the unit-demand case.

5.5.4 Managerial Insights

While Sections 5.5.2 and 5.5.3 concentrated primarily on the algorithmic aspects of the two problems, we now turn to the analysis of their practical applications. First, we compare the problems with respect to their differing objective functions. Subsequently, we analyze both problems using more realistic instances characterized by a skewed order distribution and demand.

To compare the average total costs between the MZPRP and the BMZPRP, the objective function of the BMZPRP must be modified slightly. Currently, the model minimizes the length of the largest tour of an instance, but the tours of all other zones are not necessarily optimized. However, this is essential for a comparison of the total costs. Therefore, the objective function (5.3a) of the BMZPRP is replaced by the following:

		unit demand						general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	tl	$a = 10$	25	50	100	200
10	1	156.2	235.2	301.9	357.3	403.3		189.0	289.3	364.0	429.7	460.8
	2	97.0	127.7	158.4	186.2	206.0		113.2	157.2	193.6	221.7	238.4
	3	85.2	98.8	113.1	129.8	147.6		94.5	120.8	139.2	169.4	181.4
	4	75.4	88.4	93.0	105.4	117.8		85.0	99.3	118.3	133.0	149.6
	5	69.6	85.8	88.0	88.0	88.0		78.2	87.6	88.0	88.0	88.0
20	1	218.3	344.3	468.2	596.1	711.5		260.6	421.9	574.9	732.1	853.6
	2	124.8	179.7	241.5	303.1	359.5		144.5	223.5	293.4	376.6	437.8
	3	98.6	131.6	169.0	208.4	244.6		113.2	164.0	206.3	262.8	306.2
	4	85.4	108.4	130.4	157.0	186.4	4	98.6	133.4	163.6	199.5	225.6
	5	79.4	96.9	110.1	129.8	152.8	3	88.6	114.3	136.0	166.0	181.8
35	1	284.6	463.3	649.5	862.6	1083.0		339.8	558.0	788.5	1061.7	1326.6
	2	154.9	233.6	330.8	432.1	542.5		178.0	294.6	400.5	537.6	684.8
	3	117.8	165.0	223.1	295.0	367.2		138.9	200.4	277.8	364.6	458.8
	4	99.3	134.0	171.8	224.6	278.2	1	110.3	164.3	215.6	288.6	352.3
	5	87.3	113.4	143.1	182.0	225.2	9	99.9	136.8	175.7	229.5	288.0
60	1	389.0	619.4	875.2	1192.4	1571.2		451.6	726.6	1025.5	1449.4	1924.4
	2	203.7	312.4	436.8	600.5	783.3		234.6	376.7	516.7	725.8	969.6
	3	140.4	212.4	294.3	402.8	525.8		169.1	257.4	358.5	500.2	657.3
	4	115.4	165.6	224.4	304.7	396.1	2	137.0	203.7	275.9	376.4	501.4
	5	101.0	138.8	182.4	244.8	320.8	4	118.0	168.4	223.6	310.0	409.9

Table 5.8: Average cost of the largest route of the BMZPRP.

$$z_{BMZPRP^*} = \min \quad T + \lambda \sum_{z \in \mathcal{Z}} \sum_{e \in E^z} c_e^z x_e^z \tag{5.4a}$$

The weight λ is set to 0.0001. Thus, adding the total costs has no significant influence on the objective function value. Table 5.9 compares the average total costs of the two problems. The values show the ratio of the average costs of the BMZPRP to the MZPRP.

Overall, it is noticeable that balanced route lengths increase the total length only slightly compared to minimizing the total length. The instance setting with 20 aisles, five zones, and 10 articles has the highest average cost increase at 16%, while the average cost increase across all instances is only 4%. In general, a warehouse layout with several zones increases the total length of the BMZPRP compared to the MZPRP. A possible explanation is that a higher number of tours in the warehouse potentially leads to higher additional costs if the objective is to balance the tour lengths by minimizing the largest picker tour. However, in a warehouse with only one or two zones, there is no or only one additional route to the optimized route that can contribute to additional costs.

In addition, the total length of the BMZPRP is proportionally lower when the number of articles is large. This is because, with many articles, the single routes of the MZPRP become larger and more balanced, and therefore the additional cost for fully balanced routes is lower. This effect can also be observed in a weaker form when comparing unit demand with general demand. On average larger picker routes in the general-demand case lead to slightly lower additional costs than in the unit-demand case.

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.09	1.03	1.03	1.03	1.01	1.03	1.03	1.02	1.01	1.00
	3	1.10	1.07	1.04	1.04	1.06	1.06	1.02	1.02	1.02	1.00
	4	1.08	1.09	1.07	1.03	1.01	1.05	1.04	1.02	1.01	1.00
	5	1.12	1.05	1.00	1.00	1.00	1.04	1.00	1.00	1.00	1.00
20	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.08	1.03	1.03	1.02	1.01	1.03	1.03	1.02	1.02	1.00
	3	1.12	1.05	1.05	1.04	1.04	1.06	1.04	1.04	1.03	1.01
	4	1.12	1.11	1.06	1.05	1.04	1.09	1.04	1.03	1.03	1.01
	5	1.16	1.14	1.08	1.04	1.06	1.08	1.07	1.03	1.04	1.00
35	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.07	1.03	1.02	1.01	1.01	1.03	1.03	1.02	1.01	1.01
	3	1.11	1.06	1.04	1.03	1.02	1.05	1.03	1.04	1.03	1.02
	4	1.13	1.10	1.05	1.04	1.03	1.11	1.06	1.05	1.04	1.03
	5	1.15	1.12	1.06	1.05	1.04	1.08	1.06	1.05	1.04	1.02
60	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.07	1.04	1.02	1.01	1.01	1.05	1.04	1.02	1.01	1.01
	3	1.10	1.05	1.04	1.02	1.01	1.06	1.04	1.04	1.02	1.02
	4	1.13	1.10	1.05	1.04	1.02	1.07	1.05	1.05	1.04	1.03
	5	1.14	1.10	1.06	1.04	1.03	1.07	1.06	1.05	1.04	1.03

Table 5.9: The average total length of the BMZPRP compared to that of the MZPRP, excluding instances not solved within the time limit.

Table 5.10 shows the average length of the largest picker tour for all instances of the MZPRP compared with the BMZPRP. On average, the largest picker tour of the MZPRP is 9% longer than that of the BMZPRP. The greatest increase in length, 30%, occurs with 60 aisles, a pick list size of 25, and five zones. In general, the difference in length increases as the number of aisles increases. In large ware-

houses with many aisles, tours per zone increase, resulting in greater differences in length between two feasible tours of a zone. Additionally, the difference in length is greater for the unit-demand case than for the general-demand case. Notably, general-demand instances with a large pick list size of 200 and a small warehouse layout with only 10 or 20 aisles have similar lengths for the largest routes of the MZPRP and BMZPRP. This is because each zone is completely traversed in both cases due to the high number of articles to be picked in a small space.

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.15	1.06	1.11	1.06	1.03	1.05	1.04	1.04	1.01	1.00
	3	1.09	1.13	1.09	1.11	1.17	1.06	1.05	1.05	1.03	1.00
	4	1.10	1.11	1.18	1.09	1.04	1.07	1.07	1.04	1.01	1.00
	5	1.11	1.02	1.00	1.00	1.00	1.06	1.01	1.00	1.00	1.00
20	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.16	1.10	1.12	1.09	1.05	1.08	1.08	1.06	1.03	1.01
	3	1.15	1.13	1.15	1.15	1.13	1.11	1.09	1.10	1.06	1.02
	4	1.14	1.20	1.17	1.17	1.14	1.11	1.09	1.09	1.05	1.01
	5	1.13	1.19	1.20	1.13	1.17	1.11	1.13	1.08	1.07	1.00
35	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.18	1.11	1.14	1.09	1.09	1.09	1.11	1.08	1.04	1.03
	3	1.15	1.17	1.15	1.13	1.11	1.08	1.08	1.13	1.09	1.06
	4	1.21	1.23	1.20	1.18	1.15	1.19	1.16	1.14	1.10	1.05
	5	1.16	1.23	1.24	1.21	1.18	1.13	1.19	1.15	1.12	1.05
60	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.17	1.17	1.13	1.08	1.05	1.08	1.10	1.09	1.05	1.04
	3	1.18	1.21	1.18	1.15	1.12	1.09	1.11	1.13	1.10	1.07
	4	1.20	1.26	1.21	1.21	1.15	1.14	1.14	1.15	1.14	1.09
	5	1.18	1.30	1.28	1.24	1.20	1.13	1.18	1.16	1.15	1.09

Table 5.10: The average length of the largest tour of an instance of the MZPRP compared to that of the BMZPRP, excluding instances not solved within the time limit.

The instances we analyzed so far have a uniform order distribution and demand, and the scatter factor across all articles is $\alpha = 2$. This means that every article in the warehouse has the same probability of being ordered and thus appearing on the pick list. Besides, on average, each article is stored at two pick positions in the

warehouse. Another possibility that is closer to actual practice is to consider instances with a skewed distribution of orders and demand. In such settings, a small fraction of articles in the warehouse accounts for the majority of orders, whereas the remaining articles are slow-moving and requested infrequently. Furthermore, high-demand articles are stored at multiple pick positions, as indicated by a higher scatter factor, compared with articles that are ordered less frequently. We investigated the zone picking problems also based on such instances with skewed order distribution and demand. Concerning the structure of the instances and the generation procedure, we followed the approach of Weidinger (2018), Goeke and Schneider (2021), and Heßler and Irnich (2024). For the generation procedure, we assume a fully occupied warehouse with one article assigned to each pick position. A scatter factor is enforced by first placing $\xi = \lceil mC/\alpha \rceil \geq a$ distinct articles, each assigned to a unique pick position. These articles are then partitioned into three classes reflecting turnover rates. We assume 20% of the articles are fast-moving (A-articles) and 50% of the articles are slow-moving (C-articles), the remaining articles (30%) are somewhere in between and are assigned to class B. Then, additional SKUs are allocated sequentially to the still free pick positions in such a way that with a probability of 80% an article of class A is assigned to a pick position, and 15% (5%) of all pick positions in the warehouse are occupied with B-articles (C-articles). Finally, a pick list containing exactly a distinct articles is constructed by randomly selecting positions and adding previously unselected articles until the required size is reached. The parameters described in Section 5.5.1 remain the same for these instances with classified articles.

The major difference between instances with skewed order distribution and demand and the previous instances is that the scatter factor is no longer the same for all articles. Instead, articles with a higher demand have a larger scatter factor, and articles with lower demand have a smaller scatter factor. To analyze the effect of this heterogeneous scatter factor, we consider Table 5.11. Structured like Table 5.6, it shows the percentage cost difference between the SPRP-SS and the MZPRP with different numbers of zones, but this time for the instances with skewed order distribution and demand. The percentage cost difference behaves similarly to Table 5.6. More specifically, zones are worthwhile in most settings compared to SPRP-SS with only one zone. Especially with many aisles and small or medium pick list sizes, the savings are significant, with up to 15.5% in the unit-demand case and 14.7% in the general-demand case. In these cases, the savings are slightly higher than in the instances investigated first.

		unit demand					general demand				
m	$\lvert\mathcal{Z}\rvert$	$a = 10$	25	50	100	200	$a = 10$	25	50	100	200
10	1	**0.0%**	**0.0%**	0.0%	0.0%	0.0%	**0.0%**	**0.0%**	0.0%	0.0%	0.0%
	2	6.7%	2.3%	**-0.8%**	-0.4%	6.9%	7.1%	1.2%	**-0.7%**	-0.1%	7.1%
	3	13.6%	7.9%	1.1%	**-0.4%**	6.6%	14.3%	5.9%	-0.2%	-0.1%	7.0%
	4	17.2%	14.9%	8.1%	0.0%	5.2%	17.5%	13.2%	5.5%	-0.2%	5.4%
	5	24.1%	21.0%	14.2%	0.0%	**-5.2%**	21.4%	20.6%	10.7%	**-1.6%**	**-5.2%**
20	1	**0.0%**	0.0%	0.0%	0.0%	0.0%	**0.0%**	0.0%	0.0%	0.0%	0.0%
	2	0.0%	**-1.1%**	-1.1%	-0.5%	-0.3%	1.5%	**-1.2%**	-0.8%	-0.5%	-0.3%
	3	1.7%	-0.6%	**-1.1%**	-0.9%	-0.7%	3.6%	0.2%	**-1.3%**	-0.8%	-0.5%
	4	4.6%	2.7%	0.8%	**-1.1%**	-0.9%	8.3%	2.6%	0.2%	-1.1%	-0.5%
	5	12.8%	6.3%	3.5%	-0.3%	**-1.2%**	12.8%	6.1%	2.8%	**-1.2%**	**-1.2%**
35	1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	2	-3.2%	-3.8%	-1.7%	-0.4%	-0.3%	-2.6%	-2.6%	-1.0%	-0.5%	-0.2%
	3	**-8.2%**	**-5.4%**	**-2.6%**	-0.7%	-0.4%	**-7.6%**	**-4.8%**	**-1.9%**	-0.8%	-0.3%
	4	-5.5%	-5.1%	-2.4%	-1.0%	-0.6%	-4.1%	-3.8%	-1.9%	-1.0%	-0.5%
	5	-2.7%	-3.0%	-1.4%	**-1.4%**	**-0.7%**	-4.7%	-2.1%	-1.2%	**-1.3%**	**-0.6%**
60	1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	2	-7.4%	-4.7%	-2.2%	-0.8%	-0.2%	-5.9%	-3.9%	-1.5%	-0.6%	-0.2%
	3	-11.6%	-9.2%	-3.7%	-1.2%	-0.3%	-9.7%	-7.5%	-3.0%	-0.8%	-0.2%
	4	**-15.5%**	-9.7%	-4.4%	-1.9%	-0.4%	**-14.7%**	**-8.2%**	-3.4%	-1.5%	-0.4%
	5	-11.6%	**-9.8%**	**-4.5%**	**-2.6%**	**-0.8%**	-13.2%	-7.1%	**-4.0%**	**-2.0%**	**-0.8%**

Table 5.11: Percentage cost difference between the SPRP-SS and the MZPRP with skewed order distribution and demand according to the ABC analysis.

5.6 Conclusions and Outlook

This work presented the multi-zone picker routing problem (MZPRP) and the balanced multi-zone picker routing problem (BMZPRP). Both optimization problems address picker routing in zoned warehouses with scattered storage and consist of three levels. The top level involves specifying in which zone(s) each demanded article is picked. The second level addresses the scattered storage issue and determines from which pick positions the SKUs are picked within each zone. Finally, the lowest level optimizes a picker tour in each zone. Since the decisions across all three levels are interrelated, they must be made simultaneously to achieve an optimal solution. While the MZPRP seeks minimum-length picker tours, the BMZPRP balances the picker tour lengths by minimizing the length of the largest picker tour.

To the best of our knowledge, no exact solution approach for both problems has been published so far, despite the significant relevance in the industry, where major global companies combine zoning and scattered storage with picker routing.

We introduced a new solution algorithm to solve both problems exactly. The MIP-based approach uses the extended state space of Heßler and Irnich (2024) for picker routing in warehouses with scattered storage, which is based on the seminal dynamic program of Ratliff and Rosenthal (1983). Finally, a network-flow formulation is obtained and solved using a MIP solver.

An extensive computational study with a warehouse layout of up to 60 aisles and up to 200 required articles showed that the MIP-based approach is highly efficient for solving both problems. For the MZPRP, all instances are solved to optimality within milliseconds or a few seconds, depending on whether multiple pick positions have to be visited to satisfy the demand for an article (general-demand case) or the supply of an article at a pick position is guaranteed to be sufficient to fulfill the demand (unit-demand case). Moreover, our study revealed that zone picking can save up to 12.6% of the costs compared to single-zone picking. For the BMZPRP, the computation times are longer for the large instance settings with 200 articles. A few instances cannot be solved optimally within the time limit of 600 seconds. Even though the total length is not directly optimized in this problem, it is, on average, only 4% higher than for the MZPRP.

Our introduced solution approach is universally applicable since the extended state space used to build the network can be easily substituted by other state space concepts. Therefore, our mathematical formulation can serve as a basis for problem variants of the (B)MZPRP without major modifications. Future research could benefit from this feature and adapt the solution approach to possible problem variants, such as multi-block warehouses, depot modifications (Heßler and Irnich, 2024, Table 1), or heuristic routing strategies (Lüke *et al.*, 2024). Besides, the MZPRP or BMZPRP could also be integrated into the more complex *joint order batching and picker routing problem* (Wahlen and Gschwind, 2023) as a subproblem.

Acknowledgement

Parts of this research were supported by Deutsche Forschungsgemeinschaft (DFG) under grant IR 122/13 of project 555303283. This support is gratefully acknowledged.

Bibliography

Bartholdi, III, J. J. and Hackman, S. T. (2019). *Warehouse & Distribution Science*. Atlanta, GA 30332-0205 USA, release 0.98.1 edition. Online only, `https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf`.

Bock, S., Bomsdorf, S., Boysen, N., and Schneider, M. (2025). A survey on the traveling salesman problem and its variants in a warehousing context. *European Journal of Operational Research*, **322**(1), 1–14.

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.

Chen, F., Wang, H., Qi, C., and Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, **66**(1), 77–85.

Choe, K.-I. (1991). *Aisle-based order pick systems with batching, zoning, and sorting*. Dissertation, Georgia Institute of Technology.

Daniels, R. L., Rummel, J. L., and Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, **105**(1), 1–17.

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.

de Koster, R. B. M., Le-Duc, T., and Zaerpour, N. (2012). Determining the number of zones in a pick-and-sort order picking system. *International Journal of Production Research*, **50**(3), 757–771.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 419–835.

Gray, A. E., Karmarkar, U. S., and Seidmann, A. (1992). Design and operation of an order-consolidation warehouse: Models and application. *European Journal of Operational Research*, **58**(1), 14–36.

Grosse, E. H., Glock, C. H., Jaber, M. Y., and Neumann, W. P. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, **53**(3), 695–717.

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.

Gutin, G. and Punnen, A., editors (2002). *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, Dordrecht.

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

Haouassi, M., Kergosien, Y., Mendoza, J. E., and Rousseau, L.-M. (2025). The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, **63**(4), 1304–1325.

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*, **36**(6), 1417–1435.

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*, MOS-SIAM Series on Optimization, chapter 1, pages 1–33. SIAM, Philadelphia, PA.

Jane, C. C. (2000). Storage location assignment in a distribution center. *International Journal of Physical Distribution & Logistics Management*, **30**(1), 55–71.

Jane, C.-C. and Laih, Y.-W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. *European Journal of Operational Research*, **166**(2), 489–496.

Löffler, M., Boysen, N., and Schneider, M. (2022). Picker routing in AGV-assisted order picking systems. *INFORMS Journal on Computing*, **34**(1), 440–462.

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.

Masae, M., Glock, C. H., and Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, **224**, 107564.

Mellema, P. M. and Smith, C. A. (1988). Simulation analysis of narrow-aisle order selection systems. In *Proceedings of the 20th conference on Winter simulation*, pages 597–602.

Parikh, P. J. and Meller, R. D. (2008). Selecting between batch and zone order picking strategies in a distribution center. *Transportation Research Part E: Logistics and Transportation Review*, **44**(5), 696–719.

Parikh, P. J. and Meller, R. D. (2009). Estimating picker blocking in wide-aisle order picking systems. *IIE Transactions*, **41**(3), 232–246.

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

Petersen, C. G. (2002). Considerations in order picking zone configuration. *International Journal of Operations & Production Management*, **22**(7), 793–805.

Pinedo, M. L. (2022). *Scheduling*. Springer, Cham, Switzerland, 6. edition.

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. and Vis, I. F. (2006). A model for warehouse layout. *IIE transactions*, **38**(10), 799–811.

Russell, M. L. and Meller, R. D. (2003). Cost and throughput modeling of manual and automated order fulfillment systems. *IIE Transactions*, **35**(7), 589–603.

Saylam, S., Çelik, M., and Süral, H. (2023). The min–max order picking problem in synchronised dynamic zone-picking systems. *International Journal of Production Research*, **61**(7), 2086–2104.

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.

Singh, K. N. and van Oudheusden, D. L. (1997). A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, **97**(3), 571–579.

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., and Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, **305**(1), 222–239.

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

Van Gils, T., Ramaekers, K., Caris, A., and de Koster, R. B. M. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, **267**(1), 1–15.

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

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, **95**, 139–150.

Weidinger, F., Boysen, N., and Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, **274**(2), 501–515.

Wildt, C., Weidinger, F., and Boysen, N. (2025). Picker routing in scattered storage warehouses: an evaluation of solution methods based on tsp transformations. *OR Spectrum*, **47**, 35–66.

Yu, M. and De Koster, R. B. (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, **198**(2), 480–490.

Chapter 6

Solving the Skiving Stock Problem by a Combination of Stabilized Column Generation and the Reflect Arc-Flow Model

Laura Korbacher, Stefan Irnich, John Martinovic, Nico Strasdat

Abstract

The skiving stock problem represents a natural counterpart of the extensively studied cutting stock problem and requires the construction of as many large units as possible from a given set of small items. In recent years, it has been able to develop into an independent branch of research within the OR community, exhibiting a wide variety of specific applications and associated integer programs, reaching its preliminary peak with the introduction of a powerful graph-theoretic approach, the reflect arc-flow model. However, there are still many benchmark instances that even the current formulations cannot yet contribute to solving. To this end, we present a new approach that is based on the observation that solutions of very good quality can already be determined on rather sparse (arc-flow) graphs, in general. More precisely, the arc sets of these graphs are defined by appropriate patterns obtained from a stabilized column-generation approach, so that considering the complete integer reflect arc-flow model is only necessary in a few cases. Compared to the reflect+ approach originally introduced for the cutting stock problem, we apply several modifications, discuss their numerical advantages by extensive computational tests, and end up with an adaptive solution method showing the most convincing results. In particular, we succeed in optimally solving some very challenging benchmark instances for the first time.

6.1 Introduction

In this article, we study the one-dimensional *skiving stock problem* (SSP), one of the classic representatives in cutting and packing. We are given a set of *item types*, characterized by their *item sizes* and *frequency* of appearance (also called *availability*). The SSP requires the combination of these items to a maximum number of larger units each of which exhibiting at least a predefined *threshold* length. Even though the concrete notation will be introduced explicitly in the following section, we would like to refer to Figure 6.1 for the sake of illustration. Already this introductory description clearly reveals some parallels to the extensively studied one-dimensional *cutting stock problem* (CSP) – and, indeed, the scientific framing of the SSP cannot be done without having established a thorough overview of the CSP history first. In the cutting stock problem, a given demand of small items has to be satisfied by using as little stock material as possible. Before moving forward into a more detailed literature review, we point out that, although forming an obviously related pair of minimization and maximization problems and sharing the same kind of input data, the CSP and the SSP are *not* dual to each other from the perspective of mathematical optimization. Consequently, despite their obvious common features, the two problems are to be regarded as independent.

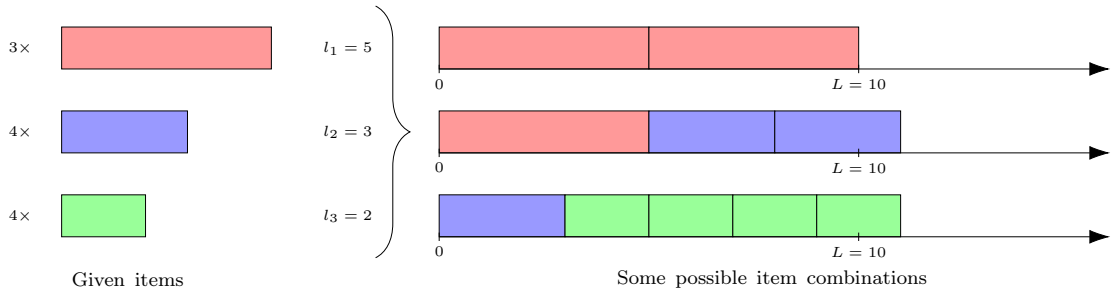


Figure 6.1: A simple instance of the skiving stock problem with three item types and threshold length $L = 10$.

6.1.1 Literature Review

The scientific history of the CSP dates back to 1939, when Kantorovich first described this problem in a Russian publication, as one of many scheduling problems arising in production processes. Although the CSP did not appear yet under its current name in this early work, this article laid important foundations for an *integer linear programming* (ILP) formulation based on assignment variables, which from today's point of view can be referred to as the textbook model, but at the same time also has two decisive drawbacks: (i) a highly symmetric solution space,

and (ii) a poor *linear programming* (LP) relaxation. Moreover, the idea of minimizing waste in one-dimensional cutting was already addressed by a few examples. As a result of the turmoil of war and post-war time, Kantorovich's contributions did not achieve international visibility within the Operations Research community until his research results were translated into English in the early 1960s, see (Kantorovich, 1960). From this moment on, an increased interest in and an intensified discussion of the topics of cutting and packing optimization could be perceived in the academic world. It is important to note that (Kantorovich, 1960, Example 5), in fact, already approaches the solution of a CSP based on *cutting patterns* (i.e., feasible combinations of items that can be cut from one piece of stock material), a concept which was taken up much more systematically by Gilmore and Gomory a little later in (Gilmore and Gomory, 1961, 1963). By the latter contributions, the still today often favored pattern-based (or set-covering) view of such optimization problems came into focus, and a novel description of the CSP as an integer linear program could be established. However, the only linear number of constraints was opposed by an exponential number of variables, and therefore also by a formulation that was difficult to handle at that time. While the integer problem could still not be addressed satisfactorily, the general approach chosen by Gilmore and Gomory had the decisive advantage that the continuous relaxation of the model led to very good approximations[1] for the actual optimal value. Moreover, the solution of the LP relaxation could be obtained effectively with the help of column generation (Desaulniers *et al.*, 2005), which also allowed to generate feasible integer solutions (e.g., by rounding procedures) of reasonably good quality. In particular, it is also due to these convincing properties of the relaxation that it still serves as an important cornerstone of some of the most powerful solution methods, such as branch-and-price algorithms, for the CSP or the closely related *bin packing problem* (BPP), see (Belov and Scheithauer, 2002), (Valério de Carvalho, 1999) or (Martello *et al.*, 1999) to mention only some (by far not exhaustive) references.

Because of the exponential number of variables in the pattern-based model already alluded to, a parallel branch of research has emerged in the literature dealing with *pseudo-polynomial formulations*. The theoretical foundations of such approaches mainly include the relations between integer and dynamic programming (viewed from a perspective of *network flows*) and were already introduced a little

[1] Although this will not be covered intensively in our article, we would like to note that this observation has developed into one of the most important (open) theoretical research questions in the field of cutting and packing to date, the so-called *MIRUP conjecture* (for the CSP) or *MIRDP conjecture* (for the SSP). For the purpose of further reading, we recommend the articles by Caprara *et al.* (2015) and Kartak *et al.* (2015) (for the CSP case) as well as Martinovic and Scheithauer (2016b, 2018) (for the SSP case) to the inclined reader, but not without mentioning that the references used therein represent the historical developments within this field much better than is possible by only citing a few contributions as examples.

later by Shapiro (1968) and Wolsey (1977). Another modeling idea, based largely on publications by Rao (1976) and Dyckhoff (1981), on the other hand, focuses on the individual cuts necessary to define a pattern (which is, more or less, a collection of cuts) and is therefore also referred to as the *one-cut model*. Even though these approaches, as we know today, combined the advantages of a relatively manageable size with a good LP bound at the same time, they were rather of theoretical use at the moment of their conception, since even moderate instances were not solvable with the just developing computer technology. For the sake of exposition, we would like to quote here a passage from (Stadtler, 1988) that summarizes well the attitude towards the alternative models at that time:

> "This leads to the conclusion, that all cutting stock problems that can be solved by one-cut models could also be solved by the column generation approach for the complete-cut model (but not vice versa). [...] The set of real world cutting stock problems solvable by the one-cut model is only a subset of those which could be tackled by the column generation approach."

Even though the importance of numerical aspects was manifested in discrete optimization at least since the well-known book (Nemhauser and Wolsey, 1988), it was not until around the turn of the millennium that these pseudo-polynomial models could be profitably used to solve the CSP. One of the first contributions is due to Valério de Carvalho (1999), who succeeded to solve exactly larger instance sizes, including some open instances of the BPP, by implementing a column (and row) generation algorithm (for a flow-based approach) that reduced the computational efforts considerably. Thanks to these first numerical successes, a very detailed theoretical overview article (Valério de Carvalho, 2002), and the steadily progressing development of suitable hardware components and (commercial) optimization software, the scientific discussion of the alternative models was significantly revived. The advantages already hinted at in the early contributions mentioned above were finally given the opportunity to prove themselves in meaningful practical experiments. From that moment on, any of the approaches, i.e., the pattern-based model and the pseudo-polynomial formulations, has seen numerous improvements over the years, all of which fostering contributions to further increase the size of problems that can be solved in a reasonable amount of time. We would like to highlight the following research activities in particular as significant milestones:

- stabilization techniques for column generation, limiting the fluctuation of dual prices so that a faster convergence is obtained (Valério de Carvalho, 2005; Ben Amor *et al.*, 2006; Gschwind and Irnich, 2016),
- reduction methods for the one-cut model (Martinovic *et al.*, 2018) and for the flow-based formulations (Brandão and Pedroso, 2016), the latter of which

culminating in the very powerful *reflect arc-flow* approach proposed by Delorme and Iori (2020).

While both these major techniques will be explained in detail in the corresponding main parts of the article, here we just mention the survey articles (Delorme *et al.*, 2016) and (de Lima *et al.*, 2022) for a good overview of further contributions in terms of the CSP and corresponding modeling frameworks.

The SSP has found its way into the scientific discussion much later than the CSP, but it has nevertheless followed a strikingly similar trajectory. More precisely, the SSP was first described in the Ph.D. thesis of Assmann (1983) and the follow-up article of Assmann *et al.* (1984), namely, conceptually slightly misleading, as a *dual bin packing problem* (DBPP). As already indicated at the beginning, this does not at all mean a duality in the sense of mathematical optimization, but merely an interaction of the two problems in terms of content. While the DBPP primarily addressed the processing of a very heterogeneous list of items (i.e., all of which have a frequency of one), the term SSP was first used to describe a concrete practical application in the paper industry (Johnson *et al.*, 1997; Zak, 2003), but henceforth has stood for the actual counterpart to the CSP, i.e., a problem with a very homogeneous list of items whose elements can be grouped into item types. Although being similar at first view, different solution methods have been developed for any of the two variants to effectively tackle the few (but existing) problem-specific challenges. We refer the reader to the research of Labbé *et al.* (1995) and Peeters and Degraeve (2006), introducing early branch-and-bound based approaches for the DBPP, as well as the article by Zak (2003) focussing on a pattern-based approach, the so-called *standard model*, for the SSP case. Despite its still young age, a certain range of application fields for the SSP has opened up in the last two decades, especially, where responsible and sustainable handling of raw materials or offcuts is desirable. In addition to the examples from the paper industry, here we further mention manufacturing and inventory planning (Arbib *et al.*, 2002), multiprocessor scheduling problems (Alvim *et al.*, 2004), further practical case studies (Agoston, 2019), and wireless communications (Martinovic *et al.*, 2017; Tragos *et al.*, 2013). Especially in the last mentioned area, in which the optimal reconstruction of unused portions of the frequency spectrum is concerned, approaches based on the SSP have contributed to a substantial improvement of the previous state of research. Moreover, the advancing scientific discussion of the SSP proved useful even when it appeared only as a subproblem within combined cutting-and-packing applications (Chen *et al.*, 2019; Kłosowski *et al.*, 2018; Wang *et al.*, 2020).

In recent years, this development of an increasingly broadening range of relevant application fields has been favored primarily by the fact that significant progress has also been made with respect to the theory of alternative and more sophisti-

cated modeling techniques. Most notably, the first holistic research in terms of pseudo-polynomial formulations for the SSP has been conducted in Martinovic and Scheithauer (2016a): The authors not only proposed an *arc-flow model* and a *one-stick model* for the problem under investigation, but also constructively proved the equivalence of their LP relaxations among one another and to that of the pattern-based standard model. This not only provided additional modeling options, but also highlighted their general and computational strengths at the same time, bringing together observations that had remained separate for several decades in the CSP case. However, motivated by the results in Delorme and Iori (2020), some serious improvements of the original arc-flow model have been proposed in Martinovic *et al.* (2020). To be more precise, the authors of that article made use of reversed loss arcs and the idea of the reflect arc-flow model, that (roughly speaking) only requires to deal with the first half of the vertex set to obtain ILP formulations of less challenging size. Moreover, the aforementioned piece of research contained the first systematic approach to collect a plethora of differently characterized benchmark instances for the SSP, which then served as a solid basis for extensive computational experiments clearly showing the convincing performance of the new formulations. De Lima *et al.* (2023) present a more generic approach for arc-flow models of pseudo-polynomial size which combines column generation, variable-fixing based on reduced costs, and an asymmetric branching scheme. Experiments on the SSP show clear improvements in terms of average computation times for smaller benchmark instances in comparison to the just mentioned arc-flow formulation and the reflect arc-flow formulation. However, they did not present results of their new approach for the very large, challenging SSP instances that we will consider.

6.1.2 Overview and Contribution

Despite all the progress on solving the SSP, there is still a large number of unresolved instances within the various classes of test sets. This justifies further research on several possible algorithmic refinements. In this sense, we adapt and refine the reflect+ algorithm of Delorme and Iori (2020) that was originally designed to solve CSP and BPP instances. Reflect+ is an exact algorithm trying to reduce the computational effort in the average case. To this end, a sequence of reflect arc-flow models with only a subset of the arcs/variables are solved by a *mixed integer programming* (MIP) solver, each providing a heuristic primal solution. The linear relaxation of a pattern-based formulation is solved beforehand to compute a dual bound and to identify promising patterns and (implicitly) also promising arcs for the integer reflect arc-flow formulation. When primal and dual bound coincide, optimality is proven. Otherwise, the next larger reflect arc-flow model of the sequence is considered. The last model is the complete reflect arc-flow model,

in which however provably non-optimal arcs are eliminated using reduced-cost arguments.

Algorithm 2 summarizes the adapted and refined reflect+ algorithm that we suggest for solving the SSP. Steps 2–4 are related to the solution of the linear relaxation of the pattern-based formulation. Compared to the reflect+ algorithm of Delorme and Iori, we differ in the following aspects:

- For pricing, we use the smaller arc-flow network. This improves the run time (the impact is however minor) and makes sure that all generated patterns can be later transformed into arcs of the reflect arc-flow model. In contrast, Delorme and Iori (2020) use COMBO from (Martello *et al.*, 1999) that solves a binary knapsack problem first heuristically and exactly only if the heuristic fails.
- We stabilize the column-generation process with the help of dual inequalities, which is particularly beneficial for very large-scale SSP instances, for which otherwise the column-generation process is too slow so that reflect+ already fails at this stage. Solutions to stabilized pattern-based models consist of a mix of pattern columns and columns representing the dual inequalities. A retransformation into a pure pattern-based solution is then necessary (see Step 4).

The result is two sets P_B and P_{LP} of those patterns present in the final basic solution of the linear relaxation and all generated patterns, respectively.

Algorithm 2 Reflect+ for the Skiving Stock Problem

1: Greedy heuristic $H^{(1)}$ ▷ LB_0
2: Build initial patterns P_{init} for the column-generation (CG) procedure with subset-sum procedure $H^{(2)}$
3: Use stabilized CG to solve linear relaxation of pattern-based model ▷ z_{LP} and $UB = \lfloor z_{LP} \rfloor$
4: Transformation of the CG solution into a pure pattern-based solution ▷ P_B and P_{LP}
5: **Level 1:** Solve reflect arc-flow model with arc set $\mathcal{A}_1$ constructed with P_B ▷ LB_1
6: **Level 2:** Solve reflect arc-flow model with arc set $\mathcal{A}_2$ constructed with P_{LP} ▷ LB_2
7: **Level 3:** Solve reflect arc-flow model with arc set $\mathcal{A}_3$ where unpromising arcs fixed to zero ▷ LB_3
8: $LB \leftarrow max(LB_0, LB_1, LB_2, LB_3)$
9: Compute reduced costs $\tilde{c}_{ij}$ of arcs (i, j) using path-reduced costs from the CG solution
10: **while** $(LB < UB)$ **do**
11: **Level 4:** Fix arc variables x_{ij} to zero if $\tilde{c}_{ij} > z_{LP} - UB$ to obtain arc set $\mathcal{A}_4$ ▷ $\mathcal{A}_4$
12: Solve reflect arc-flow model with arc set $\mathcal{A}_4$ ▷ LB_4
13: $LB \leftarrow max(LB, LB_4)$ and $UB \leftarrow UB - 1$
14: **end while**

As a preparatory step (Step 9), reduced costs of all variables of the corresponding arc-flow formulation are computed. In contrast to the suggestion of Delorme and Iori (2020), we obtain them directly from the linear relaxation of the pattern-based formulation exploiting again that pricing uses the arc-flow network. Transferring these reduced costs to the reflect arc-flow formulation is possible.

The Steps 5–13 are describing how the sequence of reflect arc-flow models is used. At Levels 1 to 3 (Steps 5, 6, and 7), the solution approach is heuristic because

the restricted models are built with heuristic rules to thin out the arc/variable set. In contrast, the loop (Steps 9–13) iteratively solves exact models based on the assumption that the computed lower bound LB is the optimal objective. As long as the new computed solution (if any) is worse than UB, then UB is decreased by one and the process repeats. Note that if the MIRDP conjecture holds, then not more than two traversals of the loop of Level 4 result.

We describe the details of all steps and the required notation of the reflect+ algorithm in the following sections with a focus on the algorithmic refinements (also underlined in the pseudocode of Algorithm 2) compared to the algorithm of Delorme and Iori (2020) for the CSP. Our reflect+ algorithm for the SSP relies on several independent algorithmic components that were identified as helpful during the development phase of the approach. A major complication is that some algorithmic components are only accelerating the reflect+ algorithm for some types of instances, and that – even worse – the overall performance depends also on the interplay between these components. As a result, the final reflect+ algorithm that we suggest is a compromise that balances the use of the new algorithmic components who compete for a limited time budget. It should also be noted that even the best-performing exact approaches for the SSP to date have all been evaluated on proper subsets of the available benchmark sets (de Lima *et al.*, 2022).

6.1.3 Structure of the Paper

The remainder of the paper is structured as follows. Section 6.2 introduces notation and presents the greedy heuristic and subset-sum procedure. Pattern-based formulations and the solution of their linear relaxation with the help of a stabilized column-generation algorithm are discussed in Section 6.3. Section 6.4 then elaborates the construction of restricted reflect arc-flow models and their solution with a MIP solver. Computational results are presented in Section 6.5. The paper closes with final conclusions drawn in Section 6.6.

6.2 Preliminaries and Heuristic Components

Let us start by formally introducing the SSP. Let $\mathbb{N} = \{1, 2, 3, \dots\}$ denote the natural numbers and $\mathbb{Z}_+ = \{0\} \cup \mathbb{N}$ the non-negative integers.

Definition 1. *An* instance *E of the SSP is given by a tuple $(m, \mathbf{l}, L, \mathbf{b}) \in \mathbb{N} \times \mathbb{N}^m \times \mathbb{N} \times \mathbb{N}^m$ with the interpretation that m is the number of different items types, $\mathbf{l} = (l_1, \dots, l_m)$ the vector of item sizes (lengths), L the threshold length, and $\mathbf{b} = (b_1, \dots, b_m)$ the item frequencies. For convenience, we refer to $I = \{1, \dots, m\}$ as the* item (type) set.

Without loss of generality, we assume the item types to be ordered decreasingly with respect to size, i.e., $l_1 > l_2 > \ldots > l_m$ has to be satisfied. Moreover, $l_1 < L$ can be assumed, because any item equal to or larger than L does not have to be considered within the optimization.

Definition 2. *Let $E = (m, \mathbf{l}, L, \mathbf{b})$ denote an SSP instance.*

(i) Any vector $\mathbf{p} \in \mathbb{Z}_+^m$ satisfying $\mathbf{l}^\top \mathbf{p} \geq L$ is called a pattern *(a.k.a. packing pattern).*

(ii) A pattern $\mathbf{p} \in \mathbb{Z}_+^m$ is minimal *if there is no other pattern $\tilde{\mathbf{p}} \in \mathbb{Z}_+^m$ with $\tilde{\mathbf{p}} \leq \mathbf{p}$ (in a componentwise sense).*

(iii) A pattern $\mathbf{p}$ is proper *if $\mathbf{p} \leq \mathbf{b}$ (in a componentwise sense).*

(iv) The set of all patterns *is denoted by $P(E)$, the set of all minimal patterns by $P^\star(E)$, and the set of all proper patterns by $P_p(E)$.*

Before dealing with exact methods for solving the SSP in the following sections, two heuristic components will be discussed here first: the greedy heuristic $H^{(1)}$ and the subset-sum procedure $H^{(2)}$. Both methods have already been briefly mentioned in Algorithm 2, but without explaining their concrete mechanisms. $H^{(1)}$ provides a lower bound for the optimal value and a feasible solution for a given SSP instance (together with an associated set of patterns used), whereas from $H^{(2)}$ we obtain only an expedient set of patterns. While the former represents purely numerical information that can be provided by several SSP heuristics with similar or even identical quality, the selection of, in a sense, "promising" patterns is crucial for the further progress of the reflect+ algorithm. We therefore distribute the two aforementioned purposes of such an approximate solution over two different methods, using:

- the very simple greedy heuristic $H^{(1)}$ to generate a first lower bound,
- the somewhat more sophisticated subset-sum procedure $H^{(2)}$ to provide a reasonable set of, in a sense "low-waste", initializing patterns for the column-generation process.

The greedy heuristic $H^{(1)}$ follows the procedure already presented in (Peeters and Degraeve, 2006) and (Martinovic *et al.*, 2020). By that, we mean that a bin is first filled with as many copies as possible of the currently largest available item type $i^\star \in I$ (so that the threshold is not yet exceeded). Among all items then still available, one is searched for that completes the previous configuration to a feasible pattern while wasting as little material as possible. If such an item does not exist, the bin is first further filled with as many items of the type $i^\star + 1$ (again, without exceeding the capacity), and then it is checked again whether a completing item (in the previously defined sense) exists.

In contrast, the subset-sum procedure $H^{(2)}$ builds up to m patterns based on the subset-sum problem. More precisely, we iteratively fix an item type $i \in I$ and apply the following steps:

S1: Define a new threshold length $L_i := L - l_i$ and a new set of items: $I_i := I \setminus \{i\}$.

S2: Solve a subset-sum problem to find a subset of items (of I_i), each of which appearing at most once, whose item lengths add up to the new threshold length L_i. If there is no such combination, find a subset of items with the shortest pattern length larger than L_i.

S3: Add item i to this subset and store the set of items as a new pattern.

We implemented the algorithm of Cormen *et al.* (2009) to tackle the subset-sum problems. Typically, the main advantage of the subset-sum procedure is that most of the resulting patterns have a pattern length equal to or only slightly larger than the threshold length, whereas patterns obtained from the greedy heuristic are often strictly longer than L. In other words, one could say that the patterns constructed using procedure $H^{(2)}$ are generally more promising in the sense that they exhibit better material utilization. As we will see later, this idea is particularly helpful for those benchmark instances whose integer solution consists of low-waste or even waste-free patterns (like AI/ANI), see Section 6.5.

Remark 1. *Note that the columns produced by $H^{(2)}$ do not represent a feasible solution of the SSP, in general. Hence, we do not refer to this method by the term "heuristic".*

6.3 Pattern-based Formulations and Solution with Stabilized Column Generation

With the definition of patterns, it is possible to formulate pattern-based models of the SSP as proposed by Zak (2003). Such a model can be formulated for any set of patterns P, e.g., the complete set $P = P(E)$ or a subset of it. Let $\lambda_{\mathbf{p}} \in \mathbb{Z}_+$ for each $\mathbf{p} = (p_i)_{i \in I} \in P$ count how often the respective pattern is built, then we obtain the pattern-based formulation for the pattern set P:

$$(F_P) \qquad z(F_P) = \max \sum_{\mathbf{p} \in P} \lambda_{\mathbf{p}} \tag{6.1a}$$

$$\text{subject to} \quad \sum_{\mathbf{p} \in P} p_i \lambda_{\mathbf{p}} \leq b_i, \qquad i \in I, \tag{6.1b}$$

$$\lambda_{\mathbf{p}} \in \mathbb{Z}_+, \qquad \mathbf{p} \in P. \tag{6.1c}$$

The objective function (6.1a) maximizes the number of constructed patterns, while constraints (6.1b) make sure that at most b_i items of type $i \in I$ are used. The domain of the decision variables is stated in (6.1c).

Any pattern set $P \supseteq P^\star(E) \cap P_p(E)$ guarantees that the corresponding solution is optimal for the instance E. Indeed, by definition, any feasible solution to the integer problem only uses proper patterns; and it is always possible to shorten non-minimal patterns by releasing superfluous material which can possibly be used for further constructions. As a consequence of that, any optimal solution to the LP problem has to consist of minimal patterns only. Due to these observations, the integer models for $P(E)$, $P^\star(E)$, and $P_p(E)$ are equivalent, i.e.,

$$z(F_{P(E)}) = z(F_{P^\star(E)}) = z(F_{P_p(E)}).$$

The linear relaxation of the pattern based model (6.1) is denoted by $LP(F_P)$, its objective value by $z_{LP}(F_P)$. For the linear relaxations, we have

$$z_{LP}(F_{P^\star(E)}) = z_{LP}(F_{P(E)}) \qquad \text{and} \qquad z_{LP}(F_{P_p(E)}) \leq z_{LP}(F_{P(E)}),$$

where, in the latter case, $<$ is possible for some SSP instances E. Consequently, using proper patterns may lead to a better linear-relaxation bound, as observed in Martinovic and Scheithauer (2016c) or, in parts, also in (Delorme and Iori, 2020, Subsection 2.2) and (Kartak *et al.*, 2015, Section 2) for the closely-related CSP. In contrast, the first relation has to be an equation as any non-minimal pattern can be reduced to an element of $P^\star(E)$ even in the LP relaxation.

6.3.1 Stabilized Column Generation

Given the generally exponentially large number of patterns and variables in the aforementioned models, they cannot be solved directly in their integer version. However, the continuous relaxation of such a pattern-based formulation can be solved efficiently in practice using the column-generation method. Here, one starts with a small set of patterns and iteratively checks whether more patterns need to be added to this set by solving a pricing problem. In our case, we use the arc-flow network to handle the pricing subproblems and add up to 500 new patterns with negative reduced cost to the pattern pool per iteration[2].

The column-generation process for solving the linear relaxation of (6.1) produces new patterns complementing those of the initial pattern set P_{init}, i.e., Step 3 of Algorithm 2. For the quality of the linear-relaxation bound, we are interested to only generate patterns from the set $P^*(E) \cap P_p(E)$. However, other patterns

[2]Recall that the reduced or opportunity cost of a pattern $\mathbf{p}$ is $-1+\boldsymbol{\pi}^\top \mathbf{p}$ for a dual-prices vector $\boldsymbol{\pi}$ in the linear maximization problem. For the selection of not more than 500 new patterns, patterns are sorted by reduced cost (breaking ties with the smallest index of all included items) and those with smaller reduced cost are selected first. Note that the upper bound of 500 new patterns per iteration is a heuristic choice. Observe that not limiting the number of patterns per step might lead to some memory issues.

$\mathbf{p} \in P(E) \setminus (P^*(E) \cap P_p(E))$ may be generated also because of the following two algorithmic components.

First, pricing problems for CSP and SSP are solved as a type of shortest-path problem over a digraph (V, A) in which vertices $v \in V$ correspond to the possible lengths of partial patterns, i.e., values $0, 1, 2, \ldots, L$ for the CSP and likewise $0, 1, 2, \ldots, L, L+1, \ldots, L+l_1-1$ for the SSP, and arcs $(j, j+l_i) \in A$ represent the item types $i \in I$ (Gilmore and Gomory, 1963; Zak, 2003). This can lead to patterns containing a number of items larger than their demands in the CSP and larger than their frequencies in the SSP. In contrast, by solving a binary knapsack problem only binary patterns which are proper and maximal are generated for the BPP. Our pricing algorithm uses the SSP arc-flow network for pricing out patterns and solves a shortest-path problem over it (a detailed definition of the network can be found in Martinovic *et al.*, 2020, Section 2). The use of the arc-flow network reduces the occurrence of patterns that are not proper. Furthermore, it guarantees that patterns are minimal. Neither the arc-flow nor the reflect arc-flow model can however guarantee that only proper patterns occur (see again Martinovic *et al.*, 2020, Remark 4 and Example used in Theorem 8). As a result, we generally have the relationship

$$z_{LP}(F_{P^\star(E)\cap P_p(E)}) \leq z_{LP}(F_{P_{CG}(E)}) \leq z_{LP}(F_{P(E)})$$

where $P_{CG}(E)$ refers to the pattern set computed with the column-generation procedure. Note that strict inequalities are possible.

Second, we use a stabilization approach with dual inequalities for the column-generation algorithms. Let $\boldsymbol{\pi} = (\pi_i)_{i\in I}$ be the dual variables to the constraints (6.1b). Any inequality $\mathbf{a}^\top \boldsymbol{\pi} \geq c$ with $(\mathbf{a}, c) \in \mathbb{R}^{m+1}$ is a dual inequality corresponding with a column $\mathbf{a}$ with cost $c \in \mathbb{R}$ in the primal formulation (6.1). Ben Amor *et al.* (2006) classify dual inequalities into

- *dual-optimal inequalities* (DOIs) are dual inequalities that do not cut off any dual optimal solution $\boldsymbol{\pi}^\star \in \Pi^\star$ (we denote the set of dual optimal solutions by $\Pi^\star$ in the following)
- *deep dual-optimal inequalities* (DDOIs). The latter term refers to a set $\{\mathbf{a}_k^\top \boldsymbol{\pi} \geq c_k\}$ of dual inequalities and means that the inequalities together do not cut off the entire dual optimal space, i.e., $\Pi^\star \cap \{\boldsymbol{\pi} \in \mathbb{R}_+^m : \mathbf{a}_k^\top \pi \geq c_k \text{ for all } k\} \neq \varnothing$.

Ben Amor *et al.* (2006) and Gschwind and Irnich (2016) prove several equivalent properties of DDOIs, among others, that the primal formulation extended by columns corresponding to a set of dual inequalities and the primal formulation itself provide the same linear-relaxation bound if and only if the dual inequalities are DDOIs. Moreover, this is equivalent to the existence of a constructive procedure that transforms any optimal mixed solution (patterns and columns resulting

from dual inequalities) into a pure pattern solution of the same cost. The point is now that this transformation may produce patterns that are not proper. The type of produced patterns depends on both the initial pattern set P_{init}, the pattern-generation process in column generation, and the type of dual inequalities added to the *restricted master problem (RMP)*. Therefore, we describe families of dual inequalities more precisely.

Following the taxonomy of Gschwind and Irnich (2016) and Heßler *et al.* (2018), the most general family of dual inequalities that they consider is defined as follows: For any item $j \in I$, subset $S \subset I \setminus \{j\}$, and weights $(w_i)_{i \in S} \in \mathbb{N}^{|S|}$ with $\sum_{i \in S} w_i l_i \geq l_j$, the inequality

$$\sum_{i \in S} w_i \pi_i - \pi_j \geq 0$$

is called *weighted subset inequality* (WSI) for $(j, S, (w_i)_{i \in S})$. Several WSIs defined by $(j_k, S_k, (w_i^k)_{i \in S_k})$ indexed by $k \in K$ lead to an extended primal model in which (6.1b) is replaced by

$$\sum_{\mathbf{p} \in P} p_i \lambda_{\mathbf{p}} - \sum_{k \in K : j_k = i} y_k + \sum_{k \in K : i \in S_k} w_k y_k \leq b_i, \qquad i \in I, \tag{6.1b'}$$

with additional non-negative continuous variables

$$y_k \geq 0, \qquad k \in K. \tag{6.2}$$

If all weights are equal to one, i.e., $w_i = 1$ for all $i \in S$, the dual inequality is a *subset inequality* (SI) for the pair (j, S). The interpretation of an SI defined by (j, S) for the primal model (6.1) is that one can replace an item of type j by the items of type $i \in S$ in every pattern without making the pattern infeasible. If $S = \{i\}$ is a singleton set (assuming $l_i > l_j$), the inequality becomes $\pi_i - \pi_j \geq 0$ and is denoted as a *pair inequality* (PI) for the item pair (j, i). The dual interpretation of the PI for (j, i) is that one unit of supply of item type i is at least as useful as one unit of supply of item type j, i.e., $\pi_i \geq \pi_j$. The set of all PIs (with $\mathcal{O}(m^2)$ elements) is equivalent to the set of *ranking inequalities* (RIs) $\pi_i - \pi_{i+1} \geq 0$ for $i \in I \setminus \{m\}$ (recall that we assume $l_1 > l_2 > \cdots > l_m$). The set of RIs comprises only $\mathcal{O}(m)$ elements.

Table 6.1 summarizes what is known about the relationship between different families of dual inequalities and the linear-relaxation of the extended linear pattern-based formulations.

		RIs ⊂ PIs	⊂ SIs	⊂ WSIs
Pattern-based formulation	$P = P_p(E)$‡	DDOIs	—	—
(6.1a), (6.1b'), $\lambda_{\mathbf{p}} \geq 0$, and (6.2)	$P = P_{CG}(E) \subset$	DDOIs	—	—
with pattern set P	$P = P(E)$	DOIs	DOIs	DOIs

Table 6.1: Families of dual inequalities (DIs) and their relationship to pattern-based formulations defined over different pattern sets; entries "—" indicate that the family of dual inequalities is not necessarily a set of DOIs or DDOIs; some inequalities may however be DOIs or DDOIs depending on the SSP instance E. (‡: Note that $P_{CG}(E) \setminus P_p(E) \neq \varnothing$ is possible.)

6.3.2 Transformation into a Pure-Pattern Solution

A solution obtained by stabilized column generation typically contains classical pattern vectors, but also "artificial" columns referring to some DDOIs. Of course, one could simply ignore the DDOI-variables and only rely on the pattern variables of the LP relaxation for the construction of thinned out graphs. However, we would lose valuable information by this restriction. Thus, it is strongly recommended to transform LP solutions with DDOIs into pure-pattern solutions, i.e., solutions consisting of patterns only.

The following steps are required to obtain a pure-pattern solution (a similar procedure for CSP was suggested by Valério de Carvalho (2005) and a more general procedure by Gschwind and Irnich (2016)):

1. Select a DDOI $(j_k, S_k, \mathbf{w}^k)$ that is part of the LP solution and find all patterns of the solution containing the item j_k. Store these patterns in a list.
2. Sort this list of patterns in increasing order by the value y_k of the DDOI-variables, see (6.1b').
3. Take the first pattern $\mathbf{p}$ of the list: Transform the item j_k into items $i \in S_k$ (according to the multiplicities $\mathbf{w}^k$). If $\lambda_{\mathbf{p}} \geq y_k$ in the LP solution, eliminate the DDOI completely, decrease the value $\lambda_{\mathbf{p}}$ by y_k, and construct a pattern $\mathbf{p}'$ in which item j_k is replaced by items $i \in S_k$. Insert $\mathbf{p}'$ into the list with value $\lambda_{\mathbf{p}'} := y_k$. Otherwise, convert the pattern $\mathbf{p}$ into $\mathbf{p}'$ as before, delete pattern $\mathbf{p}$ from the list, and reduce the solution value y_k by $\lambda_{\mathbf{p}}$.
4. If the solution value of the DDOI is still positive, go to Step 3 again. In the opposite case (that is, the DDOI is eliminated), take the next DDOI (if available) and go back to Step 1.

Typically, especially for larger instance sizes, there will be multiple options to choose a pattern to be transformed for the considered DDOI. In our internal pre-

calculations, we obtained slightly better results whenever as many patterns as possible have been transformed. Hence, in Step 2 of the above list, the patterns are sorted in increasing order by the value of the item that will be transformed. By that, the solution value of the artificial DDOI column is distributed over the maximum number of patterns in the final pure-pattern solution.

A well-known issue related to the patterns generated during the stabilized column-generation procedure is that they can obtain items in a larger quantity than allowed by the input data of the given instance (Ben Amor *et al.*, 2006). Such non-proper patterns are undesirable as discussed above.

Remark 2. *We observe more non-proper patterns at the end of the stabilized column-generation process than without stabilization. The above transformation can transform non-proper patterns into proper patterns. The negative effects of a missing transformation can later be observed in Table 6.4 for the large GI instances.*

6.4 Restricted Reflect Arc-Flow Models

Flow formulations form a powerful tool in cutting and packing, as they combine important structural properties (e.g., a good LP relaxation) with a large illustrativeness and a generally manageable model size, and so they can typically be handled well by ILP solvers. The reflect arc-flow model has recently emerged as the most important representative within this group (Delorme and Iori, 2020; Martinovic *et al.*, 2020). In contrast to classical graph-based approaches, this formulation is particularly convincing in that, roughly speaking, only half the bin size has to be modeled and, thus, numerous variables can be saved. This is significant because in conventional networks the number of active vertices and arcs is very large, especially in the second half of the graph, due to the ever-increasing combination possibilities of the items. In return for these savings, there is no longer a classical representation of patterns by paths from the source to a sink, but a pattern is modeled as a composition of two sub-paths that must be suitably combined. Without going into all conceivable details, it can be stated that a reflect graph consists of different classes of arcs. Using $R := L/2$ as an abbreviation, it can be obtained from a classical arc-flow network with arc set $\mathcal{A}$ according to the following rules:

- Any arc $(u, v) \in \mathcal{A}$ with $u < v \leq R$ is kept as a *standard arc* (u, v, s),
- Any arc $(u, v) \in \mathcal{A}$ with $u < R < v$ is replaced by a *reflected arc* $(u, L-v, r)$,
- Any arc $(u, v) \in \mathcal{A}$ with $R \leq u < v$ is omitted.

In this construction, R serves as the reflection point, and fractional coordinates can be avoided by scaling any item length by a factor of 2, if required. Moreover, we introduce *loss arcs* (d, e, ℓ) with $e = succ_{\mathcal{U}}(d)$, specifying vertex e as the direct

successor of vertex d. Loss arcs are added to the network, starting at the lowest-indexed vertex that is the head of a reflected arc and continuing step-by-step until R is reached. Finally, an artificial arc (R, R, r) has to be added to be able to combine two sub-paths representing exactly half of the bin size. Let us refer to the obtained vertex and arc sets by $\mathcal{U}$ and $\mathcal{A}$, respectively. Moreover, we define $\mathcal{U}^\star := \mathcal{U} \setminus \{0\}$ and $\mathcal{A}^\star := \mathcal{A} \setminus \{(R, R, r)\}$ for the sake of an easier notation in the following ILP model. Introducing $\xi_{de\kappa} \in \mathbb{Z}_+$ to indicate the units of flow carried by arc $(d, e, \kappa) \in \mathcal{A}$, where κ is a generic label for the arc type, we obtain the reflect arc-flow model for the SSP proposed in Martinovic *et al.* (2020):

$$(F_\mathcal{A}) \qquad z_\mathcal{A} = \max \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} \tag{6.3a}$$

$$\text{subject to} \sum_{\substack{(d,e,s)\in\mathcal{A}:\\ e-d=l_i}} \xi_{des} + \sum_{\substack{(d,e,r)\in\mathcal{A}:\\ e=L-d-l_i}} \xi_{der} \le b_i, \qquad i \in I, \tag{6.3b}$$

$$\sum_{(0,e,s)\in\mathcal{A}} \xi_{0es} + \sum_{(0,e,r)\in\mathcal{A}} \xi_{0er} = 2 \sum_{(d,e,r)\in\mathcal{A}} \xi_{der}, \tag{6.3c}$$

$$\sum_{(d,e,s)\in\mathcal{A}} \xi_{des} + \sum_{(e,f,\ell)\in\mathcal{A}} \xi_{ef\ell} = \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} + \sum_{(d,e,\ell)\in\mathcal{A}} \xi_{de\ell} + \sum_{\substack{\kappa\in\{r,s\},\\ (e,f,\kappa)\in\mathcal{A}}} \xi_{ef\kappa}, \qquad e \in \mathcal{U}^*, \tag{6.3d}$$

$$\sum_{(d,e,\ell)\in\mathcal{A}} \xi_{de\ell} + \sum_{(d,e,r)\in\mathcal{A}} \xi_{der} \ge \sum_{(e,f,\ell)\in\mathcal{A}} \xi_{ef\ell}, \qquad e \in \mathcal{U}^*, \tag{6.3e}$$

$$\xi_{de\kappa} \in \mathbb{Z}_+, \qquad (d, e, \kappa) \in \mathcal{A}^*, \tag{6.3f}$$

$$\xi_{R,R,r} \in \mathbb{Z}. \tag{6.3g}$$

Without going too much into the details, we would like to mention that the objective function minimizes the total flow traversed in the network. Furthermore, in the order of their occurrence, the constraints demand that the item supply is respected (see (6.3b)), any pattern is composed of two subpatterns that are linked by a reflected arc (see (6.3c)), the flow conservation is obeyed (see (6.3d)), and the use of loss arcs is consistent with the other decisions taken in the network (see (6.3e)). For reflect+, we extend the formulation of Martinovic *et al.* (2020) by adding additional variables that follow the same logic as the RIs in the pattern-based model. More precisely, let $t_i \in \mathbb{Z}_+$ for $i > 1$ mimic what an RI for items $i-1$ and i is doing: replace item i by longer item $i-1$ (knowing that $l_{i-1} > l_i$ holds). Therefore, constraints (6.3b) are replaced by the following constraints:

$$\sum_{\substack{(d,e,s)\in\mathcal{A}:\\ e-d=l_i}} \xi_{des} + \sum_{\substack{(d,e,r)\in\mathcal{A}:\\ e=L-d-l_i}} \xi_{der} - \begin{cases} t_i, & i > 1 \\ 0, & \text{otherwise} \end{cases} + \begin{cases} t_{i+1}, & i < m \\ 0, & \text{otherwise} \end{cases} \le b_i \tag{6.3b'}$$

Remark 3. *Similar conditions could also be added to describe the exchange of two or more items by one object with a larger total length. However, it was shown in Delorme and Iori (2020) that this generally does not lead to any improvement. The drawback is the relatively large number of additional constraints so that here we just focus on the RI-like exchanges.*

In our algorithm, we do not use the complete set of arcs in many places, but only subsets $\mathcal{A}' \subset \mathcal{A}$ of them in order to be able to work on thinned out graphs. It is therefore indispensable to describe how exactly a subset P of patterns is integrated into the network, since this largely determines the final combination possibilities. To this end, we run through the given subset of patterns and repeat the following procedure for any fixed pattern:

- We start with the largest item i appearing in the pattern $\mathbf{p}$ and add its corresponding arc(s) (starting at vertex 0; as often as the coefficient p_i indicates) to the network. We then place the arc of the second largest item next to it and process the remaining items equivalently until the reflection point is reached or exceeded.
- Continuing with decreasing item length, we include them in the graph (again starting at vertex 0) in the same way as before. Obviously, this time all items will be used before exceeding the reflection point.
- Combine the two paths by adding loss arcs or the artificial reflected arc, if required.

Before we deal with the numerical test calculations in the next section, we would now like to conclude by going into more detail about the individual stages of our algorithm:

Level 1: At first we solve $F_{\mathcal{A}_1}$, i.e., the reflect arc-flow model with just the pattern set P_B containing the patterns that are used in the basis solution found by column generation. By that, we obtain a feasible solution and a lower bound LB_1 for the unknown optimal value. In case the gap between LB_1 and the upper bound UB is less than one, we have already found an optimal solution and can stop the algorithm.

Level 2: We proceed in the same way as in Level 1, but this time solve $F_{\mathcal{A}_2}$ which originates from the typically much larger pattern pool P_{LP}, that is, all patterns that were generated during column generation.

Level 3: As an auxiliary tool, we first solve the linear relaxation of $F_{\mathcal{A}}$ and obtain the solution $\bar{\xi}$. We use the information provided by that specific solution to build the set $\mathcal{U}_n := \{u \in \mathcal{U} \colon \bar{\xi}_{de\kappa} < \epsilon, \forall (d, e, \kappa) \in \mathcal{A} \text{ incident to } u\}$ for $\epsilon = 10^{-6}$, which contains all vertices through which effectively no flow units pass. We then solve $F_{\mathcal{A}}$ with subsequent addi-

tional constraints

$$\xi_{de\kappa} = 0, \qquad d \in \mathcal{U}_n, (d, e, \kappa) \in \mathcal{A}, \tag{6.4}$$

which is equivalent to dealing with $F_{\mathcal{A}_3}$, a formulation that does not contain the unpromising arcs at all. On the one hand, the calculation effort gets a lot smaller with these constraints, but on the other hand there might be some arcs fixed to zero now that are part of an optimal integer so that the solution obtained in this level is just a lower bound with a gap of at least one.

Level 4: This level contains an iterative mechanism. At first, we assume that $UB := \lfloor z_{LP} \rfloor$ is the true optimal value. In case the reduced cost of an arc (from $\mathcal{A}$) is larger than the difference between z_{LP} and UB, this arc is not required for an optimal solution and its flow can be fixed to zero. By that, we obtain an arc set $\mathcal{A}_4$. The solution of $F_{\mathcal{A}_4}$ provides the lower bound LB_4. In case an optimal integer solution has not been found during an iteration, the *tentative upper bound* UB is decreased by one before the next iteration starts.
Whereas Delorme and Iori (2020) use reduced costs from the LP relaxation of the reflect arc-flow model, we found out that it is slightly advantageous to use path-reduced costs from the pattern-based formulation of the column-generation algorithm instead (see Section 6.5). Path-reduced costs can be computed via a bidirectional search technique as proposed by Irnich *et al.* (2010). Given the reduced costs of a path, the corresponding arc-variables in the arc-flow model can be fixed to zero in case the path-reduced costs are too high. Irnich *et al.* (2010) also show how the reduced costs of the arc-variables can be calculated from the reduced costs of paths.

6.5 Computational Results

The entire reflect+ algorithm is implemented in C++ and compiled with Microsoft Visual Studio 2017 into 64-bit single thread code. We use a column-generation framework that relies on CPLEX 12.10 for (re)optimizing the RMPs. Gurobi 9.1.0 is utilized when solving ILP formulations[3]. Default values are kept for all parameters of the two solvers, except from setting the number of threads to one. In addition, Gurobi exerts the barrier algorithm. The computational study was executed on a 64-bit Microsoft Windows 10 computer with Intel® Core™ i7-5930k

[3]This type of MIP solver was also used in the works of related SSP algorithm that we compare the reflect+ algorithm with.

CPU clocked at 3.5 GHz and 64 GB of RAM. The time limit is set to 3600 seconds per instance.

6.5.1 Results on SSP Benchmark Instances

We let the refined reflect+ algorithm run with all new components (underlined in green in Algorithm 2). Subsequently, we will then systematically analyze the impact of each main algorithmic component in Section 6.5.2, ending up with an *adaptive method* suggested in Section 6.5.3.

In our numerical investigation, we focus in particular on those classes of SSP benchmark instances that could only be solved by the ILP solver using the traditional reflect arc-flow model from Martinovic *et al.* (2020) to a limited extent or not at all within the time limit of one hour. These are the following instance classes:

- A and B, proposed in (Martinovic *et al.*, 2020). In fact, Class A contains instances whose constructions were already given in (Labbé *et al.*, 1995) and (Peeters and Degraeve, 2006), whereas Class B consists of randomly generated instances. Note that the original instances of Class A are not available anymore, and so they had to be rebuilt in (Martinovic *et al.*, 2020) sticking to the instructions from the literature. Although this class comprises three different sets A1, A2, and A3, only the first two found their way into (Martinovic *et al.*, 2020), whereas all three sets of instances have since then been accessible under `https://github.com/wotzlaff/ssp-data`. In this article, we therefore focus on the particularly difficult instances from A2 and A3[4] as well as B, since internal tests have shown that A1 is unsuitable for marking any differences between the various approaches.
- AI/ANI of Delorme *et al.* (2016), which are characterized by particularly many feasible patterns and the fact that any optimal solution is composed of waste-free item combinations only. Furthermore, it should be noted that all ANI instances have an integrality gap of one and are, thus, even more challenging.
- GI of Gschwind and Irnich (2016), which are mainly characterized by a huge bin size of up to 1.5 million and thus generally lead to very large networks (and also very large ILPs). Instances of this class are available at `https:`

[4]For the sake of completeness, we note that instances of A3 are very difficult, because they typically contain particularly many small item sizes. To be more precise, the set consists of 600 instances formed by 60 groups of 10 instances each. Any group is uniquely defined by a triple $(m, b_{\max}, l_{\max})$ denoting the number of item types, an upper bound for the values b_i, and an upper bound for the maximum item size, respectively. Any instance uses $L = 10000$, and the parameters are chosen from the following sets: $m \in \{50, 75, 100, 150, 250\}$, $b_{\max} \in \{20, 100, 200\}$, and $l_{\max} \in \{2500, 5000, 7500, 9999\}$.

`//logistik.bwl.uni-mainz.de/research/benchmarks/`.

We run reflect+ and compare the state-of-the-art results obtained with the compact formulation (6.3) both with our implementation and that of Martinovic *et al.* (2020). Table 6.2 provides aggregated results. For each subclass of instances (the number of instances is given in column #), we list the average computation times (*time*) in seconds and the number (#opt) of instances solved to proven optimality. Moreover, we indicate the best values by using bold font.

		Refined reflect+		ILP solver with reflect arc-flow model			
				Our implementation		Martinovic *et al.*	
Instances	#	*time*	#opt	*time*	#opt	*time*†	#opt
A2	1050	**124.5**	**1025**	250.9	1019	250.9	1011
A3	600	**202.6**	**575**	958.9	464	–	–
B	160	**112.7**	**160**	457.9	152	970.7	126
AI201	50	22.5	**50**	**20.3**	**50**	26.3	**50**
AI402	50	2024.4	28	**1801.1**	**32**	2016.9	31
ANI201	50	274.5	**50**	**206.0**	**50**	238.5	**50**
ANI402	50	3586.0	**1**	**3548.2**	**1**	3600.0	0
GI125	80	**29.6**	**80**	1794.8	40	1802.1	40
GI250	80	**153.2**	**80**	1836.1	40	1854.0	40
GI500	80	**707.2**	**79**	3004.6	32	3584.2	2
GI750	40	**1105.9**	**39**	3600.0	0	–	–
GI1000	40	**2351.6**	**33**	3600.0	0	–	–

Table 6.2: Reflect+ vs. ILP solver using the reflect arc-flow formulation on large benchmark instances of Martinovic *et al.* (2020); Gschwind and Irnich (2016); Delorme *et al.* (2016). (†: Martinovic *et al.* (2020) use Gurobi 8.0.1 on a PC with an AMD A10-5800K CPU with 16GB of RAM.)

The main observations can be summarized as follows:

- The most pronounced effects can be seen for the instance Classes A and B. In each of the subclasses examined, both the number of optimally solved instances and the computing time required could be improved compared to the state of the art. In total, 125 additional instances were successfully solved by reflect+, 111 of which coming from A3, which contains instances with a particularly large number of small items. Such items usually lead to many arcs in the networks under consideration and thus to many variables in the

associated ILPs. Consequently, the benefits of reflect+, compared to reflect arc-flow, are specifically large for such instances.

- For the ANI instances, it is clear for theoretical reasons that reflect+ can in principle lead to no improvement, since all levels of the algorithm must be executed due to the positive integrality gap. In particular, reflect+ must solve the complete reflect arc-flow ILP at the end (as well as handle the additional levels before in vain).
- For the AI instances, reflect+ does not lead to any noticeable improvements here either. Presumably, this is due to the fact that an optimal solution consists of nothing but waste-free patterns. It seems that the required representatives of these patterns, however, are not generated to a sufficient extent in the column-generation process and are therefore not present in the thinned-out graphs constructed from them.
- Whereas the ILP solver using the reflect arc-flow formulation of Martinovic *et al.* (2020) was only able to deal with some easier subclasses of the GI set (not involving the maximum bin size of 1.5 million units), reflect+ is able to successfully cope with almost all the instances, even if the number of item types is particularly high. Altogether, only nine GI instances (seven of which coming from the most challenging parameter setting with $m = 1000$) could not be solved by reflect+ in reasonable time. This is a clear improvement compared to the traditional reflect arc-flow approach.

Altogether, we highlight that reflect+ is able to solve almost 95% (2200 out of 2330) of the given benchmark instances to proven optimality. This is an increase of 17% compared to the previous state of the art, that is, reflect arc-flow solved with the most recent version of Gurobi. In particular, note that our results are on the same level (in fact, even slightly better for the largest set A2) as the ones obtained by a recently published sophisticated solution method of de Lima *et al.* (2023, Table 6). However, that approach only considered the sets A1, A2, and B, so that further comparisons are not possible within the scope of this article.

6.5.2 Evaluation of Algorithmic Components

In this subsection, we take a closer look at the actual influence and effect of the individual algorithmic components. To this end, we start with an important supplement to Table 6.2, in which we list (see Table 6.3) the average percentage computation time for the key elements of Algorithm 2. These data can also be used in the following experiments to support the argumentation there. It is clearly visible that column generation consumes the largest share (in some cases well over 80%) of the total computation time for most instance classes (especially any subset of A, B, and GI). Although we did not touch the in-depth investigation of our algorithmic components yet, this is a clear indicator that a stabilization

technique within column generation is mandatory. Contrary to this, only for the ANI instances this share is almost marginal. This is also due to the fact that Algorithm 2 runs through all levels and thus also has to solve the ILP belonging to the complete reflect arc-flow graph. Moreover, we also see that the subset-sum procedure is especially time-consuming for the GI instances. Here, there are particularly large bin capacities and at the same time a relatively large number of items (that is, together, many states of possible bin fillings), so that determining the actual best possible material usage is generally more difficult than with other instances.

Instances	Reflect+ levels	Subset-sum procedure	Column generation procedure	Fixation of arcs
A2	25.2%	0.7%	**73.9%**	0.2%
A3	17.1%	0.2%	**82.4%**	0.3%
B	0.7%	0.1%	**99.0%**	0.1%
AI201	**50.2%**	0.0%	49.8%	0.0%
AI402	**84.7%**	0.0%	15.3%	0.0%
ANI201	**95.8%**	0.0%	4.2%	0.0%
ANI402	**97.8%**	0.0%	2.2%	0.0%
GI125	1.8%	39.1%	**58.7%**	0.4%
GI250	0.4%	17.2%	**82.1%**	0.2%
GI500	1.9%	5.4%	**92.6%**	0.1%
GI750	4.1%	5.7%	**90.1%**	0.1%
GI1000	0.4%	2.8%	**96.7%**	0.0%

Table 6.3: Time shares of different components of reflect+.

Now we compare the fully-fledged reflect+ (referred to as Setting (1) in the following) with versions of reflect+ in which always one of main algorithmic components is switched off. Specifically, this means without the component:

(2): subset-sum procedure (only greedy heuristic),
(3): stabilized column generation (unstabilized),
(4): retransformation into a pure pattern-based solution (pattern columns only), and
(5): reduced costs from the LP relaxation of the reflect arc-flow model instead of path-reduced costs from the column generation.

Settings (2)–(5) represent the novelties in which our reflect+ algorithm differs from the primary reflect+ algorithm for the CSP presented in (Delorme and Iori, 2020). Table 6.4 summarizes the comparison between the settings (1)–(4) and Table 6.5 the comparison with setting (5) in a more detailed fashion.

		Setting (1) refined reflect+		Setting (2) only greedy heuristic		Setting (3) non-stabilized column generation		Setting (4) no pattern retransformation	
Instances	#	*time*	#opt	*time*	#opt	*time*	#opt	*time*	#opt
A2	1050	124.5	1025	139.3	1022	123.3	1025	122.5	1025
A3	600	202.6	575	192.2	578	198.8	574	197.4	575
B	160	112.7	160	33.1	160	111.4	160	110.2	160
Subtotal	1810	149.3	1760	147.4	1760	147.3	1759	146.2	1760
AI201	50	22.5	50	31.5	50	22.4	50	23.3	50
AI402	50	2024.4	28	2571.4	19	2193.8	25	2034.9	28
ANI201	50	274.5	50	305.4	50	293.4	50	278.8	50
ANI402	50	3586.0	1	3600.0	0	3582.6	1	3575.9	1
Subtotal	200	1476.9	129	1627.1	119	1523.1	126	1478.2	129
GI125	80	29.6	80	18.6	80	29.5	80	26.8	80
GI250	80	153.2	80	105.2	80	165.2	80	139.9	80
GI500	80	707.2	79	592.0	79	857.0	78	707.3	79
GI750	40	1105.9	39	639.7	39	1249.4	39	1169.0	38
GI1000	40	2332.7	34	1430.5	38	2557.3	23	2332.0	34
Subtotal	320	638.4	312	437.7	316	677.3	300	656.1	311
Total	2330	332.4	2201	314.3	2195	338.2	2185	330.6	2200

Table 6.4: Effects of adapted settings in the refined reflect+ algorithm.

We highlight the following facts:

- For instances of Classes A and B, no large differences can be observed. Most of the instances are solved by any of the four methods in approximately the same time, while other instances are too hard to deal with in all the settings. The most important observation probably is that the randomly generated instances of Class B do not seem to benefit from patterns having a good material utilization. As a result, omitting the subset-sum procedure leads to a considerable speed-up in this case.
- In contrast, however, we see impressively in the example of class AI402 in Table 6.4 that employing the subset-sum procedure can lead to a significant improvement in overall performance, since about 50% more instances can be solved optimally. It can be concluded that, unlike the greedy heuristic, the subset-sum method produces more "useful" patterns with little or no waste. Therefore, noticeably better feasible SSP solutions can be constructed in the thinned-out networks considered at the various stages of our overall algo-

rithm. Even if these improvements do not quite reach the level of the reflect arc-flow model of Martinovic *et al.* (2020), the findings make a significant contribution to improving the classical reflect+ algorithm presented by Delorme and Iori (2020) in an instance-specific way by clever preprocessing.

- The effects for the GI instances are best seen by looking at the most challenging subclass GI1000 in Table 6.4. For these instances (with a very large number of different item types), executing column generation (before entering Level 1 of our algorithm) is already very difficult and, consequently, omitting stabilization techniques leads to large performance degradation. More precisely, 11 to 16 instances less were solved than in all other settings.
- Comparing the number of optimally solved instances with Setting (1) and Setting (4) in Table 6.4, only one instance from GI750 stands out that can be solved with our refined reflect+ algorithm, but without transforming the DDOI columns it is not solved exactly within the time limit. Regarding the average computation times of these two settings, no significant advantage for the refined reflect+ is apparent at first glance. Except for random effects, the computation times hardly differ. Recall that an optimal solution of the AI/ANI instances must consist exclusively of exact patterns $\mathbf{p}$, i.e., patterns with $\mathbf{l}^\top \mathbf{p} = L$. Since this is also true for the continuous relaxation, no DOIs can be used in the LP solution, so no transformation of patterns occurs either. Although for the remaining instance sets it seems that the transformation brings only a minimal benefit, later we will demonstrate a systematic advantage of this action regarding the levels of reflect+ that need to be passed.
- It is noticeable that in the GI instances the use of the subset-sum procedure is rather harmful. In the classes GI125 to GI750 in Table 6.4, this can be seen above all in the sometimes significantly larger computation times compared to the mere application of the greedy heuristic. For the most difficult class GI1000, one even misses the optimal solution of four instances. To explain this, we give two reasons: On the one hand, running the subset-sum procedure itself takes significantly more time than an ordinary greedy procedure, especially when such auxiliary problems have to be solved in advance for up to $m = 1000$ different item types. However, from our point of view, the crucial factor is that, on the other hand, GI instances, in general, do not rely on low-waste patterns in the optimal solution. Substantial effort is invested by the subset-sum procedure to finally generate patterns that are not necessarily needed to successfully tackle the given instance. This obviously has a negative overall impact on the numerical results.
- Last but not least, it is slightly beneficial to use the path-reduced costs from the pattern-based formulation of the column-generation approach instead of

the reduced costs from the reflect arc-flow model in Level 4 of reflect+. This positive effect can be observed in Table 6.5 which contains only instances solved exactly in Level 4. The choice of instances is due to the fact that the reduced costs are only relevant in the last level of reflect+. Except for GI125 and ANI402, which each contain only one instance in this case, the average computation time is faster when path-reduced costs are used.

		Setting (1) refined reflect+			Setting (5) reduced cost from reflect arc-flow model		
Instances	#	Overall time	Time Level 4	#opt	Overall time	Time Level 4	#opt
A2	1050	4.1	1.3	1	6.3	1.3	1
AI201	50	54.3	23.5	6	62.7	31.5	6
AI402	50	1005.4	538.2	9	1217.7	745.6	9
ANI201	50	274.5	244.8	50	287.8	257.8	50
ANI402	50	2902.3	2655.7	1	2801.2	2568.7	1
GI125	80	44.6	31.5	1	21.5	8.6	1

Table 6.5: The two different ways to calculate the reduced costs in Level 4 of reflect+: comparison of instances solved optimally in Level 4.

Next, we analyze the relevance of the different levels for reflect+. For that purpose, Table 6.6a contains information up to which level the reflect+ algorithm was run through for the different instances. We use #term to specify the number of instances that terminate at this level (solved or timed out), supplementary #sol refers to the number of instances that were solved on this last level. A few instances of Class A2 could already be solved heuristically to optimality, these are included in the column of Level 1. Moreover, we see that most of the unsolved instances of Classes A and B are already timed out in Level 1, meaning that even stabilized column generation is not able to deal with these very hard instances in reasonable time. This provides further justification for why these instances have been very insensitive to the deactivation of individual components of the overall algorithm in Table 6.4. Concerning the AI set, most of the instances were solved to optimality in Level 2. Levels 3 and 4 were necessary to solve a few more instances and only for one instance the optimal solution could already be found in Level 1. Looking at AI402 isolated, it is noticeable that about half of the instances reaching Level 2 could not be solved on this level within the time limit. Due to the already mentioned integrality gap of at least one of all ANI instances, all instances of this

class can be solved only (if at all) on the last level of the reflect+ algorithm. In contrast, a different picture emerges for the GI set: Nearly all instances are solved to optimality already on the first level. Only one instance of the subclasses GI125, GI500, and GI750 is solved exactly in Level 2. Concluding, the patterns from the solution of the column-generation procedure are highly relevant for the GI set. While the patterns from the linear relaxation of the pattern-based formulation are useless for the AI instances, the larger set of all patterns generated during the column-generation process seems to be beneficial for the subsequent levels of the reflect+ algorithm.

			Level 1		Level 2		Level 3		Level 4	
Instances	#	#opt	#term	#sol	#term	#sol	#term	#sol	#term	#sol
A2	1050	1025	1008	985	41	39	–	–	1	1
A3	600	575	591	567	7	6	2	2	–	–
B	160	160	158	158	2	2	–	–	–	–
AI201	50	50	1	1	38	38	5	5	6	6
AI402	50	28	–	–	33	17	2	2	15	9
ANI201	50	50	–	–	–	–	–	–	50	50
ANI402	50	1	–	–	30	–	1	–	19	1
GI125	80	80	78	78	1	1	–	–	1	1
GI250	80	80	80	80	–	–	–	–	–	–
GI500	80	79	79	78	1	1	–	–	–	–
GI750	40	39	39	38	1	1	–	–	–	–
GI1000	40	34	34	34	–	–	–	–	–	–

(a) Default Setting (1): with pattern retransformation after column-generation process.

			Level 1		Level 2		Level 3		Level 4	
Instances	#	#opt	#term	#sol	#term	#sol	#term	#sol	#term	#sol
GI125	80	80	78	78	–	–	1	1	1	1
GI500	80	79	79	78	–	–	1	1	–	–
GI750	40	38	39	38	–	–	1	–	–	–

(b) Setting (4): without pattern retransformation after column-generation process (only rows with differing entries compared to part (a) are displayed).

Table 6.6: Comparison of reached levels of refined reflect+ algorithm.

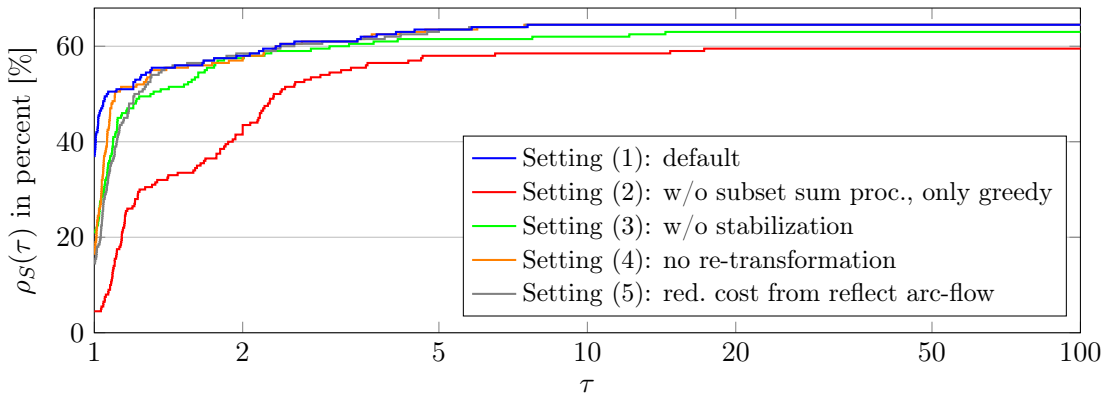
As already mentioned before, it is important to transform a solution containing

DDOI columns into a pure-pattern solution. A comparison of the data from Tables 6.6a and 6.6b provides a rationale for this: Table 6.6b is structured in the same way as the prior Table 6.6a and contains the data of Setting (4) from Table 6.4. For more clarity, Table 6.6b lists only instance sets with instances that were solved to optimality on a different (that is, later) level than with the refined reflect+. We can see the reason why reflect+ solved one instance more to optimality of subclass GI750 in comparison to Setting (4). While the transformed pure-patterns from the column-generation process suffice to build the optimal integer solution, the patterns of the column-generation solution including DDOI columns lost valuable information that is necessary for the optimal solution. Therefore, this instance needs to run through the next Level 3 where the time limit is exceeded and no solution is found. In GI500 and GI750, there are two additional instances with the same behavior, but in these cases the instances can be solved successfully in Level 3. These examples do not lead to substantial improvements of the average computational time, but it is clearly a systematic benefit.

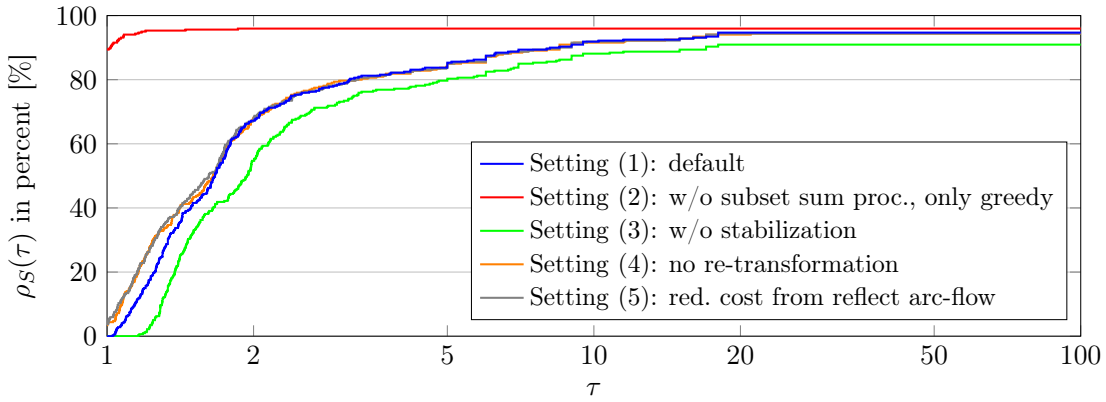
Finally, we compare the different settings with the help of performance profiles (Dolan and Moré, 2002). For a set of algorithms $\mathcal{S}$ (the five settings in our case), the performance profile $\rho_S(\tau)$ of an algorithm $S \in \mathcal{S}$ describes the ratio of instances that can be solved by S within a factor τ compared to the fastest algorithm, i.e. $\rho_S(\tau) = \left|\left\{I \in \mathcal{I} : t_I^S / t_I^* \leq \tau\right\}\right| / |\mathcal{I}|$ in which $\mathcal{I}$ is the set of instances, t_I^S is the computation time of algorithm S when applied to instance $I \in \mathcal{I}$, and t_I^* is the smallest computation time among all algorithms of set $\mathcal{S}$ for instance I. Unsolved instances are taken into account with $t_I^S = \infty$ (assuming also that $t_I^* = \infty$ gives $t_I^S / t_I^* = \infty$). In particular, the values of a profile at the two points $\tau = 1$ and $\tau = \infty$ can be interpreted well: $\rho_S(1)$ is the percentage of instances for which S is the fastest, and $\rho_S(\infty)$ is the percentage of instances that are solved by S within the time limit.

Figure 6.2 shows the performance profiles comparing the five settings separately for the AI/ANI and GI instances. We did not include the sets A and B here due to the only marginal differences observed earlier. Many of the above findings can be confirmed now. For the AI/ANI instances, see Figure 6.2a, the default setting of the reflect+ algorithm that integrates all four new algorithmic components is for 37% of the instances the fastest algorithm. Almost the entire profile $\rho_{\text{Setting (1)}}(\tau)$ is above the other profiles of all other settings, characterizing the default reflect+ algorithm as the clear winner here. The subset-sum procedure is the most beneficial algorithmic component for the AI/ANI instances, as can be seen from the profile of Setting (2). Settings (3)–(5) are a little bit inferior to the default setting with only Setting (3) solving significantly fewer AI/ANI instances to proven optimality.

The relationship between the default setting and Setting (2) not using the subset-sum procedure is completely opposite for the GI instance, see Figure 6.2b. The



(a) For the AI/ANI instances.



(b) For the GI instances.

Figure 6.2: Performance profiles of the five different settings of the reflect+ algorithm; note the logarithmic scale of the τ-axis.

reflect+ version without the subset-sum procedure shows a striking performance: It is the fastest algorithm for 89% of the GI instances. It is also superior regarding the instances solved exactly with the one-hour time limit, but the absolute number of optima is comparable to all other settings but Setting (3). This is what we already found in Table 6.4 and explained by the need of stabilizing the column-generation phase in particular for the large-scale GI instances.

6.5.3 Adaptive Reflect+ Algorithm

In the previous subsection, we saw that, on average, the modifications we highlighted in Algorithm 2 result in a very competitive solution procedure, but on

closer inspection, the effect of the subset-sum procedure is very different for each benchmark set. Both Table 6.4 and Figure 6.2 suggest that either heuristic procedure can be helpful in providing an appropriate pattern pool. The subset-sum procedure seemed to be useful when patterns with very good material usage must be present in an optimal solution. The need for patterns with very good material usage might be a priori seen by checking whether the sum of all given item lengths is an integer multiple of the threshold length. Note that AI instances were constructed so that this property holds exactly. Instead of always choosing either the columns of the greedy heuristic or those of the subset-sum procedure, we propose the following *adaptive reflect+ algorithm* using the following criterion: Invoke the subset-sum procedure if and only if

$$\frac{\mathbf{l}^\top \mathbf{b}}{L} - \left\lfloor \frac{\mathbf{l}^\top \mathbf{b}}{L} \right\rfloor \leq 0.1. \tag{6.5}$$

The intention behind this heuristic selection rule is that patterns with good material utilization are somewhat more likely to be in an optimal solution if the aggregated item sizes just exceed an integer multiple of L by a reasonably small percentage (here: 10%). Note that Condition (6.5) just depends on the input-data of an instance and can be easily checked in advance. The results of all three variants

- always subset-sum (that is, Setting (1))
- always greedy (that is, Setting (2))
- adaptive approach according to criterion (6.5)

are listed in Table 6.7.

First, it can be seen that the adaptive reflect+ algorithm does not lead to any change in the AI/ANI instances. In fact, this is clear given the construction of these instances, because there the total length of all items always matches exactly an integer multiple of the threshold length L. Thus, the subset-sum procedure is always executed. If we look at the other two types of benchmark instances instead, we notice that the adaptive reflect+ algorithm leads to moderate improvements there on average. While, compared to Setting (1), two instances less are solved for set A2, five more instances of the more difficult set A3 can now also be treated successfully. Moreover, one can see a clear speedup for the instances of Class B. Compared to Setting (2), the computation times of the adaptive reflect+ algorithm are on a similar level for Classes A and B, but the latter overall yields more instances solved optimally. Looking at the other benchmark sets, we see that Setting (2) performs worst for the AI/ANI instances, while the same is true for Setting (1) applied to the GI instances. For the latter, however, Setting (2) and the adaptive approach again deliver similarly good results. Overall, most instances can be solved optimally with the adaptive reflect+ algorithm; moreover, the average

		Setting (1)		Setting (2)		adaptive reflect+ algorithm			
Instances	#	*time*	#opt	*time*	#opt	#subset	#greedy	*time*	#opt
A2	1050	124.5	1025	139.3	1022	123	927	137.3	1023
A3	600	202.6	575	192.2	578	59	541	171.2	580
B	160	112.7	160	33.1	160	12	148	38.7	160
AI201	50	22.5	50	31.5	50	50	0	22.5	50
AI402	50	2024.4	28	2571.4	19	50	0	2024.4	28
ANI201	50	274.5	50	305.4	50	50	0	274.5	50
ANI402	50	3586.0	1	3600.0	0	50	0	3586.0	1
GI125	80	29.6	80	18.6	80	5	75	19.4	80
GI250	80	153.2	80	105.2	80	9	71	109.6	80
GI500	80	707.2	79	592.0	79	9	71	596.9	79
GI750	40	1105.9	39	639.7	39	1	39	664.2	39
GI1000	40	2332.7	34	1430.5	38	6	34	1570.3	38
Total	2330	332.4	2201	314.3	2195	424	1906	298.7	2208

Table 6.7: The adaptive reflect+ algorithm chooses Setting (1) or Setting (2) depending on the characteristics of the instance.

computation time is about 10% below that of Setting (1), i.e., the most convincing method so far.

The superiority of the adaptive approach is also once again evident in the performance profile depicted in Figure 6.3. Due to the quite similar results for Classes A and B, only the instances AI/ANI and GI were used here.

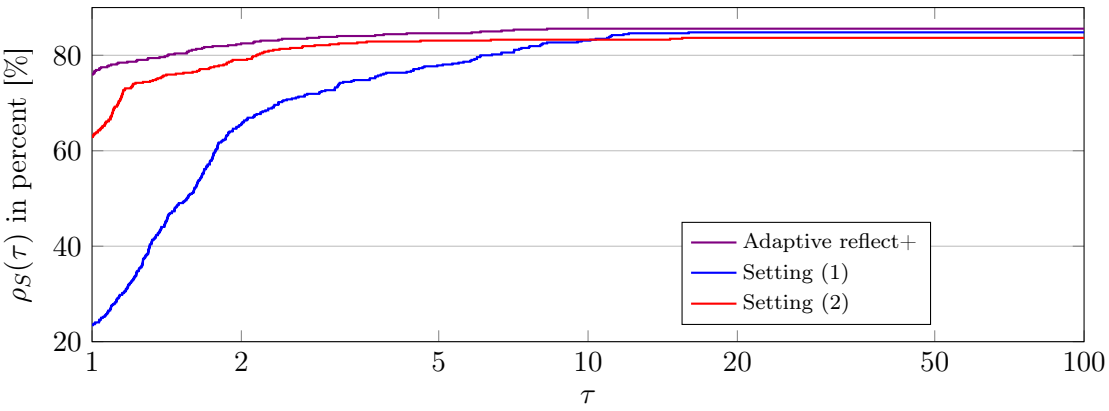


Figure 6.3: Performance profile (for the AI/ANI and GI instances) for the adaptive reflect+ algorithm in comparison with Setting (1) and (2) of the reflect+ algorithm.

6.6 Conclusions

In this article, we have introduced a refined reflect+ algorithm with algorithmic components tailored to the SSP. In general, this solution strategy is a multi-stage procedure, based largely on the fact that an optimal solution can often be found even if not all arcs of the ordinary reflect model are included, see (Delorme and Iori, 2020). Therefore, feasible points in thinned-out graphs are first determined, and then an attempt is made to prove their optimality using appropriate bounds. If this fails, an adapted (larger) network is considered in the next step. On the basis of extensive computational tests with different challenging benchmark classes, it can be shown that a large number of instances can be solved optimally for the first time with this approach. Within the context of these numerical investigations, we have worked out precisely the performance contributions of any of the refinements we have applied to the basic reflect+ algorithm. Thus, we succeed in obtaining valuable insights into the general numerical behavior of various instance classes and are able to derive concrete recommendations for their efficient practical treatment. The result of these studies is an adaptive reflect+ algorithm showing the most convincing properties (on average).

For future research, two main issues are particularly interesting:

(1) One effect observed during the computational evaluation remains unclear to us even after analyzing in detail the instance classes and their behavior of reaching and passing the different levels of the reflect+ algorithm: While using the subset-sum procedure to equip the column-generation phase with columns of very good (exact) patterns, producing no or almost no loss, this component is clearly beneficial for the difficult AI/ANI benchmark, but rather harmful for the large-scale GI benchmark. Future research might find new ways to analyze and better understand this part of the column-generation process and its impact on subsequent phases of reflect+.

(2) The work of de Lima *et al.* (2023) exploits pseudo-polynomial formulations in an innovative fashion presenting an approach using column generation and its reduced-cost information to define a kind of core subproblems solved separately with an ILP solver. This is possible because of a completely new branching scheme for the subproblems. Overall, the approach of de Lima *et al.* combines important techniques also used in reflect+ in a new way. It would be interesting to see results of this and other new approaches also for the challenging A3, AI/ANI, and GI benchmark sets.

Bibliography

Agoston, K. (2019). The effect of welding on the one-dimensional cutting-stock problem: The case of fixed firefighting systems in the construction industry. *Advances in Operations Research*. Article ID 6507054.

Alvim, A., Ribeiro, C., Glover, F., and Aloise, D. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, **10**, 205–229.

Arbib, C., Di Iorio, C., Marinelli, F., and Rossi, F. (2002). Cutting and reuse: an application from automobile component manufacturing. *Operations Research*, **50**(6), 923–934.

Assmann, S. F. (1983). *Problems in discrete applied mathematics*. Ph.D. thesis, Mathematics Department, Massachusetts Institute of Technology.

Assmann, S. F., Johnson, D. S., Kleitman, D. J., and Leung, J. Y.-T. (1984). On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, **5**, 502–525.

Belov, G. and Scheithauer, G. (2002). A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, **141**(2), 274–294.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Brandão, F. and Pedroso, J. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers and Operations Research*, **69**, 56–67.

Caprara, A., Dell'Amico, M., Diaz Diaz, J., Iori, M., and Rizzi, R. (2015). Friendly bin packing instances without integer round-up property. *Mathematical Programming B*, **150**, 5–17.

Chen, Y., Song, X., Ouelhadj, D., and Cui, Y. (2019). A heuristic for the skiving and cutting stock problem in paper and plastic film industries. *International Transactions in Operational Research*, **26**(1), 157–179.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. The MIT Press, 3rd edition.

de Lima, V. L., Alves, C., Clautiaux, F., Iori, M., and de Carvalho, J. M. V. (2022). Arc flow formulations based on dynamic programming: Theoretical foundations and applications. *European Journal of Operational Research*, **296**(1), 3–21.

de Lima, V. L., Iori, M., and Miyazawa, F. K. (2023). Exact solution of network flow models with strong relaxations. *Mathematical Programming*, **197**, 813–846.

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.

Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2005). *Column Generation*. Springer, New York, NY.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research*, **29**(6), 1092–1104.

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**(6), 849–859.

Gilmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—Part II. *Operations Research*, **11**(6), 863–888.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

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.

Irnich, S., Desaulniers, G., Desrosiers, J., and Hadjar, A. (2010). Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, **22**(2), 297–313.

Johnson, M., Rennick, C., and Zak, E. (1997). Skiving addition to the cutting stock problem in the paper industry. *SIAM Review*, **39**(3), 472–483.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management Science*, **6**(4), 366–422.

Kartak, V., Ripatti, A., Scheithauer, G., and Kurz, S. (2015). Minimal proper non-IRUP instances of the one-dimensional cutting stock problem. *Discrete Applied Mathematics*, **187**, 120–129.

Kłosowski, G., Kozłowski, E., and Gola, A. (2018). Integer linear programming in optimization of waste after cutting in the furniture manufacturing. In A. Burduk and D. Mazurkiewicz, editors, *Intelligent Systems in Production Engineering and Maintenance – ISPEM 2017*, pages 260–270, Cham. Springer International Publishing.

Labbé, M., Laporte, G., and Martello, S. (1995). An exact algorithm for the dual bin packing problem. *Operations Research Letters*, **17**, 9–18.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, **45**(3), 414–424.

Martinovic, J. and Scheithauer, G. (2016a). Integer linear programming models for the skiving stock problem. *European Journal of Operational Research*, **251**(2), 356–368.

Martinovic, J. and Scheithauer, G. (2016b). Integer rounding and modified integer rounding for the skiving stock problem. *Discrete Optimization*, **21**, 118–130.

Martinovic, J. and Scheithauer, G. (2016c). The proper relaxation and the proper gap of the skiving stock problem. *Mathematical Methods of Operations Research*, **84**(3), 527–548.

Martinovic, J. and Scheithauer, G. (2018). Characterizing IRDP-instances of the skiving stock problem by means of polyhedral theory. *Optimization*, **67**(10), 1797–1817.

Martinovic, J., Jorswieck, E., Scheithauer, G., and Fischer, A. (2017). Integer linear programming formulations for cognitive radio resource allocation. *IEEE Wireless Communication Letters*, **6**(4), 494–497.

Martinovic, J., Scheithauer, G., and Valério de Carvalho, J. (2018). A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, **266**(2), 458–471.

Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., and Strasdat, N. (2020). Improved flow-based formulations for the skiving stock problem. *Computers and Operations Research*, **113**, 104770.

Nemhauser, G. and Wolsey, L. (1988). *Integer and Combinatorial Optimization*. Wiley, New York.

Peeters, M. and Degraeve, Z. (2006). Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem. *European Journal of Operational Research*, **170**(2), 416–439.

Rao, M. (1976). On the cutting stock problem. *Journal of the Computer Society of India*, **7**, 35–39.

Shapiro, J. F. (1968). Dynamic programming algorithms for the integer programming problem—I: The integer programming problem viewed as a knapsack type problem. *Operations Research*, **16**(1), 103–121.

Stadtler, H. (1988). A comparison of two optimization procedures for 1- and 1 1/2-dimensional cutting stock problems. *Operations-Research-Spektrum*, **10**(2), 97–111.

Tragos, E., Zeadally, S., Fragkiadakis, A., and Siris, V. (2013). Spectrum assignment in cognitive radio networks: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, **15**(3), 1108–1135.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, **141**(2), 253–273.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Wang, D., Xiao, F., Zhou, L., and Liang, Z. (2020). Two-dimensional skiving and cutting stock problem with setup cost based on column-and-row generation. *European Journal of Operational Research*, **286**(2), 547–563.

Wolsey, L. (1977). Valid inequalities, covering problems and discrete dynamic programs. *Annals of Discrete Mathematics*, **1**, 527–538.

Zak, E. J. (2003). The skiving stock problem as a counterpart of the cutting stock problem. *International Transactions in Operational Research*, **10**(6), 637–650.

Chapter 7

Conclusion

The main goal of this thesis was to advance the development of exact algorithms for picker routing and packing problems, two fundamental classes of combinatorial optimization problems with significant practical relevance. This chapter summarizes the main findings and highlights the methodological contributions.

The first part of the thesis addressed various picker routing problems. In Chapter 2, we presented the first exact approach that solves the SPRP-SS with heuristic routing policies, including `traversal`, `return`, `largest gap`, `midpoint`, and `composite`. The method employs a DP formulation that is subsequently transformed into an IP model and solved using an MIP solver. Additionally, we considered the impact of class-based storage policies on routing efficiency. Through extensive computational experiments, the study evaluated various routing and storage policy combinations, revealing how strategic alignment between these two aspects can significantly improve picking performance. Among all implemented heuristic routing policies, the combination of `largest gap` and the *within-aisle* storage policy yields the shortest tours. The effects of article scattering vary by storage policy. The results provide valuable insights into the design of efficient warehouse operations and demonstrate the potential of exact and model-based approaches for better decision-making in practical logistics environments.

In Chapter 3, we proposed two new MIP formulations, EG and RG, for the SPRP-SS that replace sets of parallel edges in the original shortest-path network by Heßler and Irnich (2024) with more compact subnetworks. These modifications significantly reduce the network size. Importantly, the new formulations yield linear-size MIP models in both the number of aisles and pick positions, improving upon the quadratic complexity of the original model. Computational experiments demonstrated substantial efficiency gains: the new models, especially RG, reduced the number of edges by more than two-thirds and achieved average speedups of more than fourfold for large instances, with computation times remaining below two seconds even for the most complex cases.

In Chapter 4, we provided the first problem-specific exact algorithm for the SPRP-SS in parallel-aisle warehouses with more than two blocks. The proposed

B&C method employs a hierarchy of subtour elimination constraints, efficiently solved through max-flow/min-cut procedures. We compared our algorithm with a competing B&C algorithm that uses a GTSP model, since SPRP-SS instances can be solved as GTSP instances. Computational results show that the new formulation achieves smaller integrality gaps and yields more optimal solutions within time limits than the GTSP-based approach, especially for large and complex instances. For smaller cases, however, the GTSP-based method remains the faster option.

In Chapter 5, we introduced two new optimization problems, MZPRP and BMZPRP, which incorporate picker routing in zoned warehouses with scattered storage. Methodologically, we presented an exact MIP-based solution approach for these problems, extending the state-space network formulation of Heßler and Irnich (2024), which is itself grounded in the dynamic program of Ratliff and Rosenthal (1983). The resulting network-flow MIP formulation integrates all three decision levels – article-to-zone assignment, zone-level SKU selection, and within-zone routing – into a unified optimization model. Computational experiments on large-scale instances demonstrated that the proposed approach efficiently solves the MZPRP to optimality within milliseconds or seconds and achieves strong performance for the BMZPRP, with only a few large instances unsolved within the time limit. The results show that zone picking can reduce total tour costs by up to 12.6% compared to single-zone picking, while the BMZPRP achieves well-balanced picker workloads with only minor increases in total tour length.

Chapter 6 formed the second part of the thesis in terms of subject matter and considered the SSP as a representative of packing problems. We presented a refined reflect+ algorithm for the problem. The approach uses a multi-stage framework: it first identifies feasible solutions in reduced graphs, and then iteratively expands the network only when necessary to prove optimality, significantly reducing computational effort. Extensive computational experiments showed that this approach can solve many challenging benchmark instances to proven optimality for the first time. The study also analyzed the individual contributions of the algorithmic refinements, leading to an adaptive reflect+ variant with robust performance across various instance classes.

Bibliography

Agoston, K. (2019). The effect of welding on the one-dimensional cutting-stock problem: The case of fixed firefighting systems in the construction industry. *Advances in Operations Research*. Article ID 6507054.

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey.

Alvim, A., Ribeiro, C., Glover, F., and Aloise, D. (2004). A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, **10**, 205–229.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2003). Concorde-03.12.19. Website. `https://www.math.uwaterloo.ca/tsp/concorde/index.html`.

Arbib, C., Di Iorio, C., Marinelli, F., and Rossi, F. (2002). Cutting and reuse: an application from automobile component manufacturing. *Operations Research*, **50**(6), 923–934.

Assmann, S. F. (1983). *Problems in discrete applied mathematics*. Ph.D. thesis, Mathematics Department, Massachusetts Institute of Technology.

Assmann, S. F., Johnson, D. S., Kleitman, D. J., and Leung, J. Y. T. (1984). On a dual version of the one-dimensional packing problem. *Journal of Algorithms*, **5**(4), 502–525.

Bartholdi, III, J. J. and Hackman, S. T. (2019). *Warehouse & Distribution Science*. Atlanta, GA 30332-0205 USA, release 0.98.1 edition. Online only, `https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf`.

Behnisch, P., Glock, C. H., Grosse, E. H., and Ries, J. M. (2017). Auf dem Weg zum Warehouse 4.0? – Zum aktuellen Stand der Automatisierung in der Lagerhaltung. Technical Report 84808, Institut for Business Studies (BWL), Department of Business Administration, Economics and Law, Darmstadt Technical University. (in German).

Belov, G. and Scheithauer, G. (2002). A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, **141**(2), 274–294.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Bock, S. and Boysen, N. (2023). Routing replenishment workers: The prize collecting traveling salesman problem in scattered storage warehouses. *INFORMS Journal on Computing*, **36**(1), 3–20.

Bock, S., Bomsdorf, S., Boysen, N., and Schneider, M. (2025). A survey on the traveling salesman problem and its variants in a warehousing context. *European Journal of Operational Research*, **322**(1), 1–14.

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.

Brandão, F. and Pedroso, J. (2016). Bin packing and related problems: general arc-flow formulation with graph compression. *Computers and Operations Research*, **69**, 56–67.

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.

Caprara, A., Dell'Amico, M., Diaz Diaz, J., Iori, M., and Rizzi, R. (2015). Friendly bin packing instances without integer round-up property. *Mathematical Programming B*, **150**, 5–17.

Çelik, M. and Süral, H. (2018). Order picking in parallel-aisle warehouses with multiple blocks: complexity and a graph theory-based heuristic. *International Journal of Production Research*, **57**(3), 888–906.

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

Chen, F., Wang, H., Qi, C., and Xie, Y. (2013). An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, **66**(1), 77–85.

Chen, Y., Song, X., Ouelhadj, D., and Cui, Y. (2019). A heuristic for the skiving and cutting stock problem in paper and plastic film industries. *International Transactions in Operational Research*, **26**(1), 157–179.

Choe, K.-I. (1991). *Aisle-based order pick systems with batching, zoning, and sorting*. Dissertation, Georgia Institute of Technology.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. Computer science. MIT Press, Cambridge, Massachusetts and London, England, 3rd edition.

Cormier, G. and Gunn, E. A. (1992). A review of warehouse models. *European Journal of Operational Research*, **58**(1), 3–13.

Cornuéjols, G., Fonlupt, J., and Naddef, D. (1985). The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, **33**(1), 1–27.

Daniels, R. L., Rummel, J. L., and Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, **105**(1), 1–17.

de Koster, R. and van der Poort, E. (1998). Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. *IIE Transactions*, **30**(5), 469–480.

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.

de Koster, R. B. M., Le-Duc, T., and Zaerpour, N. (2012). Determining the number of zones in a pick-and-sort order picking system. *International Journal of Production Research*, **50**(3), 757–771.

de Lima, V. L., Alves, C., Clautiaux, F., Iori, M., and de Carvalho, J. M. V. (2022). Arc flow formulations based on dynamic programming: Theoretical foundations and applications. *European Journal of Operational Research*, **296**(1), 3–21.

de Lima, V. L., Iori, M., and Miyazawa, F. K. (2023). Exact solution of network flow models with strong relaxations. *Mathematical Programming*, **197**, 813–846.

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.

Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors (2005). *Column Generation*. Springer, New York, NY.

Desrosiers, J., Lübbecke, M., Desaulniers, G., and Gauthier, J. B. (2024). Branch-and-price. Les Cahiers du GERAD G-2024-36, Groupe d'études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91**(2), 201–213.

Drury, J. (1988). Towards more efficient order picking. Technical report, The Institute of Materials Management, Cranfield, UK.

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research*, **29**(6), 1092–1104.

Fischetti, M., Salazar-Gonzalez, J.-J., and Toth, P. (2002). The generalized traveling salesman and orienteering problems. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*, chapter 13, pages 609–662. Kluwer, Dordrecht.

Frazelle, E. (2002). *World-Class Warehousing and Material Handling*. McGraw-Hill, New York.

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.

Gibson, D. R. and Sharp, G. P. (1992). Order batching procedures. *European Journal of Operational Research*, **58**(1), 57–67.

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**(6), 849–859.

Gilmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—Part II. *Operations Research*, **11**(6), 863–888.

Goeke, D. and Schneider, M. (2021). Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, **33**(2), 419–835.

Gray, A. E., Karmarkar, U. S., and Seidmann, A. (1992). Design and operation of an order-consolidation warehouse: Models and application. *European Journal of Operational Research*, **58**(1), 14–36.

Grosse, E. H., Glock, C. H., Jaber, M. Y., and Neumann, W. P. (2015). Incorporating human factors in order picking planning models: framework and research opportunities. *International Journal of Production Research*, **53**(3), 695–717.

Gschwind, T. and Irnich, S. (2016). Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, **28**(1), 175–194.

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.

Gue, K. R. and Meller, R. D. (2009). Aisle configurations for unit-load warehouses. *IIE Transactions*, **41**(3), 171–182.

Gutin, G. and Punnen, A., editors (2002). *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization*. Kluwer, Dordrecht.

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

Haouassi, M., Kergosien, Y., Mendoza, J. E., and Rousseau, L.-M. (2025). The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, **63**(4), 1304–1325.

Hausman, W. H., Schwarz, L. B., and Graves, S. C. (1976). Optimal storage assignment in automatic warehouse systems. *Management Science*, **22**, 629–638.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, **126**(1), 106–130.

Henn, S., Koch, S., Gerking, H., and Wäscher, G. (2013). A U-shaped layout for manual order-picking systems. *Logistics Research*, **6**(4), 245–261.

Heskett, J. L. (1963). Cube-per-order index–a key to warehouse stock location. *Transportation and Distribution Management*, **3**(1), 27–31.

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. https://logistik.bwl.uni-mainz.de/files/2022/04/LM-2022-03.pdf.

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*, **36**(6), 1417–1435.

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.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer.

Irnich, S., Desaulniers, G., Desrosiers, J., and Hadjar, A. (2010). Path-reduced costs for eliminating arcs in routing and scheduling. *INFORMS Journal on Computing*, **22**(2), 297–313.

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*, MOS-SIAM Series on Optimization, chapter 1, pages 1–33. SIAM, Philadelphia, PA.

Jane, C. C. (2000). Storage location assignment in a distribution center. *International Journal of Physical Distribution & Logistics Management*, **30**(1), 55–71.

Jane, C.-C. and Laih, Y.-W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. *European Journal of Operational Research*, **166**(2), 489–496.

Johnson, M., Rennick, C., and Zak, E. (1997). Skiving addition to the cutting stock problem in the paper industry. *SIAM Review*, **39**(3), 472–483.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management Science*, **6**(4), 366–422.

Kartak, V., Ripatti, A., Scheithauer, G., and Kurz, S. (2015). Minimal proper non-IRUP instances of the one-dimensional cutting stock problem. *Discrete Applied Mathematics*, **187**, 120–129.

Khan, I., Maurer, O., Pätzold, J., Pszona, P., and Salchow, J.-D. (2024). Joint order selection, allocation, batching and picking for large scale warehouses. Technical report, arXiv 2401.04563. `https://doi.org/10.48550/arXiv.2401.04563`.

Kłosowski, G., Kozłowski, E., and Gola, A. (2018). Integer linear programming in optimization of waste after cutting in the furniture manufacturing. In A. Burduk and D. Mazurkiewicz, editors, *Intelligent Systems in Production Engineering and Maintenance – ISPEM 2017*, pages 260–270, Cham. Springer International Publishing.

Korte, B. and Vygen, J. (2018). *Combinatorial Optimization: Theory and Algorithms*. Springer, Berlin, Heidelberg, 6th edition.

Labbé, M., Laporte, G., and Martello, S. (1995). An exact algorithm for the dual bin packing problem. *Operations Research Letters*, **17**, 9–18.

Löffler, M., Boysen, N., and Schneider, M. (2022). Picker routing in AGV-assisted order picking systems. *INFORMS Journal on Computing*, **34**(1), 440–462.

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.

Lüke, L., Hessenius, A., and Irnich, S. (2026). A linear-size model for the single picker routing problem with scattered storage. *European Journal of Operational Research*, **332**(1), 132–143.

Martello, S., Pisinger, D., and Toth, P. (1999). Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, **45**(3), 414–424.

Martinovic, J. and Scheithauer, G. (2016a). Integer linear programming models for the skiving stock problem. *European Journal of Operational Research*, **251**(2), 356–368.

Martinovic, J. and Scheithauer, G. (2016b). Integer rounding and modified integer rounding for the skiving stock problem. *Discrete Optimization*, **21**, 118–130.

Martinovic, J. and Scheithauer, G. (2016c). The proper relaxation and the proper gap of the skiving stock problem. *Mathematical Methods of Operations Research*, **84**(3), 527–548.

Martinovic, J. and Scheithauer, G. (2018). Characterizing IRDP-instances of the skiving stock problem by means of polyhedral theory. *Optimization*, **67**(10), 1797–1817.

Martinovic, J., Jorswieck, E., Scheithauer, G., and Fischer, A. (2017). Integer linear programming formulations for cognitive radio resource allocation. *IEEE Wireless Communication Letters*, **6**(4), 494–497.

Martinovic, J., Scheithauer, G., and Valério de Carvalho, J. (2018). A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, **266**(2), 458–471.

Martinovic, J., Delorme, M., Iori, M., Scheithauer, G., and Strasdat, N. (2020). Improved flow-based formulations for the skiving stock problem. *Computers and Operations Research*, **113**, 104770.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2019). Optimal order picker routing in the chevron warehouse. *IISE Transactions*, **52**(6), 665–687.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2020a). Optimal order picker routing in a conventional warehouse with two blocks and arbitrary starting and ending points of a tour. *International Journal of Production Research*, **58**(17), 5337–5358.

Masae, M., Glock, C. H., and Grosse, E. H. (2020b). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, **224**, 107564.

Masae, M., Glock, C. H., and Vichitkunakorn, P. (2021). A method for efficiently routing order pickers in the leaf warehouse. *International Journal of Production Economics*, **234**, 108069.

Mellema, P. M. and Smith, C. A. (1988). Simulation analysis of narrow-aisle order selection systems. In *Proceedings of the 20th conference on Winter simulation*, pages 597–602.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problems. *Journal of Association for Computing Machinery*, **7**, 326–329.

Nemhauser, G. and Wolsey, L. (1988). *Integer and Combinatorial Optimization*. Wiley, New York.

Öztürkoğlu, Ö. and Hoser, D. (2019). A discrete cross aisle design model for order-picking warehouses. *European Journal of Operational Research*, **275**(2), 411–430.

Öztürkoğlu, Ö., Gue, K. R., and Meller, R. D. (2012). Optimal unit-load warehouse designs for single-command operations. *IIE Transactions*, **44**(6), 459–475.

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

Parikh, P. J. and Meller, R. D. (2008). Selecting between batch and zone order picking strategies in a distribution center. *Transportation Research Part E: Logistics and Transportation Review*, **44**(5), 696–719.

Parikh, P. J. and Meller, R. D. (2009). Estimating picker blocking in wide-aisle order picking systems. *IIE Transactions*, **41**(3), 232–246.

Park, Y. H. and Webster, D. B. (1989). Design of class-based storage racks for minimizing travel time in a three-dimensional storage system. *International Journal of Production Research*, **27**(9), 1589–1601.

Peeters, M. and Degraeve, Z. (2006). Branch-and-price algorithms for the dual bin packing and maximum cardinality bin packing problem. *European Journal of Operational Research*, **170**(2), 416–439.

Petersen, C. G. (1995). Routeing and storage policy interaction in order picking operations. In *Decision Sciences Institute Proceedings*, volume 3, pages 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. (2002). Considerations in order picking zone configuration. *International Journal of Operations & Production Management*, **22**(7), 793–805.

Petersen, C. G. and Schmenner, R. W. (1999). An evaluation of routing and volume-based storage policies in an order picking operation. *Decision Sciences*, **30**(2), 481–501.

Petersen, C. G., Aase, G. R., and Heiser, D. R. (2004). Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, **34**(7), 534–544.

Pinedo, M. L. (2022). *Scheduling*. Springer, Cham, Switzerland, 6. edition.

Prunet, T., Absi, N., and Cattaruzza, D. (2025). A note on the complexity of the picker routing problem in multi-block warehouses and related problems. *Annals of Operations Research*, **347**, 1595–1605.

Pugliese, L. D. P. and Guerriero, F. (2013). A reference point approach for the resource constrained shortest path problems. *Transportation Science*, **47**(2), 247–265.

Rao, M. (1976). On the cutting stock problem. *Journal of the Computer Society of India*, **7**, 35–39.

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.

Revenant, P., Cambazard, H., and Catusse, N. (2025). A note about a transition of Ratliff and Rosenthal's order picking algorithm for rectangular warehouses. *Operations Research Letters*, **62**, 107325.

Roodbergen, K. J. (2001). *Layout and Routing Methods for Warehouses*. Ph.D. thesis, RSM Erasmus University, 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.

Roodbergen, K. J. and Vis, I. F. (2006). A model for warehouse layout. *IIE transactions*, **38**(10), 799–811.

Saylam, S., Çelik, M., and Süral, H. (2023). The min–max order picking problem in synchronised dynamic zone-picking systems. *International Journal of Production Research*, **61**(7), 2086–2104.

Saylam, S., Çelik, M., and Süral, H. (2024). Arc routing based compact formulations for picker routing in single and two block parallel aisle warehouses. *European Journal of Operational Research*, **313**(1), 225–240.

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.

Schmidt, J. and Irnich, S. (2022). New neighborhoods and an iterated local search algorithm for the generalized traveling salesman problem. *EURO Journal on Computational Optimization*, **10**, 100029.

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.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, Berlin Heidelberg.

Shapiro, J. F. (1968). Dynamic programming algorithms for the integer programming problem—I: The integer programming problem viewed as a knapsack type problem. *Operations Research*, **16**(1), 103–121.

Singh, K. N. and van Oudheusden, D. L. (1997). A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, **97**(3), 571–579.

Stadtler, H. (1988). A comparison of two optimization procedures for 1- and 1 1/2-dimensional cutting stock problems. *Operations-Research-Spektrum*, **10**(2), 97–111.

Su, Y., Zhu, X., Yuan, J., Teo, K. L., Li, M., and Li, C. (2023). An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, **305**(1), 222–239.

Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, **22**(2), 215–225.

Theys, C., Bräysy, O., Dullaert, W., and Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, **200**(3), 755–763.

Tompkins, J., White, J., Bozer, Y., Frazelle, E., and Tanchoco, J. (2003). *Facilities Planning*. John Wiley & Sons, Hoboken, NJ.

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

Tragos, E., Zeadally, S., Fragkiadakis, A., and Siris, V. (2013). Spectrum assignment in cognitive radio networks: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, **15**(3), 1108–1135.

Valério de Carvalho, J. M. (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, **86**, 629–659.

Valério de Carvalho, J. M. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research*, **141**(2), 253–273.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

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., Ramaekers, K., Caris, A., and de Koster, R. B. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, **267**(1), 1–15.

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. S. (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.

Wang, D., Xiao, F., Zhou, L., and Liang, Z. (2020). Two-dimensional skiving and cutting stock problem with setup cost based on column-and-row generation. *European Journal of Operational Research*, **286**(2), 547–563.

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, **95**, 139–150.

Weidinger, F. and Boysen, N. (2018). Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, **52**(6), 1412–1427.

Weidinger, F., Boysen, N., and Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, **274**(2), 501–515.

Wildt, C., Weidinger, F., and Boysen, N. (2025). Picker routing in scattered storage warehouses: an evaluation of solution methods based on TSP transformations. *OR Spectrum*, **47**, 35–66.

Wolsey, L. (1977). Valid inequalities, covering problems and discrete dynamic programs. *Annals of Discrete Mathematics*, **1**, 527–538.

Wolsey, L. A. (1998). *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York.

Yu, M. and De Koster, R. B. (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, **198**(2), 480–490.

Zak, E. J. (2003). The skiving stock problem as a counterpart of the cutting stock problem. *International Transactions in Operational Research*, **10**(6), 637–650.