

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

Stefan Irnich^{a,*}, Laura Lücke^a

^a*Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

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.

Keywords: warehousing; picker routing; multiple blocks; dynamic programming; integer programming

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 (Boysen et al., 2019; van Gils et al., 2018). 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., 2020). 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. These figures explain why research on order picking operations is extensive and of high practical relevance.

*Corresponding author.

Email addresses: irnich@uni-mainz.de (Stefan Irnich), lueke@uni-mainz.de (Laura Lücke)

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 ([Hefler and Irnich, 2022](#)). 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, 2001b](#)). 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 \(2001a\)](#) 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 [Hefler and Irnich \(2022\)](#), 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).

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 framework, 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 \(2001b\)](#) 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. 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 [Hefler and Irnich \(2024\)](#), who presented a generic approach to extend DP state space to incorporate scattered storage (for an overview, see Table 1 in [Hefler 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. \(2025\)](#) who present a linear-size model that reduced the number of parallel arcs compared to the model of [Hefler 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. [Hefler 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.

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 $J = \{1, 2, \dots, m\}$ and number the aisles $j \in J$ from left to right. The

blocks are numbered from back (=top) to front (=bottom) by $k \in B = \{1, 2, \dots, b\}$. 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 $k + 1 \in K = \{1, 2, \dots, b, b + 1\}$ that lie perpendicular to the aisles, see Figure 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. 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 1 shows a small instance with three blocks, four cross-aisles, five aisles, and $10 = |S|$ different articles to be collect. An optimal solution for the unit-demand case is also shown.

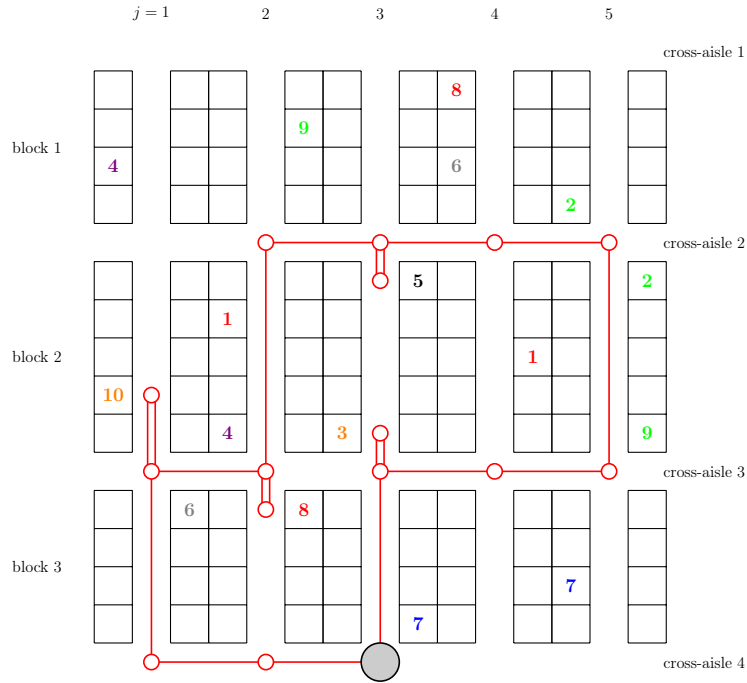


Figure 1: Solution of a 3-SPRP-SS instance with four cross-aisles and five aisles.

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.

1.3. Structure

Section 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 3. Section 4 presents the associated branch-and-cut algorithm. Computational results are presented and discussed in Section 5. The paper closes with final conclusions in Section 6.

2. The Relaxed State Space for the k -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 (2001a)'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 2.1 with the description of the classical state spaces of Ratliff and Rosenthal and Roodbergen and de Koster (2001a). Subsequently, the additional transitions as introduced by Heßler and Irnich (2024) are presented in Section 2.2. The relaxed state space that we use for the k -SPRP-SS is presented in Section 2.3.

2.1. Classical State Spaces

The DPs of Ratliff and Rosenthal (1983) and Roodbergen and de Koster (2001a) 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 O, 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.

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

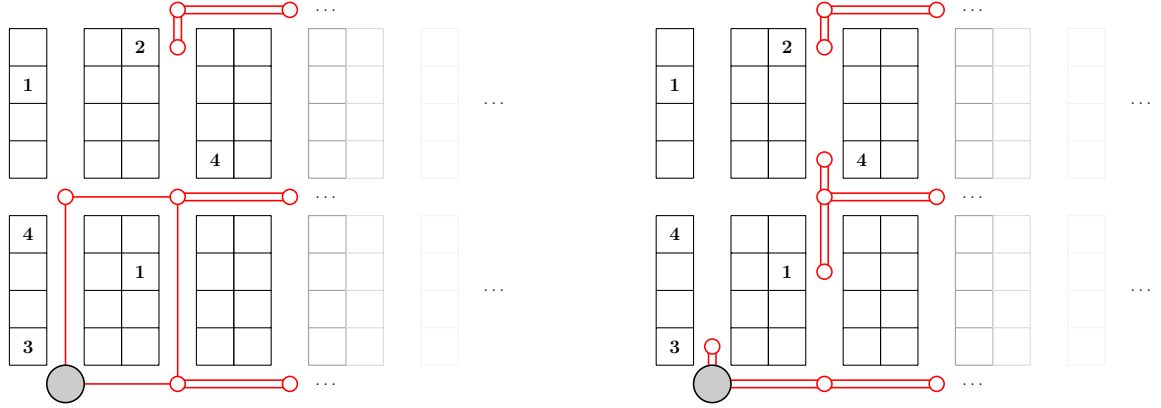
$$\mathcal{S}_1 = \{\text{UU.1}, \text{OE.1}, \text{EO.1}, \text{EE.1}, \text{EE.2}, \text{OO.0}, \text{OO.1}\},$$

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

$$\mathcal{S}_2 = \{\text{OOO.0}, \text{OOO.1}, \text{EOO.1}, \text{OE.1}, \text{OOE.1}, \text{EE.1}, \text{EOE.1}, \text{OEE.1}, \text{EEE.1}, \text{UUO.1}, \text{UOU.1}, \text{OUU.1}, \text{EUU.1}, \text{UEU.1}, \text{UUE.1}, \\ \text{EE.2}, \text{EOE.2}, \text{OEE.2}, \text{EEE.2a-bc}, \text{EEE.2b-ac}, \text{EEE.2c-ab}, \text{EUU.2}, \text{UEU.2}, \text{UUE.2}, \text{EEE.3}\}$$

(see Roodbergen and de Koster, 2001a). 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.

Figure 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.



(a) One component continues through cross-aisle 1 (denoted by **a** in (Roodbergen and de Koster, 2001a)), 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 2: Two PTSs, both with an even degree at all intersection points in aisle 3, but different unconnected components.

However, there exist different possibilities of (dis)connected cross-aisles or blocks, so the state includes this information. In Figure 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 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 (2001a), 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. Regarding the sets E_{jk}^{aisle} , 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. The action **void** occurs if a sub-aisle is not entered at all.

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 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, there exists at most one relevant aisle action per type in every aisle, e.g., only one **top**, one **bottom**, and one **gap**, because each demanded article is stored at a unique pick position in the warehouse. 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, [Hefler 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.

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 [Hefler and Irnich \(2024\)](#) through the introduction of demand-covering constraints in the MIP. We will use the same idea in Section 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, several states 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 $\{0, U, E\}$ with an even number of U entries. It is straightforward to see that the relaxed state space has only $N(b)$ different states, where N follows the recursion $N(b) = 2N(b-1) + bN(b-2)$.

The advantage of the relaxed state space is that the number of states per stage grows moderately. Table 1 compares the number of states of the relaxed state space with the original DP of [Ratcliff and Rosenthal \(1983\)](#) for single-block warehouses, the DP for two-block warehouses by [Roodbergen and de Koster \(2001b\)](#), and the DP of [Pansart et al. \(2018\)](#) for multi-block warehouses.

Number b of blocks	1	2	3	4	5	6	7	8	9	10
Number of states per stage										
in DP of Ratcliff 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 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 [Ratcliff 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 **EUQUOE** into **EUEUOE** (interesting positions underlined). The cross-aisle action **210100** translates the same state **EUOUOE** into **EUOUO0**.

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 3 and 4).

3. Network-Flow Formulation of the b -SPRP-SS

Recall that (V, E) describes the relaxed state space introduced in Section 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**. The set of edges with a non-negative supply of article $s \in S$, we denote by $e \in E_s$. 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:

$$z_{b\text{-SPRP-SS}} = \min \sum_{e \in E} c_e x_e \quad (1a)$$

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

$$\sum_{e \in E_s} b_{se} x_e \geq q_s \quad \forall s \in S \quad (1c)$$

$$\{\text{subtour-elimination constraints}\} \quad (1d)$$

$$x_e \in \{0, 1\} \quad \forall e \in E \quad (1e)$$

The objective function (1a) minimizes the total length of the picker tour. Flow conservation through the relaxed state space is enforced by constraints (1b). Constraints (1c) guarantee that the demand for each article on the pick list is fully met. The *subtour-elimination constraints* (SECs) (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 4.1 does not need to be connected. However, subtours are clearly infeasible. Finally, the domain restrictions of the decision variables are specified in (1e).

Some remarks about the structure of formulation (1) are due: For the SPRP (no scattering), the equivalence of DP and linear programming implies that the SPRP can be solved solely with (1a), (1b), and (1e) 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., (1a), (1b), (1c), and (1e).

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 (1) has $\mathcal{O}(N(b) \cdot mb)$ variables. There are $|V|$ flow-conservation constraints (1b) and $|S|$ demand-covering constraints (1c). Since the number of connectivity constraints (1d) is exponential in the warehouse size, we will use a B&C algorithm described next to solve the model.

4. Branch-and-Cut Algorithm for the b -SPRP-SS

In this section, we formally introduce the SECs to be used in formulation (1). A major complication stems from the fact that already testing whether or not a tentative solution 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. The support graph is much smaller than the relaxed state space. Therefore, feasibility checking and the separation of violated SECs can be performed efficiently.

We assume that a partial formulation of (1) is solved, i.e., a model consisting of (1a), (1b), (1c), and (1e) with no or just a subset of the SECs (1d). Let this solution be $\bar{x} = (\bar{x}_e)_{e \in E}$.

4.1. Warehouse Graph and Support Graph

We motivate the warehouse graph and support graph with the help of an example. Figure 3a shows a small 3-SPRP-SS instance with five aisles, four cross-aisles, and three blocks. This picker tour 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 $b+1 \notin B$).

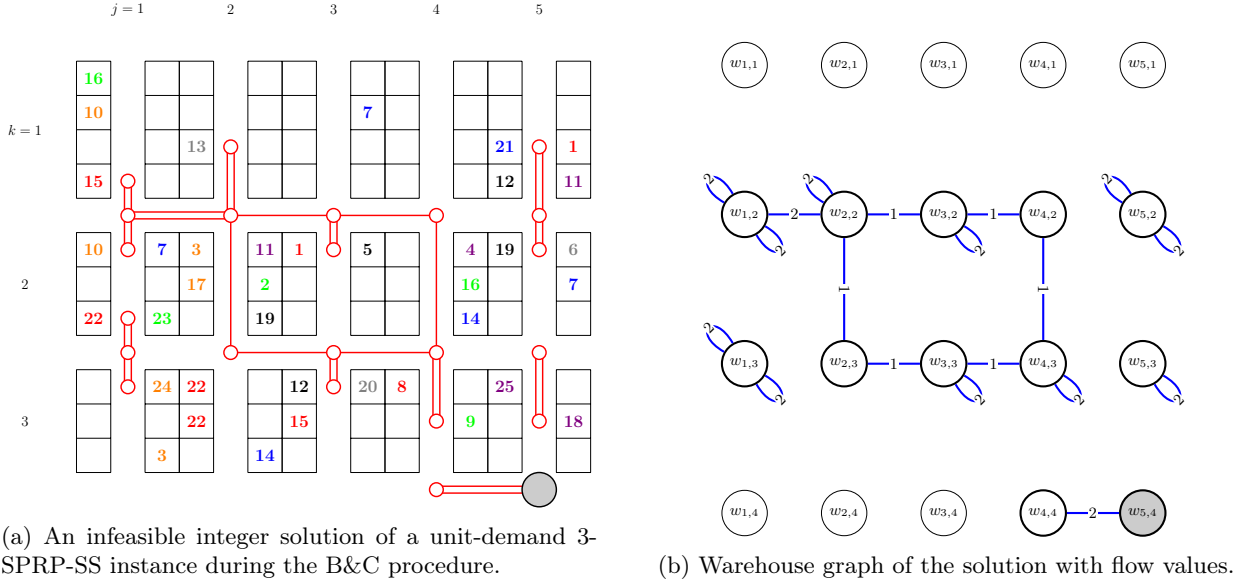


Figure 3: Infeasible tentative solution of a 3-SPRP-SS instance and the corresponding warehouse graph.

The warehouse graph $(W, \mathcal{E})$ is an undirected multi-graph with loops. The set $\mathcal{E}$ of edges of the warehouse graph includes all possible ‘horizontal’ and ‘vertical’ connections between the intersection points, which are defined by

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

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 $e \in E_j^{\text{cross}}$ 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

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

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}^{1pass} (E_{jk}^{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

$$\bar{f}_{w_{jk}, w_{j,k+1}} = \sum_{e \in E_{jk}^{\text{ipass}}} \bar{x}_e + \sum_{e \in E_{jk}^{\text{2pass}}} 2\bar{x}_e. \quad (2b)$$

The solution shown in Figure 3a 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 3b, this subtour is represented by the edge $(w_{4,4}, w_{5,4})$ with the flow value 2.

Up to now, we are not able to detect the two subtours in aisle $j = 5$ and the short subtour in aisle $j = 1$ depicted in Figure 3a. The reason is that aisle actions with movements in an aisle not traversing the 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.

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

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

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

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

where E_{jk}^{top} , E_{jk}^{bottom} , and E_{jk}^{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 $E_{jk}^{\text{collect}} = E_{jk}^{\text{ipass}} \cup E_{jk}^{\text{2pass}} \cup E_{jk}^{\text{top}} \cup E_{jk}^{\text{bottom}} \cup E_{jk}^{\text{gap}}$.

In the example in Figure 3a, 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 3b.

Finally, let a possibly fractional solution $\bar{x} = (\bar{x}_e)_{e \in E}$ with flow values $\bar{f} = (\bar{f}_e)_{e \in \mathcal{E}}$ defined by (2) be given. We define the support graph $(W_{\bar{f}}, \mathcal{E}_{\bar{f}}, \bar{f})$ as the edge-weighted graph spanned by the edges with positive flow. Formally, the edge set is $\mathcal{E}_{\bar{f}} = \{e \in \mathcal{E} : \bar{f}_e > 0\}$ and the vertex set is the span $W_{\bar{f}} = W(\mathcal{E}_{\bar{f}})$. In Figure 3b, the support graph contains only the vertices w_{jk} of the warehouse graph that are depicted in bold.

4.2. Subtour-Elimination Constraints

Different types of SECs are needed to exclude all possible subtours. Figure 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 $\mathcal{S}$, $\delta(\mathcal{S})$ 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 (2a), (2b), and (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 $\mathcal{S}_s$ 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.

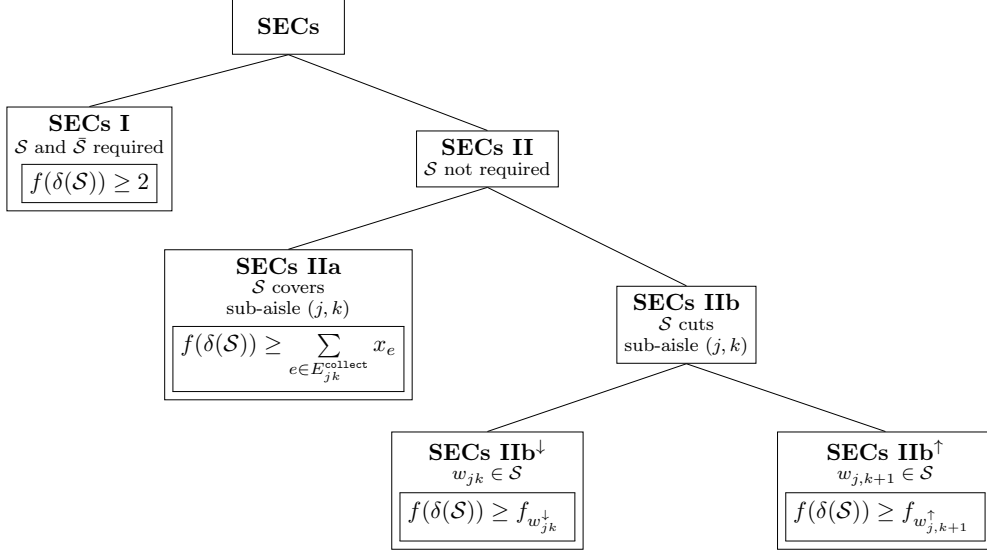


Figure 4: Overview of the different types of SECs.

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 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.

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

$$f(\delta(\mathcal{S})) \geq 2. \quad (3a)$$

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

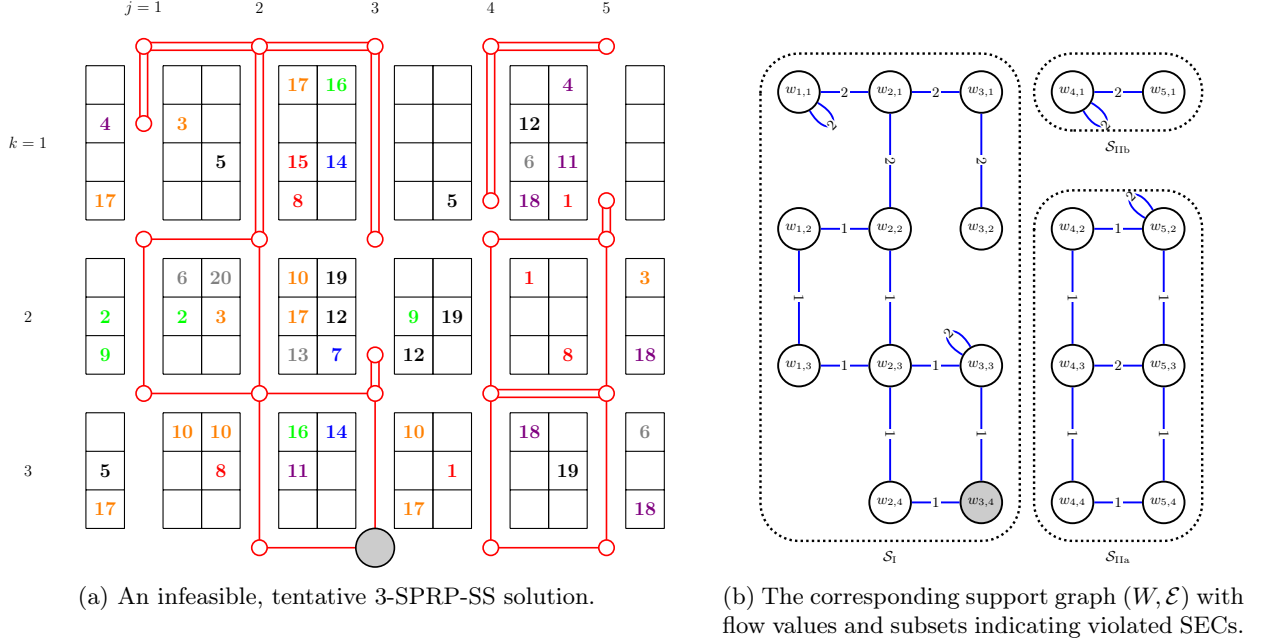


Figure 5: Different types of violated SECs for an infeasible, tentative solution of a unit-demand 3-SPRP-SS instance.

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 (2b). Similarly, positive values of $f_{w_{j,k}}^\downarrow$ or $f_{w_{j,k+1}}^\uparrow$ indicate that the sub-aisle (j, k) is at least entered from the upper or lower end, implying $f(\delta(\mathcal{S})) \geq f_{w_{j,k}}^\downarrow$ and $f(\delta(\mathcal{S})) \geq f_{w_{j,k+1}}^\uparrow$.

The three inequalities can be aggregated into

$$f(\delta(\mathcal{S})) \geq 2 \cdot \sum_{e \in E_{jk}^{\text{collect}}} x_e. \quad (3b)$$

Inequality (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 right-hand side assumes values between 0 and 1 compared to $f_{w_{jk}, w_{j, k+1}}$, $f_{w_{j,k}}^\downarrow$, and $f_{w_{j,k+1}}^\uparrow$ which can assume values between 0 and 2. Indeed, inequality (3b) dominates the three above inequalities.

Figure 5 shows another example for the subset $\mathcal{S}_{IIa} = \{w_{jk} : 4 \leq j \leq 5, 2 \leq k \leq 4\}$. This subset $\mathcal{S}_{IIa}$ does not include the depot. Moreover, the four sub-aisles included in $\mathcal{S}_{IIa}$ 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 $\mathcal{S}_{IIa}$. As a result, $\mathcal{S}_{IIa}$ is not required. Since the sub-aisles $(4, 2)$, $(4, 3)$, $(5, 2)$, and $(5, 3)$ are covered by $\mathcal{S}_{IIa}$, the SEC (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

$$f(\delta(\mathcal{S})) \geq f_{w_{jk}}^\downarrow \quad (3c)$$

which is a SEC of Type Iib[↓]. If it separates the lower part, this implies

$$f(\delta(\mathcal{S})) \geq f_{w_{j,k+1}}^\uparrow \quad (3d)$$

which is a SEC of Type Iib[↑].

The subset $\mathcal{S}_{\text{Iib}} = \{w_{4,1}, w_{5,1}\}$ highlighted in Figure 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 (3c) of Type Iib[↓] is violated for $\mathcal{S}_{\text{Iib}}$.

Note that the subset $\mathcal{S}_{\text{Iia}}$ can serve as an example of a violated SEC (3d) of Type Iib[↑] with $(j, k) = (5, 2)$.

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 flow values $\bar{f} = (\bar{f}_\epsilon)_{\epsilon \in \mathcal{E}}$. 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 (3a) of Type I is violated (if both $\mathcal{S}$ and $\bar{\mathcal{S}}$ are required), or the SEC (3b) of Type Iia is violated (if $\mathcal{S}$ covers a sub-aisle), or the SEC (3c) or (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). In the following, we will determine minimum cuts in which the subsets $\mathcal{S}$ and/or $\bar{\mathcal{S}}$ must include certain subsets $X \subset W$ or $Y \subset \bar{W}$ 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 , creating a new support graph, or by adding internal edges with a high cost in a preparatory step. In the following, we will use the term (X, Y) -max flow/min cut for such a problem.

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 (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).

SECs of Type Iia. For SECs (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 $2 \cdot \sum_{e \in E_{jk}^{\text{collect}}} \bar{x}_e$. If the flow value is smaller, a violated SEC (3b) of Type IIa is found for $\mathcal{S}$ and sub-aisle (j, k) .

SECs of Type IIb. For SECs (3d) and (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 $(w_{jk}, w_{j,k+1})$ -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 $\bar{f}_{w_{jk}^\downarrow}$ or $\bar{f}_{w_{j,k+1}^\uparrow}$ (or both), then a violated SEC (3c) or (3d) is found for sub-aisle (j, k) , respectively.

5. Computational Results

This section on computational experiments and results discusses the computational setup in Section 5.1, describes the generation of b -SPRP-SS instances used in the experiments in Section 5.2, analyzes the lower bounds at the root node of the B&C in Section 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 5.4.

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 8.5.0. 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 reimplementaion in C++ of the B&C algorithm of Fischetti et al. (2002) provided by Hefler and Irnich (2024), where the callable library of CPLEX is also used. In (Hefler 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.

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 5.3 and 5.4.

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 Hefler 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 1). When possible, all blocks have the same number of cells; otherwise they differ by no more than one cell. 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.

Number of blocks aisles		Scatter factor	Number of articles							
			$a = 20$		$a = 30$		$a = 40$		$a = 50$	
			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
	$m = 5$	$\alpha = 1.2$	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 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).

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 $\{1, \dots, \min(6, \sum_p b_{sp})\}$. 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/>.

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[↓], and IIb[↑] (see Section 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 $LB_{GTSP}^1(I)$ and $LB_{NF}^1(I)$ the respective lower bounds for a unit-demand 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 algo-

rithm. 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 .

Table 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.

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 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 GTS without UB provides 781 proven optimal solutions, GTSP with UB provides 750, NF without UB provides 1115, and NF with UB provides 1186. The superiority of the NF model can be attributed to the much better dual bound (see Section 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 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 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). For larger scatter factors, NF is generally recommended for larger instances. 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.

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.

			Number of articles															
Number of blocks	aisles	Scatter factor	$a = 20$				$a = 30$				$a = 40$				$a = 50$			
			GTSP		NF		GTSP		NF		GTSP		NF		GTSP		NF	
			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	8	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	28	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	17	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	39	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	15	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	35	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	11	10	7	3	12
		2.5	20	20	17	18	13	12	9	16	6	4	7	11	0	0	2	5
		3.0	20	20	13	19	1	1	8	15	1	1	6	13	0	0	0	4
		Subtotal	80	80	61	72	54	53	48	67	47	44	31	51	30	27	16	36

Table 3: Number of instances solved to proven optimality.

Table 5 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.

6. Conclusions and Outlook

In this paper, we have presented the first problem-tailored exact algorithm for the 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 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 SPRP-SS 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

			Number of articles							
Number of		Scatter factor	$a = 20$		$a = 30$		$a = 40$		$a = 50$	
blocks	aisles		GTSP	NF	GTSP	NF	GTSP	NF	GTSP	NF
$b = 3$	$m = 5$	$\alpha = 1.2$	0.03	0.42	0.10	0.43	0.75	0.47	5.98	0.37
		1.8	0.63	0.93	16.85	0.80	199.05	1.00	300.00	0.88
		2.5	14.03	1.21	233.91	1.29	300.00	1.61	300.00	2.15
		3.0	90.01	1.38	300.00	1.45	300.00	3.88	300.00	3.48
		Subtotal	26.18	0.99	137.97	0.99	201.09	1.74	229.58	1.72
	$m = 10$	$\alpha = 1.2$	0.03	2.96	0.09	4.35	0.48	6.11	1.40	6.17
		1.8	0.36	3.08	5.43	4.93	47.42	5.79	207.78	16.03
		2.5	3.85	5.65	181.93	8.72	274.97	17.14	300.00	20.27
		3.0	35.47	9.63	283.93	17.81	300.00	41.04	300.00	45.12
		Subtotal	9.93	5.33	117.85	8.95	156.52	17.52	205.47	21.90
$b = 4$	$m = 5$	$\alpha = 1.2$	0.04	6.38	0.10	5.37	0.38	3.06	2.22	2.29
		1.8	0.45	7.32	4.52	9.12	101.42	10.94	264.61	8.36
		2.5	14.17	9.91	167.57	11.45	300.00	19.36	300.00	28.03
		3.0	76.61	9.64	296.60	13.11	300.00	26.30	300.00	34.15
		Subtotal	22.81	8.31	117.20	9.76	176.63	14.91	219.39	18.21
	$m = 10$	$\alpha = 1.2$	0.04	101.93	0.10	100.61	0.31	120.80	0.89	165.21
		1.8	0.45	64.79	2.48	86.31	28.67	172.51	184.91	226.10
		2.5	4.55	66.93	149.29	126.61	253.68	198.63	300.00	256.77
		3.0	19.80	85.98	289.19	153.19	292.66	173.75	300.00	274.02
		Subtotal	6.21	79.91	110.26	116.68	143.83	166.42	199.46	230.52

Table 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.

			Number of articles							
Number of blocks	aisles	Scatter factor	$a = 20$		$a = 30$		$a = 40$		$a = 50$	
			$C = 20$	30	$C = 20$	30	$C = 20$	30	$C = 20$	30
$b = 3$	$m = 5$	$\alpha = 1.2$	0.39	0.45	0.45	0.42	0.47	0.47	0.33	0.40
		1.8	0.94	0.92	0.71	0.88	0.84	1.15	0.86	0.90
		2.5	0.91	1.52	1.08	1.49	1.36	1.86	2.09	2.21
		3.0	1.52	1.25	1.12	1.77	3.29	4.46	3.91	3.05
		Subtotal	0.94	1.03	0.84	1.14	1.49	1.98	1.80	1.64
	$m = 10$	$\alpha = 1.2$	2.96	2.96	4.56	4.14	5.20	7.02	6.97	5.36
		1.8	2.87	3.30	4.59	5.27	3.24	8.35	18.23	13.82
		2.5	6.04	5.25	6.28	11.17	24.63	9.64	23.92	16.63
		3.0	7.96	11.29	17.30	18.32	44.13	37.95	47.71	42.54
		Subtotal	4.96	5.70	8.18	9.73	19.30	15.74	24.21	19.59
$b = 4$	$m = 5$	$\alpha = 1.2$	4.75	8.00	4.65	6.09	2.30	3.81	1.89	2.70
		1.8	4.97	9.66	5.93	12.31	8.71	13.17	6.41	10.31
		2.5	4.91	14.91	5.10	17.80	12.42	26.29	25.41	30.65
		3.0	6.86	12.42	6.31	19.90	22.40	30.20	25.58	42.71
		Subtotal	5.37	11.25	5.50	14.03	11.46	18.37	14.82	21.59
	$m = 10$	$\alpha = 1.2$	76.36	127.51	63.52	137.71	80.82	160.78	154.10	176.32
		1.8	40.58	88.99	44.56	128.05	169.09	175.93	195.27	256.94
		2.5	57.21	76.64	128.43	124.79	195.77	201.49	251.60	261.93
		3.0	55.24	116.73	119.22	187.15	146.37	201.13	257.69	290.35
		Subtotal	57.35	102.47	88.93	144.42	148.01	184.83	214.66	246.39

Table 5: Average solution time (in seconds) of the B&C algorithm using NF with UB for $C = 20$ and 30.

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 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).

References

- R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.
- N. Boysen, R. de Koster, and F. Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411, 2019. doi:[10.1016/j.ejor.2018.08.023](https://doi.org/10.1016/j.ejor.2018.08.023).
- H. Cambazard and N. Catusse. Fixed-parameter algorithms for rectilinear Steiner tree and rectilinear traveling salesman problem in the plane. *European Journal of Operational Research*, 270(2):419–429, 2018. doi:[10.1016/j.ejor.2018.03.042](https://doi.org/10.1016/j.ejor.2018.03.042).
- M. Çelik and H. Süral. 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, 2018. doi:[10.1080/00207543.2018.1489154](https://doi.org/10.1080/00207543.2018.1489154).
- R. L. Daniels, J. L. Rummel, and R. Schantz. A model for warehouse order picking. *European Journal of Operational Research*, 105(1):1–17, 1998. doi:[10.1016/S0377-2217\(97\)00043-X](https://doi.org/10.1016/S0377-2217(97)00043-X).
- R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501, 2007. doi:[10.1016/j.ejor.2006.07.009](https://doi.org/10.1016/j.ejor.2006.07.009).
- J. Drury. Towards more efficient order picking. Technical report, The Institute of Materials Management, Cranfield, UK, 1988.
- M. Fischetti, J.-J. Salazar-Gonzalez, and P. Toth. 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, 2002. doi:[10.1007/0-306-48213-4_13](https://doi.org/10.1007/0-306-48213-4_13).
- E. Frazelle. *World-Class Warehousing and Material Handling*. McGraw-Hill, New York, 2002.
- D. Goeke and M. Schneider. Modeling single-picker routing problems in classical and modern warehouses. *INFORMS Journal on Computing*, 33(2):436–451, 2021. doi:[10.1287/ijoc.2020.1040](https://doi.org/10.1287/ijoc.2020.1040).
- J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21, 2007. doi:[10.1016/j.ejor.2006.02.025](https://doi.org/10.1016/j.ejor.2006.02.025).
- R. W. Hall. Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, 25(4):76–87, 1993. doi:[10.1080/07408179308964306](https://doi.org/10.1080/07408179308964306).
- M. Haouassi, Y. Kergosien, J. E. Mendoza, and L.-M. Rousseau. The picker routing problem in mixed-shelves, multi-block warehouses. *International Journal of Production Research*, 63(4):1304–1325, 2025. doi:[10.1080/00207543.2024.2374845](https://doi.org/10.1080/00207543.2024.2374845).
- K. Hefler and S. Irnich. A note on the linearity of Ratliff and Rosenthal's algorithm for optimal picker routing. *Operations Research Letters*, 50(2):155–159, 2022. doi:[10.1016/j.orl.2022.01.014](https://doi.org/10.1016/j.orl.2022.01.014).
- K. Hefler and S. Irnich. Exact solution of the single picker routing problem with scattered storage. *INFORMS Journal on Computing*, 36(6):1417–1435, 2024. doi:[10.1287/ijoc.2023.0075](https://doi.org/10.1287/ijoc.2023.0075).
- L. Lüke, K. Hefler, and S. Irnich. The single picker routing problem with scattered storage: modeling and evaluation of routing and storage policies. *OR Spectrum*, 46(3):909–951, 2024. doi:[10.1007/s00291-024-00760-4](https://doi.org/10.1007/s00291-024-00760-4).
- L. Lüke, A. Hessenius, and S. Irnich. A linear-size model for the single picker routing problem with scattered storage. *European Journal of Operational Research*, 2025. doi:[10.1016/j.ejor.2025.11.002](https://doi.org/10.1016/j.ejor.2025.11.002).
- M. Masae, C. H. Glock, and E. H. Grosse. Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, 224:107564, 2020. doi:[10.1016/j.ijpe.2019.107564](https://doi.org/10.1016/j.ijpe.2019.107564).
- L. Pansart, N. Catusse, and H. Cambazard. Exact algorithms for the order picking problem. *Computers and Operations Research*, 100:117–127, 2018. doi:[10.1016/j.cor.2018.07.002](https://doi.org/10.1016/j.cor.2018.07.002).
- C. G. Petersen. An evaluation of order picking routing policies. *International Journal of Operations and Production Management*, 17(11):1098–1111, 1997. doi:[10.1108/01443579710177860](https://doi.org/10.1108/01443579710177860).
- T. Prunet, N. Absi, and D. Cattaruzza. A note on the complexity of the picker routing problem in multi-block warehouses and related problems. *Annals of Operations Research*, 347:1595–1605, 2025. doi:[10.1007/s10479-025-06481-3](https://doi.org/10.1007/s10479-025-06481-3).
- H. D. Ratliff and A. S. Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983. doi:[10.1287/opre.31.3.507](https://doi.org/10.1287/opre.31.3.507).
- P. Revenant, H. Cambazard, and N. Catusse. A note about a transition of ratliff and rosenthal's order picking algorithm for rectangular warehouses. *Operations Research Letters*, 62:107325, 2025. doi:[10.1016/j.orl.2025.107325](https://doi.org/10.1016/j.orl.2025.107325).

- K. J. Roodbergen and R. de Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001a. doi:[10.1016/s0377-2217\(00\)00177-6](https://doi.org/10.1016/s0377-2217(00)00177-6).
- K. J. Roodbergen and R. de Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001b. doi:[10.1080/00207540110028128](https://doi.org/10.1080/00207540110028128).
- S. Saylam, M. Çelik, and H. Süral. 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, 2024. doi:[10.1016/j.ejor.2023.08.018](https://doi.org/10.1016/j.ejor.2023.08.018).
- M. Schiffer, N. Boysen, P. S. Klein, G. Laporte, and M. Pavone. Optimal picking policies in e-commerce warehouses. *Management Science*, 68(10):7497–7517, 2022. doi:[10.1287/mnsc.2021.4275](https://doi.org/10.1287/mnsc.2021.4275).
- J. Schmidt and S. Irnich. New neighborhoods and an iterated local search algorithm for the generalized traveling salesman problem. *EURO Journal on Computational Optimization*, 10:100029, 2022. doi:[10.1016/j.ejco.2022.100029](https://doi.org/10.1016/j.ejco.2022.100029).
- K. N. Singh and D. L. van Oudheusden. A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 97(3):571–579, 1997. doi:[10.1016/S0377-2217\(96\)00313-X](https://doi.org/10.1016/S0377-2217(96)00313-X).
- Y. Su, X. Zhu, J. Yuan, K. L. Teo, M. Li, and C. Li. An extensible multi-block layout warehouse routing optimization model. *European Journal of Operational Research*, 305(1):222–239, 2023. doi:[10.1016/j.ejor.2022.05.045](https://doi.org/10.1016/j.ejor.2022.05.045).
- R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975. doi:[10.1145/321879.321884](https://doi.org/10.1145/321879.321884).
- C. Theys, O. Bräysy, W. Dullaert, and B. Raa. Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763, 2010. doi:[10.1016/j.ejor.2009.01.036](https://doi.org/10.1016/j.ejor.2009.01.036).
- J. Tompkins, J. White, Y. Bozer, E. Frazelle, and J. Tanchoco. *Facilities Planning*. John Wiley & Sons, Hoboken, NJ, 2003.
- C. A. Valle, J. E. Beasley, and A. S. da Cunha. Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, 262(3):817–834, 2017. doi:[10.1016/j.ejor.2017.03.069](https://doi.org/10.1016/j.ejor.2017.03.069).
- T. van Gils, K. Ramaekers, A. Caris, and R. B. de Koster. 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, 2018. doi:[10.1016/j.ejor.2017.09.002](https://doi.org/10.1016/j.ejor.2017.09.002).
- T. S. Vaughan. The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897, 1999. doi:[10.1080/002075499191580](https://doi.org/10.1080/002075499191580).
- F. Weidinger. Picker routing in rectangular mixed shelves warehouses. *Computers and Operations Research*, 95:139–150, 2018. doi:[10.1016/j.cor.2018.03.012](https://doi.org/10.1016/j.cor.2018.03.012).
- F. Weidinger, N. Boysen, and M. Schneider. Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, 274(2):501–515, 2019. doi:[10.1016/j.ejor.2018.10.021](https://doi.org/10.1016/j.ejor.2018.10.021).
- C. Wildt, F. Weidinger, and N. Boysen. Picker routing in scattered storage warehouses: an evaluation of solution methods based on TSP transformations. *OR Spectrum*, 47:35–66, 2025. doi:[10.1007/s00291-024-00780-0](https://doi.org/10.1007/s00291-024-00780-0).