Exact Solution of the Single Picker Routing Problem with Scattered Storage

Katrin Heßler[b], Stefan Irnich[*,a]

[a]*Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*
[b]*Global Data Strategy & Analytics, Schenker AG, Kruppstraße 4, 45128 Essen, Germany.*

Abstract

We present a new modeling approach for the single picker routing problem with scattered storage (SPRP-SS). The SPRP-SS assumes that an article is, in general, stored at more than one pick position. The task is then the simultaneous selection of pick positions for requested articles and the determination of a minimum-length picker tour collecting the articles. It is a classical result of Ratliff and Rosenthal that, for given pick positions, an optimal picker tour is a shortest path in the state space of a dynamic program with a linear number of states and transitions. We extend the state space of Ratliff and Rosenthal so that every feasible picker tour is still a path. Furthermore, the additional requirement to make consistent selections and grouping decisions can be modeled as additional constraints in shortest-path problems. We propose to solve these problems with a MIP solver. We will explain why this approach is not only convenient and elegant but also generic: it covers optimal solutions that use heuristic routing policies for the picker tours, can be applied for different warehouse layouts (we present additional results for a two-block parallel-aisle warehouse), and can incorporate further extensions. Computational experiments with a direct MIP solver-based approach for the SPRP-SS show that the new modeling approach outperforms the available exact algorithms.

Key words: warehousing; picker routing; scattered storage

1. Introduction

Warehouse activities include receiving, storing, picking, packing, and shipping operations (Gu *et al.*, 2007). Excellent surveys introduce warehouse operations planning including storage assignment, warehouse layout planning, zoning, routing, and batching (van Gils *et al.*, 2018; Boysen *et al.*, 2019). In this work, we address picking operations in manual (non-automated) warehouses where pickers travel through the warehouse in order to collect articles from the storage locations (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 (articles) 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: Often cited is the study of Tompkins *et al.* (2003) saying that 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. 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.

In its pure form, the *single picker routing problem* (SPRP) seeks a minimum-length picker tour given the warehouse layout and the pick positions from where articles must be collected. The SPRP can be considered

[*]Corresponding author.
Email addresses: `khessler@uni-mainz.de` (Katrin Heßler), `irnich@uni-mainz.de` (Stefan Irnich)

solved: On the one hand, the seminal work of Ratliff and Rosenthal (1983) assuming a parallel-aisle single-block warehouse shows that a minimum-length picker tour can be computed with dynamic programming in linear time (Heßler 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). The application of heuristic routing policies is well justified in settings where pickers cannot perform all types of optimal tours, which can be complicated, counter-intuitive, and difficult to memorize. Instead, pickers perform tours defined by some simple rules. A little bit more involved is the routing when the policy combined is applied (Roodbergen and de Koster, 2001a). Both exact and heuristic techniques have been extended into many different directions, e.g., to other warehouse layouts (Roodbergen and Koster, 2001; Öztürkoğlu *et al.*, 2012; Çelk and Süral, 2014), non-identical start and end points (Masae *et al.*, 2020a; Löffler *et al.*, 2022), and multiple end depots (de Koster and van der Poort, 1998; Goeke and Schneider, 2021).

When one or several articles are pickable from more than one pick position, the warehouse operates as a *scattered storage* warehouse or *mixed shelves* warehouse. Recent works (Weidinger, 2018; Boysen *et al.*, 2019; Weidinger *et al.*, 2019) stress that scattered storage is predominant in modern e-commerce warehouses of companies like Amazon or Zalando. The main advantage of this 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). The *SPRP with scattered storage* (SPRP-SS) is an integrated operational planning problem characterized by two levels of decisions. The picker routing constitutes the lower-level decision, i.e., the lengths of different picker tours must be computed to evaluate higher-level decisions. The higher-level decision is, for each requested article, the selection of one or several storage positions from where a sufficient number of this article can be collected. If the selection has been made, the resulting picker routing problem is the SPRP. However, both levels are interdependent and the SPRP-SS is known to be NP-hard (Weidinger, 2018), even if optimal routing is replaced by one of the above simple heuristic routing policies (Korbacher *et al.*, 2023).

1.1. Contributions

The focus of our work is on algorithmic improvements for exactly solving the SPRP-SS. The effective solution algorithm we propose relies on the following underlying modeling approach: Every feasible picker tour is a path in the state space of the dynamic-programming approach of Ratliff and Rosenthal (1983), and vice versa. The underlying assumption is that all pick positions are known and given. However, as the picker routing problem is a subproblem of the integrated operational planning problems, the assembly of favorable pick lists creates a new situation in which the *selection* of orders or pick positions becomes essential. Our leading idea is to extend the state space of Ratliff and Rosenthal so that the selection aspect is fully modeled. In the extended state space, every feasible picker tour is still a path. The requirement to make consistent selections can be modeled as additional constraints in the shortest-path problem. We show that this problem can be solved well as binary programs with the help of established *mixed-integer (linear) programming* (MIP) solvers.

The contributions of the paper at hand can be summarized as follows:

- Based on the idea of extending the state space of Ratliff and Rosenthal, we present new binary formulations for the SPRP-SS. These are standard network-flow formulations of the origin-to-destination shortest-path problem (Ahuja *et al.*, 1993) complemented with additional constraints. The formulations are generic in the sense that they apply also to different warehouse layouts and other above-mentioned extensions, since dynamic-programming approaches are known for their solution.
- For the SPRP-SS, the direct MIP-based solution approach outperforms the very recent approach of Goeke and Schneider (2021), which uses another MIP formulation to be solved with a MIP solver, too. Note that their model is less general and restricted to the single-block parallel-aisle warehouse layout.
- For the two-block parallel-aisle warehouse layout, we compare our MIP-based solution approach with a straightforward exact algorithm for the *generalized traveling salesman problem* (GTSP Fischetti *et al.*, 2002). The latter approach is applicable only in the unit-demand case, i.e., when one unit of every article is requested. In this case, it suffices to select a single picking position for each article. The same is true if every picking position holds a sufficiently large supply of the respective article.

1.2. Structure

The paper is structured as follows. In Section 2, we review the dynamic-programming solution method of Ratliff and Rosenthal (1983) and present the new model for the SPRP-SS using an extended state space. Computational results for the SPRP-SS are then reported in Section 3. Section 4 draws final conclusions and discusses future research directions.

2. Single Picker Routing Problems

The basic SPRP assumes that a set P of pick positions in the warehouse is given. The task is to find a minimum length picker tour that starts and ends at the given depot (I/O point) 0 and traverses all positions P. Clearly, this problem can be transformed into a *traveling salesman problem* (TSP) over the vertex set $P \cup \{0\}$, where the distance d_{ij} between a pair of vertices $i, j \in P \cup \{0\}$ is calculated according to the warehouse geometry. Such an approach is very generic because it allows arbitrary warehouse layouts (Theys *et al.*, 2010). On the downside, however, a general TSP algorithm does not consider the specific warehousing situation and is typically not efficient as the TSP is an NP-hard problem (Gutin and Punnen, 2002).

2.1. The State Space of Ratliff and Rosenthal's Dynamic Program

More efficient solution approaches can exploit the warehouse layout. Starting with the seminal paper of Ratliff and Rosenthal (1983) for a standard single-block parallel-aisle warehouse, picker tours have be constructed in an aisle-by-aisle fashion giving rise to a *dynamic programming* (DP) formulation.

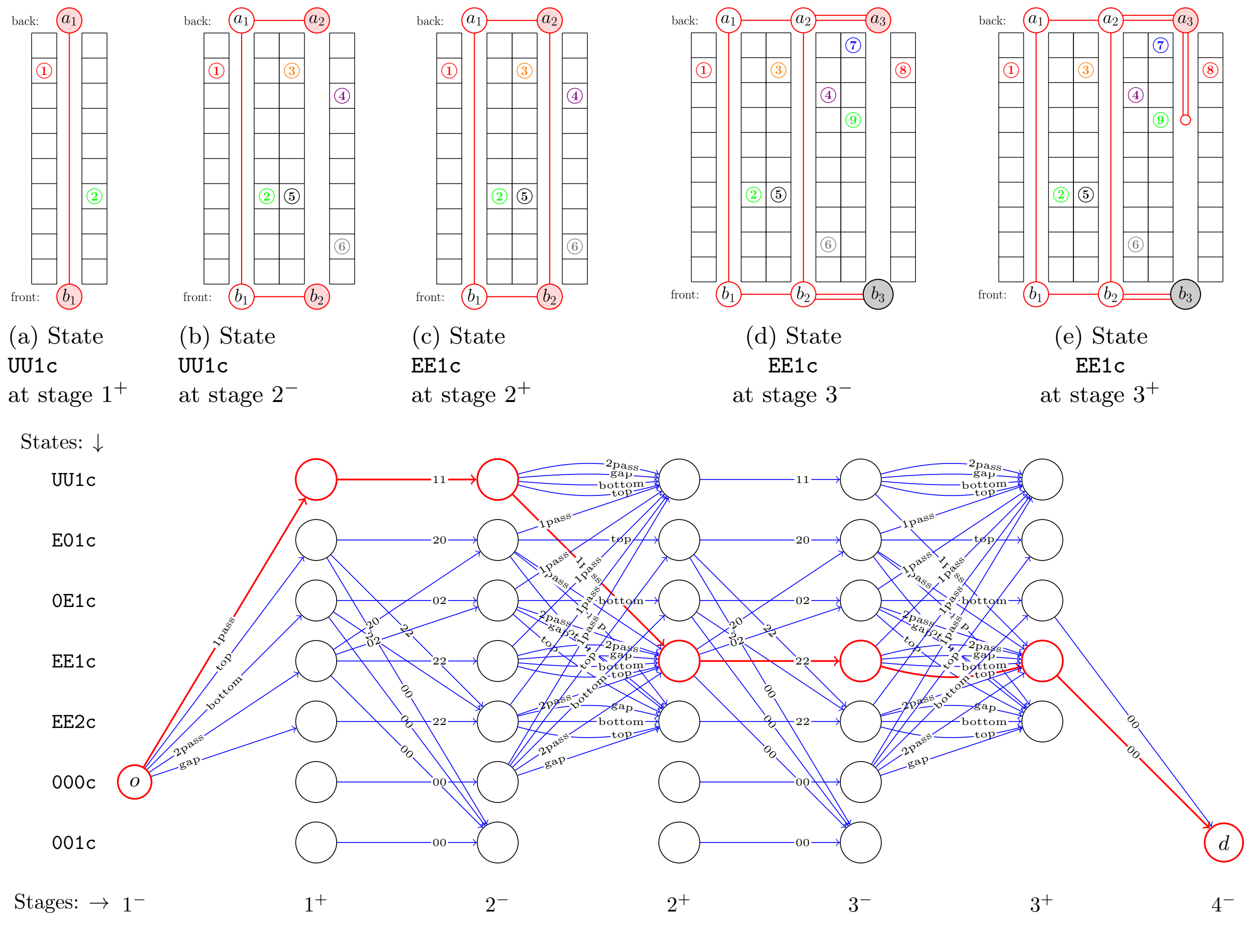
We briefly summarize the fundamental assumptions and ideas because they are essential for the understanding of our new solution algorithm. In a single-block parallel-aisle rectangular warehouse, the *stock keeping units* (SKUs) are stored in racks along both sides of several parallel (picking) aisles. (The terms *SKUs* and *articles* refer to the same objects, but we use article for what is requested by the customers, and SKU for the objects stored in the warehouse.) Cross-aisles end the back and front of each aisle; they are storage-free. The same number of equidistant cells divides each of the picking aisles into pick positions. Picking from the right-hand side, from the left-hand side, or from both sides of a pick position is considered identical when computing picker tour distances. Hence, multiple picking requests with different SKUs from the identical pick position can be modeled as a single aggregated picking request. In the basic SPRP, the pick list boils down to a set of required pick positions that the picker needs to visit in his/her picker tour. The tour starts and ends at the given I/O point or transfer point, called *depot* in the following.

Figure 1 visualizes the construction of the optimal picker tour. The dynamic program models states and transitions over so-called *partial tour subgraphs* (PTSs) which are subgraphs of an undirected graph with vertices a_j and b_j located at the back and front of each aisle $j \in J$, respectively, where $J = \{1, 2, \dots, m\}$ denotes the aisle set. One of these vertices also represents the depot. Moreover, additional vertices of the different PTSs are placed at those pick positions where the picker makes a U-turn. The states of the dynamic program represent the vertex parities (`0`, `U`, or `E` for not reached, odd (=$\underline{\text{u}}$neven) degree, and $\underline{\text{e}}$ven degree, respectively) of the vertices a_j and b_j. The PTSs comprise those parts of the picker tour that belong to the aisles 1 to j, either before the traversal of aisle j is included (we denote this stage as j^-) or after its inclusion (stage j^+). Moreover, the states also indicate the number of connected components of the PTS: The two last symbols `0c`, `1c`, and `2c` indicate the empty subgraph, a subgraph with one, or with two connected components, respectively. Ratliff and Rosenthal (1983) have shown that only seven states are possible for optimal picker tours, namely

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

States are connected in the state graph (V, E) of the DP with two types of (directed) edges: An edge from stage j^+ to $(j+1)^-$ determines a *cross-aisle action* describing the transition from aisle j to the next aisle $j+1$, i.e.,

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



(a) State UU1c at stage 1^+

(b) State UU1c at stage 2^-

(c) State EE1c at stage 2^+

(d) State EE1c at stage 3^-

(e) State EE1c at stage 3^+

(f) State space (V, E)

Figure 1: Example of a warehouse with $m = 3$ aisles, $C = 10$ cells per aisle, and nine articles to be collected, i.e., SKUs $S = \{1, 2, \dots, 9\}$; (1a)–(1e) picker tour subgraphs (PTSs) of the optimal picker tour; (1f) state space of the dynamic program and optimal sequence of states and actions (in red/thick).

where the first (second) digit gives the number of traversals of the back (front) cross-aisle. An edge from stage j^- to j^+ determines the *aisle action*, i.e.,

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

where `1pass` (`2pass`) stands for a single (double) traversal through the aisle (back to front or vice versa), `top` (`bottom`) for a traversal from the back (front) cross-aisle to the lowest (highest) pick position and backward, `gap` for entering the aisle from both sides leaving a maximum length gap in the middle, and `void` for no traversal through the aisle (the latter is only possible in aisles without products).

Each edge $e \in E_j^{cross} \cup E_j^{aisle}$ is naturally associated with a cost describing the length of the part added to the picker tour within an aisle $j \in J$ as well as between consecutive aisles j and $j + 1$. For example, the cost c_e for $e = \texttt{1pass}$ is (proportional to) the height H of the warehouse (front to back). Moreover, the cost c_e for $e = \texttt{gap}$ and aisle $j = 2$ in the example shown in Figure 1 is then $2H - 2v \cdot 4$, where v is the vertical distance between two neighboring cells in an aisle, and the factor 4 results from the largest gap between SKUs 4 and 5 stored in aisle 2. Likewise, the cost c_e for $e = 22$ is $4h$, where h is the horizontal distance between two neighboring aisles.

When solving the dynamic program, the cost 0 is assigned to the initial state $o = \texttt{000c}$ at stage 0^- in a first step. Then, stage-by-stage (i.e., for $0^+, 1^-, 1^+, 2^-, \dots$), the minimum cost is computed for all states $\sigma \in \mathscr{S}$ using the previously computed costs of states of the predecessor stage. Optimal decisions $e(\sigma) \in E_j^{cross} \cup E_j^{aisle}$ are stored for each $j \in J$. As a result, the cost attached to the final state $d = \texttt{001c}$ at stage $(m+1)^-$ is the length of a minimum-length picker tour. Backwards following the optimal decisions provides the optimal sequence of states and decisions. In Figure 1, the sequences are $(o = \texttt{000c}, \texttt{UU1c}, \texttt{UU1c}, \texttt{EE1c}, \texttt{EE1c}, \texttt{EE1c}, \texttt{001c} = d)$ and $(\texttt{1pass}, \texttt{11}, \texttt{1pass}, \texttt{22}, \texttt{top}, \texttt{00})$ (highlighted in red/bold). A complexity analysis shows that the DP can be built-up and solved in $\mathcal{O}(m+n)$ space and time, when n is the number of articles to collect (Ratliff and Rosenthal, 1983; Heßler and Irnich, 2022).

2.2. Scattered Storage

Already Daniels *et al.* (1998) mentioned that the SPRP-SS emerges in two versions: In the so-called *unit-demand case*, the requested number q_s of items is $q_s = 1$ for each article $s \in S$, where S denotes the set of different articles to be collected. The unit-demand case is equivalent to the situation that each pick position $p \in P$ holds a sufficient supply b_{sp} to satisfy the given demand q_s of article $s \in S$, i.e., $b_{sp} \geq q_s$ for all $s \in S$. Daniels *et al.* note that the resulting SPRP-SS is a special type of the *generalized traveling salesman problem* (GTSP, Fischetti *et al.*, 2002). Indeed, the sets $P_s = \{(s,p) \in S \times P : b_{sp} = 1\}$ for $s \in S$ are the clusters. The start and end point 0 of the picker tour at the depot can be included as an extra singleton cluster. The tasks of the GTSP is then to find a cost-minimal subtour that visits exactly one vertex from each cluster.

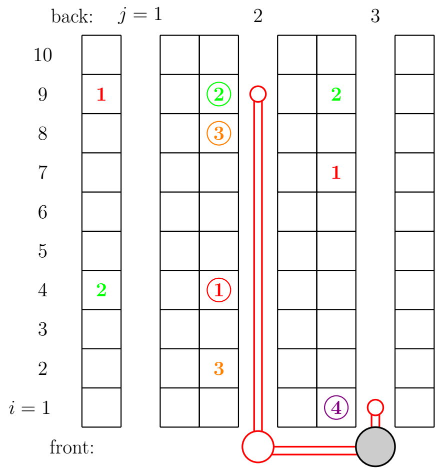
The *general-demand case* is that several items of the same article can or must be retrieved from different pick positions, which can only occur for $q_s > 1$ for some $s \in S$. Daniels *et al.* proposed a TSP-type formulation for the general-demand case of the SPRP-SS as well as a tabu search metaheuristic. 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. Moreover, he showed that, for a single-block parallel-aisle warehouse, the determination of a minimum-length picker tour is NP-hard. This remains true when optimal/exact picker tours are replaced by tours constructed with rule-based routing policies such as traversal, midpoint, largest gap, return, and composite (Korbacher *et al.*, 2023) (this is a companion paper to the paper at hand; the former focusses on SPRP-SS with different routing and storage policies and uses the techniques developed in this work for policy evaluations). Note that routing heuristics are often used in practice (Hall, 1993; Petersen, 1997).

For exact solution approaches and optimal picker routing, Weidinger (2018) 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.* complemented with MTZ-based subtour-elimination constraints (Miller *et al.*, 1960). Later, Goeke and Schneider (2021) presented an effective model (when solved with a MIP solver). This formulation of the SPRP-SS is tailored to the single-block parallel-aisle warehouse layout. We briefly summarize the model of Goeke and Schneider because we will compare their model (denoted as the GS-model in the following) with our new model presented below. The GS-model has some similarities with models known from arc routing (Corberán and Laporte, 2014). Instead of computing a sequence of consecutively visited positions, the solution of the GS model provides an Eulerian graph (connected and even), for which a picker tour can be computed afterwards (Eulerian graph first–routing second). Note that also the dynamic program of Ratliff and Rosenthal provides an Eulerian graph only. The main difference between transitions in the state space of Ratliff and Rosenthal and variables in the GS-model is that decisions in the GS-model are finer than decisions on consecutive PTSs. For example, the DP transition decides simultaneously on the traversal of the top and bottom cross-aisle, while two decisions are made in the GS-model, one for the top and one for the bottom cross-aisle. In the GS-model, the finer decision variables are then coupled by additional constraints ensuring consistency regarding even vertex degrees and connectivity.

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.

Overall, for the exact solution of the SPRP-SS with general demand, the only two evaluated solution approaches are MIP solver-based (Weidinger, 2018; Goeke and Schneider, 2021). It should also be noted that the GS-model approach outperforms the previous approach of Weidinger (2018) by at least one order of magnitude regarding MIP solver computation times and the size of warehouses that can be treated.

Formalizing the SPRP-SS is straightforward: Recall that S denotes the set of SKUs with demands q_s and P_s the set of positions from where b_{sp} units of article $s \in S$ can be collected. If the picker tour visits a subset $P' \subseteq P$ of all pick positions, the demand is fulfilled whenever $\sum_{p \in P' \cap P_s} b_{sp} \geq q_s$ holds for all $s \in S$. In this case the picker tour is *feasible* for the SPRP-SS.



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

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

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

Figure 2: SPRP with scattered storage. The pick list contains four SKUs $S=\{1,2,3,4\}$ (unit demand).

Figure 2a visualizes an instance of the SPRP-SS: There are four SKUs $S=\{1,2,3,4\}$ to be collected. We assume unit demands, i.e., $q_1=\cdots=q_4=1$. Supplies are placed as shown in the figure, i.e., the first three SKUs are stored at two or three positions each, while SKU 4 has a unique position in aisle 3. The depicted tour is feasible as it can collect all four SKUs.

2.3. Extended State Space for Scattered Storage

We describe now the extended state space for a standard warehouse with parallel aisles and a single block layout. The number of stages and the states within each stage remain identical to the state space of Ratliff and Rosenthal. The vertices of the state graph are denoted by V, i.e., V is the set of all states $\mathscr{S}$ duplicated for all stages $1^-, 1^+, 2^-, 2^+, \ldots, m^-, m^+, (m+1)^-$, see Figure 1f. Moreover, cross-aisle actions E_j^{cross} (connecting stages j^+ and $(j+1)^-$) remain identical. However, additional aisle actions have to be added.

For the SPRP-SS depicted in Figure 2a, the additional aisle actions are listed in Figure 2b. In aisle $j=1$, it is possible to pick only one SKU and also to completely skip this aisle because the SKUs 1 and 2 are also available in other aisles. In aisle $j=2$, SKU 3 must be collected, either from position $i=2$ or $i=8$ making `void` and `top`(9) impossible (the latter is the traversal from the top with a U-turn in cell $i=9$). Moreover, both `gap`(2,8) and `gap`(2,9) can collect SKUs 2 and 3 but not 1. For the last aisle $j=3$, SKU 4 must be collected so that `void`, `top`(7), and `top`(9) are not allowed. Note that all traversals `gap`(i,k) can be disregarded if i and k are neighboring positions with a non-maximal gap. As in the dynamic program of Ratliff and Rosenthal, these traversals are dominated by one with maximum gap. Overall, the set of aisle traversals becomes aisle dependent. Hence, we denote by E_j^{aisle} the resulting set of arcs connecting stages j^- and j^+ for all $j \in J$.

The aisle traversal set can be reduced by dominance considerations. In the example, $\texttt{gap}(2{,}8)$ is dominated by $\texttt{gap}(2{,}9)$ because of its higher cost. Note that this is only true in the unit-demand case, because otherwise $\texttt{gap}(2{,}8)$ can provide more items of SKU 3. Likewise, $\texttt{bottom}(8)$ is dominated by $\texttt{bottom}(4)$.

For scattered storage with arbitrary demand, any aisle traversal $e \in E_j^{aisle}$ that leaves out the positions $P'_j \subset P$ in aisle j is feasible if and only if

$$\sum_{(s,p)\in P_s:\, p\notin P'_j} b_{sp} \geq q_s \qquad \forall s \in S. \tag{1}$$

The term on the left-hand side is the total supply of the not left-out positions (it must be possible to construct a feasible solution with the reached positions and all other aisle traversals referring to different aisles).

For convenience, we define P_e as the set of positions covered by an aisle traversal $e \in E_j^{aisle}$ in aisle $j \in J$. Then, $b_{se} = \sum_{p\in P_e} b_{sp}$ is the quantity of SKU $s \in S$ that can be collected when traversing the aisle via e. We denote the edges with a non-negative supply of SKU $s \in S$ by E_s. Cross-aisle traversals have zero supply $b_{se} = 0$ for all $e \in E_j^{cross}$ and $j \in J$.

For the following analysis of the size of the network and the model, it is important to properly distinguish between the set of relevant pick positions and all pairs (s,p) with a positive supply $b_{sp} > 0$. Note that several articles $s \in S$ might be available at the same pick position p, since, e.g., different SKUs might be placed to the left- and right-hand-side of one position. Moreover, when stored in shelves, different SKUs can even be placed vertically over one another at the same side. Summarizing, the number $n = |\bigcup_{s\in S} P_s| = |\{p \in P : \exists s \in S \text{ with } b_{sp} > 0|$ of relevant positions can be smaller than the number $|\{(s,p) \in S \times P : b_{sp} > 0\}|$ of pairs with a positive supply b_{sp}.

For counting the number of edges, note first that the original state space of Ratliff and Rosenthal has only $\mathcal{O}(m)$ edges. Note also that the number of additional aisle transitions in the extended state space is dominated by those of type $\texttt{gap}(h,i)$, where h and i are two different pick positions within an aisle. The worst case is that the majority of the pick positions concentrates in one or very few aisles so that there can be $\mathcal{O}(n^2)$ edges for aisle transitions of type $\texttt{gap}(h,i)$. Therefore, the total number of edges is bounded by $\mathcal{O}(m+n^2)$.

2.4. Network-Flow Formulation

We first analyze the situation with a unique aisle for an SKU before we formalize the general case. Whenever every SKU $s \in S$ is available in only one (unique) aisle j (e.g., SKU 3 in aisles 2 and SKU 4 in aisle 3, see Figure 2), the feasibility regarding demand fulfillment is completely ensured by condition (1). Every $e \in E_j^{aisle}$ already ensures that the demand q_s can be collected in aisle j. Even more, a shortest *o*-*d*-path over the extended state graph provides an optimal picker tour. No constraints in addition to flow-conservation constraints are mandatory. Hence, defining variables $x_e \geq 0$ for all $E = \bigcup_{j\in J}(E_j^{aisle} \cup E_j^{cross})$, directly leads to the standard *o*-*d*-shortest-path model for the SPRP-SS with unique aisles per SKU:

$$\min \sum_{e\in E} c_e x_e \tag{2a}$$

$$\text{subject to} \quad \sum_{e\in\delta^+(\sigma)} x_e - \sum_{e\in\delta^-(\sigma)} x_e = \begin{cases} +1, & \text{if } \sigma = o \\ -1, & \text{if } \sigma = d \\ 0, & \text{otherwise} \end{cases} \qquad \forall \sigma \in V \tag{2b}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{2c}$$

The length of the resulting picker tour is minimized by (2a) with appropriately defined costs c_e for all edges $e \in E$. Flow conservation is ensured via (2b), where, for any state $\sigma \in V$, $\delta^+(\sigma)$ and $\delta^-(\sigma)$ denote the set of arcs leaving and entering state σ, respectively. The domain of the flow variables is given by (2c). Note that the demand fulfillment is ensured by the definition of E, since all SKUs are only available in one unique aisle j.

The flow-conservation constraints (2b) can be rewritten in a more compact form as $\mathcal{N}\mathbf{x} = \mathbf{u}_o - \mathbf{u}_d$ using the incidence matrix $\mathcal{N}$ of the state graph (V, E), the vector $\mathbf{x} = (x_e)_{e\in E}$ of the x-variables, and unit vectors $\mathbf{u}_o$ and $\mathbf{u}_d \in \{0,1\}^V$. The general SPRP-SS model for a not necessarily unique aisle per SKU is:

$$\min \sum_{e\in E} c_e x_e \tag{3a}$$

$$\text{subject to} \quad \mathcal{N}\mathbf{x} = \mathbf{u}_o - \mathbf{u}_d \tag{3b}$$

$$\sum_{e\in E_s} b_{se} x_e \geq q_s \qquad \forall s \in S \tag{3c}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{3d}$$

The difference between models (2) and (3) is that for the general case of not necessarily unique aisles, the flow variables must be forced to binary values due to the additional demand-covering constraints (3c). The latter constraints can be strengthened by replacing b_{se} with $\min\{q_s, b_{se}\}$ for all $s \in S$. In particular, this leads to a pure $\{-1,0,1\}$-coefficient matrix of formulation (3) in the unit-demand case. Moreover, SKU-covering constraints (3c) are redundant for every SKU s stored in a unique aisle. We assume both refinements in the following.

A straightforward analysis of the size of formulation (3) can be summarized as follows: Let $a = |S|$ denote the *number of different articles*. The number of variables coincides with the number of edges in the extended state space which is bounded by $\mathcal{O}(m+n^2)$. The number of constraints is bounded by $\mathcal{O}(m+a)$, where constraints (3b) contribute the summand m and constraints (3c) 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 SKU-covering constraints (3c) have $\sum_{s\in S}|E_s| = \mathcal{O}(an^2)$ non-zero coefficients.

Clearly, the network-flow model (3) hides complicated details in the definition of the underlying state space over which the flow-conservation constraints are defined. However, the model is generic and extensible, as discussed in the following section.

2.5. *Possible Extensions of the Modelling and Solution Approach*

The combined approach of
- building the state space for the SPRP,
- extending the state space by additional actions for modeling additional options resulting from scattered storage, and
- solving the o-d-shortest-path problem over the state graph with additional demand-covering constraints using a MIP solver

is not limited to single-block parallel-aisle warehouses. Indeed, for parallel-aisle warehouses with one or several 'middle' cross-aisles, Roodbergen and de Koster (2001a,b) have developed refined DP approaches. Instead of deciding traversals per aisle, the decision process is broken down into finer decisions per block, cross-aisle, and aisle. For example, with one middle cross-aisle there are three stages per aisle in the DP (see Section 3.4). More generally, a fixed number of middle cross-aisles can be handled similarly so that the computational complexity of the DP remains linear in $m + n$ although the number of states grows considerably due to the numerous cases of connectivity (Pansart *et al.*, 2018). There exist several use cases where starting and ending points of the picker tour do not coincide, for which Masae *et al.* (2020a); Löffler *et al.* (2022) adapt the DP approach of Roodbergen and de Koster (2001b). DP-based optimal picker routing has also been suggested for different warehouse layouts such as the fishbone layout (the warehouse has two diagonal cross-aisles from where picking aisles extend horizontally and vertically, see Çelk and Süral, 2014). The same authors proposed a graph transformation from the fishbone layout as well as the flying-V warehouse layout (the warehouse has parallel picking aisles perpendicular to the front and back cross-aisles, see Gue and Meller, 2009; Çelk and Süral, 2014) to the rectangular layout with two blocks as considered by Roodbergen and de Koster (2001b). Other non-conventional layouts that have been introduced in the literature are U-shaped (see Glock and Grosse, 2012; Henn *et al.*, 2013) and chevron layout (see Öztürkoğlu *et al.*, 2012). Recently, also a discrete cross aisle warehouse design has been developed in the literature (see

Ömer Öztürkoğlu and Hoser, 2019). What unifies all the DP models is that *o*-*d*-paths in the respective state space represent feasible picker tours and vice versa.

3. Computational Results

In this section, we first specify details of the computational setup and implementation. Afterwards, SPRP-SS benchmark instances are presented and results for the SPRP-SS in a single-block warehouse (Section 3.3) and a two-block warehouse (Section 3.4) are analyzed and discussed. The section closes with a cost comparison between single-block and two-block parallel-aisle warehouse layouts (Section 3.5).

3.1. Details of the Implementation

All algorithms are implemented in `C++` using the callable library of `CPLEX` 20.1.0 and compiled into 64-bit single-thread release code with Microsoft Visual Studio 2015. The computational study is performed 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. For the MIP solver, `CPLEX`'s default values of all parameters are kept except for the time limit and setting the number of available threads to one.

For the experiments with the `concorde` TSP solver, we use a Linux/Ubuntu 22.04.2 LTS installation inside the *Windows Subsystem for Linux* (WSL 2) installed on the same PC. The solver itself is compiled and installed using Alberto Santini's fork (Santini, 2022) of `concorde` based on the most recent (2003) version 03.12.19 (Applegate *et al.*, 2003).

3.2. Benchmark Instances

The recent articles of Weidinger (2018) and Goeke and Schneider (2021) describe how different instances of the SPRP-SS can be generated (we have summarized the procedure in Section A of the Appendix). The layout is that of a single-block parallel-aisle warehouse with distance 3 between neighboring aisles and distance 1 between neighboring cells/positions. An instance is characterized by a combination (m, C, n, α) of the following four values: number m of aisles, number C of cells per aisle, number n of different articles to be collected, and the scatter factor α. The scatter factor α indicates how often on average identical articles occur at different pick positions. Note that $\alpha = 1$ is the case of the classical SPRP without scattered storage.

Due to the ABC class-based instance generation process (see Section A of the Appendix), a pick list tends to have more order lines with A-articles than with B- and C-articles. Moreover, A-articles tend to have more pick positions compared to B- and C-articles. As a result, a typical pick list with n order lines has more than αn corresponding pick positions (the reader might have expected exactly αn positions). For example, instances with $\alpha = 10$ and $n = 30$ have on average 656.9 pick positions.

Goeke and Schneider (2021) generated SPRP-SS instances with the goal to perform a kind of stress test for their new formulation. The largest instances had $m = 100$ aisles (in practice, this huge number of aisles is probably not served together with a single picker). Moreover, the largest scatter factor was chosen as $\alpha = 50$. For a (large) warehouse with $m = 25$ aisles, the result is that each article occurs on average twice in each aisle.

We decided to generate instances closer to the real-world application with a different set of combinations (m, C, n, α):

- $m \in \{5, 10, 25, 50\}$: we test 10 and 50 instead of 100;
- $C \in \{30, 60, 180\}$: as in (Goeke and Schneider, 2021);
- $n \in \{3, 7, 15, 30\}$: as in (Goeke and Schneider, 2021);
- $\alpha \in \{1, 2, 5, 10\}$: we restrict the combinations by requiring $\alpha < m$. This restriction makes the instances more realistic, since otherwise a typical batch can just be collected from one or two aisles close to the depot 0.

This gives 156 feasible combinations (m, C, n, α), and particularly, 108 combinations for $\alpha > 1$. To obtain statistically firm results, we generated 50 instances per combination.

We use the identical procedure to generate SPRP-SS instances for two-block parallel-aisle warehouses by adding a middle cross-aisle to the one-block instances. For the sake of simplicity, two-block instances have the middle cross-aisle exactly in the middle, e.g., between cell 15 and 16 in a warehouse with $C = 30$. Overall, this benchmark set comprises $2 \cdot 156 \cdot 50 + 2 \cdot 108 \cdot 50 = 15{,}600 + 10{,}800 = 26{,}400$ instances (two summands for unit and general demand; factor two for one-block and two-block) available at `https://logistik.bwl.uni-mainz.de/research/benchmarks/`.

3.3. Computational Results for the Single-Block Parallel-Aisle Warehouse Layout

The only competitive exact algorithm for the SPRP-SS is, up to date, the MIP solver-based solution of the model presented by Goeke and Schneider (2021). This approach (denoted by GS for the sake of brevity) is now compared with the MIP solver-based solution of our network-flow (NF) formulation (3). We implemented and tested the two formulations, making sure that both GS and NF find identical optimal objective values for all instances.

We start with the experimental comparison of one special case. For the scatter factor $\alpha = 1$, the resulting problem reduces to a basic (non-scattered) SPRP, and the unit-demand and general-demand cases do not need to be distinguished, since they lead to identical instances (recall that we replaced b_{se} by $\min\{q_s, b_{se}\}$ for all $s \in S$ and $e \in E$, see Section 2.2). Moreover, in this special case, the original DP algorithm of Ratliff and Rosenthal constitutes a third solution approach. For each group of instances with identical parameters (m, C, n), let $t_{\text{DP}}, t_{\text{NF}}$, and t_{GS} denote the average computation time (in milliseconds) for the solution of the respective approach. Table 1 displays average times and $t_{\text{GS}}/t_{\text{NF}}$. The latter value $t_{\text{GS}}/t_{\text{NF}}$ is the geometric mean of the ratios of GS and NF solution times (over the 50 instances of each group). It is an estimate for the speedup one can expect when replacing GS by NF. In particular, all values $t_{\text{GS}}/t_{\text{NF}} > 1$ indicate that NF is faster than GS.

It is not surprising that DP is always the fastest algorithm, where all average computation times stay below 2.5 milliseconds. Moreover, the speedup values $t_{\text{GS}}/t_{\text{NF}} > 1$ are between 5.5 and 23.1, with the trend that speedups are increasing for larger warehouses (more aisles and more cells per aisle), and the reverse trend can be observed for longer pick lists (increasing n-values). The general superiority of NF over GS can be explained by the fact that the GS model does not have the integrality property, while NF—as a pure network-flow model—is having it. As a result, the MIP solver needs to not only solve the linear relaxation of the GS model, but it must close the integrality gap by cutting and branching, which takes more time than the solution of the LP providing a solution of NF directly.

Table 1: Comparison of DP, NF, and GS with computation times (in milliseconds) and speedup factor comparing GS and NF for the basic SPRP (non-scattered) and a one-block parallel-aisle warehouse.

Warehouse dimension	Number of articles in pick list															
	$n = 3$				$n = 7$				$n = 15$				$n = 30$			
(m, C)	t_{DP}	t_{NF}	t_{GS}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{GS}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{GS}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{GS}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$
(5.30)	0.2	0.7	5.9	8.4	0.2	0.8	5.3	6.5	0.2	1.1	6.9	6.6	0.2	0.9	5.2	5.7
(5.60)	0.2	0.7	5.5	8.6	0.2	0.8	6.2	7.8	0.2	0.9	5.3	5.7	0.2	0.9	5.5	5.9
(5.180)	0.1	0.6	5.1	8.8	0.2	0.8	5.2	6.7	0.2	0.9	5.2	5.9	0.2	0.9	5.5	6.0
(10.30)	0.3	1.0	9.0	8.5	0.3	1.4	11.2	8.0	0.4	1.7	11.0	6.6	0.4	1.9	13.8	7.4
(10.60)	0.3	1.1	10.0	9.6	0.3	1.4	10.1	7.1	0.3	1.8	11.6	6.6	0.4	1.9	13.8	7.5
(10.180)	0.3	1.0	8.9	8.3	0.3	1.4	13.0	9.5	0.3	1.7	12.1	7.2	0.4	1.8	12.9	7.0
(25.30)	0.8	2.4	31.5	13.2	0.9	3.1	32.1	10.5	1.0	3.9	29.1	7.6	1.0	4.8	33.2	7.0
(25.60)	0.8	2.4	50.7	20.2	0.7	3.1	42.6	13.5	0.8	3.8	36.6	9.5	0.9	4.7	37.9	8.1
(25.180)	0.7	2.4	40.4	16.8	0.8	3.0	45.5	15.3	0.9	3.9	36.8	9.4	1.0	4.9	39.8	8.3
(50.30)	1.8	5.6	75.0	11.9	2.0	6.5	76.6	11.9	1.8	7.8	86.1	10.9	2.1	9.6	64.0	6.7
(50.60)	1.7	5.6	119.7	21.3	1.8	6.7	148.1	22.2	2.1	7.8	104.2	13.1	2.1	9.7	95.1	9.9
(50.180)	1.8	5.8	116.5	18.7	1.8	6.5	151.0	23.1	1.9	7.7	126.6	16.3	2.5	10.9	215.8	19.6

For the true scattered case ($\alpha > 1$), Table 2 and Table 7 in the Appendix show aggregated results for unit demand as well as general demand, respectively. The comparison is only between GS and NF, since the direct DP approach is no longer applicable. The overall average speedup of NF over GS is 3.6. For both NF and GS, the trend is that instances with a higher number n of articles in the pick list, a larger warehouse (more aisles and cells), and a larger scatter factor α are more difficult to solve optimally, i.e., their solution by the MIP solver consumes more time. One can however expect computation times below one second for (realistically sized) warehouses with not more than 25 aisles. Even the most difficult setting for $(m, C, n, \alpha) = (50, 180, 30, 10)$ has average computation times below 10 seconds.

Table 2: Comparison of NF and GS with computation times (in milliseconds) and speedup factor comparing GS and NF for the SPRP-SS, a one-block parallel-aisle warehouse, and a sufficient number of items (unit demand; $b_{se} = q_s = 1$).

Scatter factor	Warehouse dimension	Number of articles in pick list											
		$n=3$			$n=7$			$n=15$			$n=30$		
	(m,C)	t_{NF}	t_{GS}	$\frac{t_{GS}}{t_{NF}}$	t_{NF}	t_{GS}	$\frac{t_{GS}}{t_{NF}}$	t_{NF}	t_{GS}	$\frac{t_{GS}}{t_{NF}}$	t_{NF}	t_{GS}	$\frac{t_{GS}}{t_{NF}}$
$\alpha=2$	$(5,30)$	5.5	18.9	3.7	7.2	18.7	2.8	14.3	16.0	1.2	23.5	21.1	1.0
	$(5,60)$	5.1	20.3	4.6	8.2	27.9	3.2	13.7	22.7	1.7	29.9	24.9	1.0
	$(5,180)$	5.0	19.3	4.7	8.5	43.8	5.0	16.7	29.8	1.8	36.6	24.1	0.7
	$(10,30)$	9.4	43.5	5.3	14.8	50.4	4.3	22.5	51.2	2.4	41.2	43.8	1.1
	$(10,60)$	9.2	63.7	7.3	16.6	73.0	5.3	23.1	55.8	2.8	44.2	53.9	1.3
	$(10,180)$	8.6	67.8	8.8	14.1	95.2	7.8	22.0	51.5	2.6	49.2	47.0	1.1
	$(25,30)$	16.5	104.7	6.9	23.4	128.4	6.0	31.3	110.5	3.8	65.7	128.8	2.4
	$(25,60)$	15.2	147.1	10.4	22.4	141.0	7.1	26.8	114.2	4.5	47.6	125.3	2.7
	$(25,180)$	16.5	175.5	11.0	23.4	290.0	13.0	33.3	195.4	6.8	52.6	152.9	3.3
	$(50,30)$	24.1	191.6	9.6	28.1	237.1	10.2	39.2	177.6	5.1	53.7	209.5	3.9
	$(50,60)$	24.2	244.2	11.9	29.8	371.5	12.4	38.8	371.7	9.7	49.8	221.3	4.5
	$(50,180)$	31.8	999.4	35.3	37.3	1207.7	39.8	34.8	901.8	27.6	58.7	480.1	8.4
$\alpha=5$	$(10,30)$	13.9	61.9	4.5	43.9	150.8	3.8	89.8	155.7	1.7	154.3	129.8	0.8
	$(10,60)$	18.4	84.4	5.4	44.2	175.4	4.6	124.5	276.0	2.6	274.9	251.8	0.9
	$(10,180)$	15.1	76.7	5.7	37.7	174.8	5.0	131.3	370.5	3.2	419.8	443.8	1.0
	$(25,30)$	39.3	168.2	5.7	109.4	265.5	3.8	174.2	280.1	1.9	483.9	411.1	0.9
	$(25,60)$	44.2	189.8	4.8	138.8	377.4	4.6	228.2	564.3	3.4	623.9	630.8	1.1
	$(25,180)$	37.7	344.6	10.3	144.9	948.1	10.9	296.2	1746.6	7.7	644.5	1039.8	1.9
	$(50,30)$	74.1	230.8	3.8	142.1	397.6	4.5	206.5	482.0	2.8	456.2	749.3	2.1
	$(50,60)$	85.6	467.7	8.6	153.2	910.5	12.0	252.3	1077.5	5.4	634.8	1658.1	3.6
	$(50,180)$	67.0	559.7	10.2	165.5	1876.3	13.9	404.9	6010.8	14.4	645.7	3831.0	5.2
$\alpha=10$	$(25,30)$	184.8	338.7	2.9	648.5	639.9	1.5	1807.9	1275.4	1.1	3008.6	1864.1	0.8
	$(25,60)$	181.2	497.2	3.8	1104.9	1091.5	1.8	3413.3	2751.6	1.1	4925.3	4123.8	0.9
	$(25,180)$	150.8	552.6	5.0	813.2	1783.2	3.4	3388.7	6739.9	3.1	6726.6	12 995.8	2.3
	$(50,30)$	525.5	625.5	1.7	2424.6	1971.9	2.4	2331.4	2675.7	1.7	4989.8	4206.2	1.0
	$(50,60)$	660.5	914.5	2.5	2076.2	2612.3	3.7	3530.8	4787.4	2.2	6173.1	7564.4	1.4
	$(50,180)$	516.6	1300.2	3.4	1465.9	6495.3	8.2	4447.5	15 670.6	4.5	8299.9	36 138.2	4.4

Looking more closely at the case of a sufficient number of items (unit demand, Table 2), NF is superior to GS ($t_{NF} < t_{GS}$) for almost all groups. Some exceptions occur for pick lists of size $n = 30$ and warehouses with $m \leq 25$ aisles. Even in these cases, the factor t_{GS}/t_{NF} is always greater than or equal to 0.7. For the example of $(m, C, n, \alpha) = (5, 180, 30, 2)$ with $t_{GS}/t_{NF} = 0.7$, the 30 articles are on average stored at $112.1 > 60 = 2\cdot 30$ positions in a narrow warehouse with only five aisles but 180 cells per aisle. This is a rather obscure setting. On the other extreme, the highest average speedup of value $t_{GS}/t_{NF} = 39.8$ was obtained for instances for $(m, C, n, \alpha) = (50, 180, 7, 2)$. Indeed, the speedup mainly depends on the warehouse size: the greater m and C, the greater t_{GS}/t_{NF}.

For SPRP-SS instances with general demand (Table 7 in Section B of the Appendix), the trends for what makes instances difficult is the same as in the unit-demand case. However, computation times t_{NF} are longer, e.g., the smallest (largest) times of 5 milliseconds (8.3 seconds) for the unit-demand case rise to 47 milliseconds (20.6 seconds). On average, general demand leads to a 4.7 and 2.8 times slower solution of NF and GS, respectively. The factor t_{GS}/t_{NF} shows similar trends as for unit demand, but the absolute speedup is lower. The highest value amounts to $t_{GS}/t_{NF} = 16.1$, the lowest to 0.5. The 19 cases (of 108) where NF is inferior to GS, i.e., $t_{GS}/t_{NF} < 1$, the pick list is relatively long (three cases for $n = 15$ and 16 cases for $n = 30$).

Finally, we analyze to which extent the computation time vary. To this end, we compute the *coefficient of variation* (CT) of the computation times for each group (50 instances per parameter combination (m, C, n, α)). Over the 156+108 groups, the minimum, average, and maximal CT-values are 0.35, 0.80, and 2.20 for GS compared to 0.10, 0.70, and 1.86 for NF, respectively. Overall, the computation times of NF are more predictable and stable than for GS.

In total, NF outperforms GS with the exception of a few extreme parameter combinations mentioned above.

3.4. Computational Results for the Two-Block Parallel-Aisle Warehouse Layout

Next, we consider the two-block parallel-aisle warehouse layout. For this type of warehouse, the basic (non-scattered, $\alpha = 1$) SRPR can be solved with the DP algorithm of Roodbergen and de Koster (2001b), which is a non-trivial extension of the DP approach of Ratliff and Rosenthal (1983). Section C of the Appendix characterizes and visualizes the state space of the DP, which comprises $3m + 1$ stages (for each aisle, there are aisle actions in the back block, aisle actions in the front block, and cross-aisle action; plus one stage for an artificial destination state) with 25 states each. Although it seems large, the state space (V, E) has a linear number $\mathcal{O}(m)$ of states and edges. With the techniques described in (Heßler and Irnich, 2022), also the DP can be constructed in linear time $\mathcal{O}(m + n)$ including all cost computations, where n is the length of the pick list, i.e., the number of different articles to be collected. Moreover, the non-scattered instances can be solved as TSPs. For this purpose, we use the popular publicly available `concorde` TSP solver (Applegate *et al.*, 2003), which is based on branch-and-cut and integrates a chained Lin-Kernighan TSP heuristic to compute upper bounds.

When considering scattered articles ($\alpha > 1$), the only competitive approach is the solution of a GTSP. For an exact GTSP solution, branch-and-cut algorithms outperform earlier solution attempts with dynamic programming and branch-and-bound. As far as we know, there is no branch-and-cut implementation publicly available. Therefore, we re-implemented the branch-and-cut algorithm of Fischetti *et al.* (2002). Section D of the Appendix briefly summarizes the underlying formulation and sketches similarities and minor differences of our branch-and-cut implementation in comparison to one of Fischetti *et al.* In the below evaluation, we use our GTSP solver for comparison with the new NF formulation (3) solved ‘out of the box’ with the MIP solver `CPLEX`.

Table 3: Comparison of DP, NF, and TSP with computation times (in milliseconds) as well as speedup factor comparing TSP and NF for the basic SPRP (non-scattered) and a two-block parallel-aisle warehouse.

Warehouse dimension	Number of articles in pick list															
	$n=3$				$n=7$				$n=15$				$n=30$			
(m, C)	t_{DP}	t_{NF}	t_{TSP}	$\frac{t_{\text{TSP}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{TSP}	$\frac{t_{\text{TSP}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{TSP}	$\frac{t_{\text{TSP}}}{t_{\text{NF}}}$	t_{DP}	t_{NF}	t_{TSP}	$\frac{t_{\text{TSP}}}{t_{\text{NF}}}$
$(5, 30)$	0.1	2.5	0.5	0.3	0.1	3.8	0.7	0.2	0.2	5.8	5.8	1.1	0.2	8.3	32.4	4.6
$(5, 60)$	0.1	2.6	0.5	0.3	0.1	5.1	0.9	0.2	0.2	5.7	5.9	1.3	0.2	8.4	31.4	4.6
$(5, 180)$	0.1	1.8	0.5	0.3	0.2	3.7	0.7	0.3	0.2	6.7	7.9	1.4	0.2	7.8	28.4	4.1
$(10, 30)$	0.2	4.2	0.4	0.1	0.3	5.4	0.7	0.1	0.4	11.6	5.7	0.6	0.5	15.7	19.9	1.3
$(10, 60)$	0.2	3.4	0.5	0.2	0.3	6.1	0.8	0.2	0.4	12.1	6.8	0.7	0.5	17.2	24.6	1.5
$(10, 180)$	0.2	2.9	0.4	0.2	0.3	6.5	0.7	0.1	0.4	12.2	8.0	0.8	0.5	15.9	32.4	2.2
$(25, 30)$	0.9	6.4	0.4	0.1	1.0	13.7	0.7	0.1	1.2	24.0	7.7	0.3	1.4	36.4	22.4	0.6
$(25, 60)$	0.9	8.1	0.5	0.1	1.1	13.7	0.8	0.1	1.2	24.4	6.5	0.3	1.4	36.3	19.3	0.6
$(25, 180)$	0.9	6.7	0.5	0.1	1.0	12.2	0.8	0.1	1.2	21.9	8.3	0.4	1.4	36.6	31.1	0.9
$(50, 30)$	3.2	16.2	0.5	<0.1	3.6	24.9	0.7	<0.1	3.9	35.8	10.7	0.3	4.2	54.2	33.4	0.6
$(50, 60)$	3.3	18.5	0.4	<0.1	3.8	27.0	0.7	<0.1	4.1	39.7	7.9	0.2	4.3	55.6	20.5	0.4
$(50, 180)$	3.3	14.3	0.5	<0.1	3.6	23.3	0.7	<0.1	4.0	41.2	7.0	0.2	4.2	57.7	27.7	0.5

The following computational analysis for the two-block parallel-aisle warehouse layout follows a similar sequence as in the previous section for the one-block warehouse. For the non-scattered case, i.e., the basic SPRP, we compare the DP of Roodbergen and de Koster (2001b) with the MIP-solver-based solution of the *o*-*d*-shortest path formulation (2) and the TSP solver `concorde`. The results are summarized in Table 3. As for the one-block case, the direct DP-based solution is very fast with average solution times t_{DP} between 0.1 and 4.3 milliseconds per group. For small values of $n \leq 7$, i.e., short pick lists, the TSP solver `concorde` even outperforms the DP regarding solution times. Since the structure of the TSP model is independent of the warehouse layout, the times t_{TSP} are not systematically affected by the parameters m and C but increase with the pick list length n from 0.4 to 33.4 milliseconds in the worst case. Our own NF approach is always outperformed by DP or TSP. However, for the longest pick lists ($n = 30$) and small warehouses with not more than ten aisles, the NF approach outperforms the TSP approach. As this is the classical non-scattered case, these results should just be taken as a sidenote.

For the scattered case, i.e., $\alpha > 1$, we compare our NF formulation (3) solved with the MIP-solver and the GTSP solver. As for the GS formulation analyzed in the previous sections, pre-tests have shown that the MIP solver solution times for the GTSP branch-and-cut implementation can fluctuate substantially. Due to the large number of instances to be solved, we have set a strict time limit of 60,000 milliseconds for the GTSP solver.

The results of the comparison for the two-block SPRP-SS instance with unit demand are summarized in Table 4. The number n of articles to collect has the highest impact:

- For $n = 3$, the GTSP instances have exactly four clusters (with one singleton cluster for the depot) and the GTSP solver is the clear winner. This outcome was highly predictable, because the branch-and-cut approach does not require subtour elimination, which makes it very fast. Note that even a complete enumeration of all feasible GTSP solutions is feasible in this case. These instances are easy to solve.
- For $n = 7$, average computation times t_{NF} and t_{GTSP} are in the same range. For increasing values of m and C, times tend to increase slightly, probably due to larger cost coefficients and objective values, which tend to slow down the MIP solver and so the branch-and-cut. For all groups (m, C, α), the MIP solver with NF is faster than the GTSP solver. Note that the GTSP solver could not solve some instances within the given time limit. Table 5 shows the number of solved and unsolved instances per combination of α and n. As expected, larger α-values lead to larger and more difficult instances.
- For $n = 15$, the picture changes completely and NF becomes the clear winner. Here, about one quarter of the instances with $\alpha = 5$ is solved with the branch-and-cut GTSP solver, while not a single instance with $\alpha = 10$ is solved optimally (see Table 5). The average speedup values $t_{\mathrm{GTSP}}/t_{\mathrm{NF}}$ range from 1.3 to 72.2 counting unsolved GTSP instances with 60 seconds. This means that for $\alpha \geq 5$ the true speedup is even larger.
- For $n = 15$, only instances with a small scatter factor of $\alpha = 2$ are solved exactly with the GTSP solver. Here, the speedups achieved by our NF-based algorithm range from 7.2 to 95.6 (with the same bias as above).

Comparing Tables 2 and 4, we see that the two-block SPRP-SS instances are much harder to solve than the single-block instances. This statement can be supported by comparing the average NF computation times for the 156 combinations of (m, C, n, α). More precisely, we compute the geometric mean GM of the ratios of the corresponding pairs of 2-block NF times and the 1-block NF times in the two tables. Note that the result $GM = 7.63$ is a geometric mean of 156 geometric means. The value $GM = 7.63$ is plausible as explained below:

- The number of states per stage increases by a factor of $25/7 \approx 3.57$ in the two-block DP compared to the one-block DP.
- The number of stages increases by a factor of approximately $3/2 = 1.5$ ($3m + 1$ versus $2m + 1$).
- Aisle actions and cross-aisle actions are not equally represented in the two-block state space. The number of aisle actions (counted as edges in the state space) increases by a factor of approximately $2 \cdot 25/7 = 7.14$, while the number of cross aisle actions increases by a factor of approximately the $25/7 = 3.57$.
- As a result, in formulation (3), the number of x_e-variables (related to edges $e \in E$) increases by a factor between 3.57 and 7.14. In addition, the number of flow-conservation constraints increases by a factor of 7.14. Typically, MIP solution times grow progressively with an increasing number of variables and constraints, which explains why the computed geometric mean of $GM = 7.63$ is larger than the individual factors.

Overall, the new NF-based solution approach is highly competitive for non-trivial instances of the SPRP-SS and two-block parallel-aisle warehouses. For a likely realistic scatter factor of $\alpha = 2$ (see also Korbacher *et al.*, 2023), the average NF solution times are less than 1 second even for longer pick lists with $n = 30$ different articles.

Table 4: Comparison of NF and GTSP with computation times (in milliseconds) and speedup factor comparing GTSP and NF for the SPRP-SS, a two-block parallel-aisle warehouse, and a sufficient number of items (unit demand; $b_{se} = q_s = 1$).

Scatter factor	Warehouse dimension	Number of articles in pick list											
		$n = 3$			$n = 7$			$n = 15$			$n = 30$		
	(m, C)	t_{NF}	t_{GTSP}	$\frac{t_{\text{GTSP}}}{t_{\text{NF}}}$	t_{NF}	t_{GTSP}	$\frac{t_{\text{GTSP}}}{t_{\text{NF}}}$	t_{NF}	t_{GTSP}	$\frac{t_{\text{GTSP}}}{t_{\text{NF}}}$	t_{NF}	t_{GTSP}	$\frac{t_{\text{GTSP}}}{t_{\text{NF}}}$
$\alpha = 2$	(5,30)	22.5	4.6	0.3	37.9	41.6	1.1	82.4	838.1	8.9	131.6	9784.7	66.2
	(5,60)	27.6	4.9	0.3	40.0	53.4	1.3	74.4	1436.5	15.0	171.2	16 820.1	85.0
	(5,180)	34.6	5.8	0.4	51.3	100.0	1.9	88.0	2732.2	27.4	228.1	28 601.4	95.6
	(10,30)	42.8	5.1	0.2	76.8	39.0	0.6	114.3	601.9	4.8	190.8	7296.1	39.8
	(10,60)	56.8	4.6	0.2	66.2	51.6	0.8	106.9	528.7	4.3	236.7	10 971.1	44.3
	(10,180)	41.5	4.8	0.3	72.5	70.1	1.0	113.8	620.4	4.8	207.9	11 923.6	55.1
	(25,30)	60.8	4.4	0.1	103.5	44.5	0.4	175.6	590.1	3.2	459.0	9995.3	23.7
	(25,60)	73.6	4.9	0.1	89.9	38.6	0.4	175.2	352.9	2.0	381.5	6651.6	17.0
	(25,180)	84.2	5.6	0.1	99.1	55.1	0.5	170.8	322.2	1.7	361.4	4344.7	11.8
	(50,30)	79.3	5.0	0.1	131.8	71.8	0.5	302.5	1568.3	4.7	677.0	18 461.7	28.9
	(50,60)	113.8	4.4	0.1	145.2	59.5	0.4	295.3	634.3	2.1	641.7	8475.7	13.5
	(50,180)	311.5	7.6	0.1	214.1	66.8	0.4	309.8	386.3	1.3	694.6	5006.3	7.2
$\alpha = 5$	(10,30)	226.2	16.0	0.1	677.5	3582.5	7.2	1193.9	55 505.5	56.8	1782.3	*TL*	
	(10,60)	268.0	30.5	0.2	1221.4	7956.1	15.5	1600.8	57 834.6	61.8	3511.0	*TL*	
	(10,180)	362.6	25.4	0.2	840.0	6649.9	13.4	2591.9	58 170.3	42.2	3769.1	*TL*	
	(25,30)	307.1	16.3	0.1	604.4	3741.5	9.6	1163.8	57 854.6	41.4	2690.8	*TL*	
	(25,60)	542.4	33.9	0.1	1024.4	3247.1	6.1	1648.4	52 296.2	72.2	3374.6	*TL*	
	(25,180)	546.1	25.1	0.1	2427.1	4886.9	7.5	2784.1	50 925.8	43.1	3910.5	*TL*	
	(50,30)	701.5	19.2	0.1	866.9	3722.8	5.7	2020.6	54 426.0	28.5	4936.3	*TL*	
	(50,60)	1042.7	22.1	0.0	4378.5	5083.3	5.5	2991.8	51 008.7	23.0	13 729.0	*TL*	
	(50,180)	1502.9	21.7	0.0	3989.7	5430.8	2.6	5013.4	49 855.9	20.5	13 880.6	*TL*	
$\alpha = 10$	(25,30)	7389.3	153.8	0.0	8925.1	34 389.8	4.6	9570.1	*TL*		13 396.5	*TL*	
	(25,60)	8023.3	84.4	0.0	14 667.5	33 585.2	3.0	19 454.6	*TL*		24 542.0	*TL*	
	(25,180)	6195.9	95.3	0.0	14 247.6	36 423.9	2.6	25 830.5	*TL*		42 470.7	*TL*	
	(50,30)	14 024.4	83.6	0.0	35 456.2	35 214.3	1.9	49 075.5	*TL*		57 780.0	*TL*	
	(50,60)	14 747.3	120.8	0.0	23 046.9	32 552.5	1.9	25 993.7	*TL*		89 760.9	*TL*	
	(50,180)	12 072.0	73.3	0.0	27 427.3	34 496.4	2.1	47 036.3	*TL*		66 002.8	*TL*	

Note: *TL* indicates that no instance was solved within 60 seconds (60,000 milliseconds).

Table 5: Instances solved (not solved) by branch-and-cut GTSP solver for the SPRP-SS, a two-block parallel-aisle warehouse, and a sufficient number of items (unit demand; $b_{se} = q_s = 1$).

Scatter factor	Number of articles in pick list							
	$n = 3$		$n = 7$		$n = 15$		$n = 30$	
$\alpha = 2$	600	(0)	600	(0)	600	(0)	587	(13)
$\alpha = 5$	450	(0)	447	(3)	106	(344)	0	(450)
$\alpha = 10$	300	(0)	218	(82)	0	(300)	0	(300)

Finally, Table 8 in Section B of the Appendix presents the results of the computational tests obtained for the two-block SPRP-SS instances with general demand. Recall that for this type of SPRP-SS instances, there is no alternative solution approach other than our NF formulation currently available. Comparing these results with Table 4, we see that the general-demand case is computationally more demanding than the unit-demand case. For the scenario with $\alpha = 2$ and $n = 30$ just discussed, MIP solver solution times t_{NF} can exceed 2 seconds. For larger values of $\alpha \geq 5$, the spread between average computation times of corresponding (m, C, n, α)-groups can even exceed a factor of 5.

3.5. Cost Comparison Between Single-Block and Two-Block Parallel-Aisle Warehouse Layouts

Finally, we compare the cost (z_{1b}) of optimal tours for SPRP-SSs defined for a single-block and the corresponding cost (z_{2b}) for a two-block parallel-aisle warehouse. Table 6 shows the average cost reduction in percent for instances with a sufficient number of items (the unit-demand case).

For all instance groups, the addition of the middle cross-aisle in the two-block layout results in a significant cost reduction. On the one hand, the higher the numbers m, C, and n (of aisles, cells per aisle, and articles

Table 6: Average cost reduction in percent ($100 \cdot (z_{1b} - z_{2b})/z_{1b}$) of SPRP-SS instances with two-block compared to single-block parallel-aisle warehouse layout; all instances have a sufficient number of items (unit demand; $b_{se} = q_s = 1$).

Scatter factor	Dimension (m, C)	Number of articles in pick list $n = 3$	$n = 7$	$n = 15$	$n = 30$
$\alpha = 1$	$(5, 30)$	11.2	16.2	15.1	6.2
	$(5, 60)$	10.1	17.4	19.2	10.3
	$(5, 180)$	13.9	19.0	24.5	14.5
	$(10, 30)$	11.4	19.8	20.3	17.0
	$(10, 60)$	15.1	23.6	27.7	23.1
	$(10, 180)$	15.4	25.1	31.6	26.1
	$(25, 30)$	9.1	16.0	20.1	23.5
	$(25, 60)$	12.0	19.7	24.3	27.2
	$(25, 180)$	17.3	28.9	36.2	36.6
	$(50, 30)$	6.1	10.4	16.1	20.2
	$(50, 60)$	10.4	18.5	24.9	29.1
	$(50, 180)$	14.0	25.6	33.4	36.4
$\alpha = 2$	$(5, 30)$	8.6	13.4	14.6	12.5
	$(5, 60)$	10.2	18.8	18.0	18.3
	$(5, 180)$	8.5	21.0	19.4	21.4
	$(10, 30)$	8.5	14.2	16.1	17.9
	$(10, 60)$	11.7	16.8	22.4	22.8
	$(10, 180)$	11.1	23.1	25.3	25.8
	$(25, 30)$	8.3	13.1	15.7	18.7
	$(25, 60)$	11.5	18.5	20.9	22.2
	$(25, 180)$	15.0	21.4	26.4	29.6
	$(50, 30)$	7.7	10.8	13.7	15.7
	$(50, 60)$	7.8	13.2	18.3	20.9
	$(50, 180)$	12.6	21.0	22.9	27.5
$\alpha = 5$	$(10, 30)$	3.0	12.5	13.1	15.8
	$(10, 60)$	1.6	11.7	15.8	18.7
	$(10, 180)$	3.5	12.4	18.1	19.8
	$(25, 30)$	3.9	11.4	10.9	13.7
	$(25, 60)$	3.5	13.7	18.0	17.9
	$(25, 180)$	3.2	15.3	19.6	21.0
	$(50, 30)$	7.9	10.2	10.6	11.9
	$(50, 60)$	9.9	12.9	14.2	15.6
	$(50, 180)$	6.3	16.2	23.2	22.7
$\alpha = 10$	$(25, 30)$	2.5	11.5	13.1	14.5
	$(25, 60)$	1.6	10.1	18.0	16.8
	$(25, 180)$	0.9	6.9	19.4	22.5
	$(50, 30)$	3.4	13.0	10.2	11.4
	$(50, 60)$	3.7	14.3	16.0	15.8
	$(50, 180)$	2.1	12.0	20.3	23.1

in the pick list), the higher the respective cost reduction. On the other hand, the higher the scatter factor α, the lower the cost reduction. We interpret the latter effect in the sense that, with an additional middle cross-aisle, the tour length decreases less strongly if articles are available at several positions and, especially, in several aisles. However, even with massive scattering (see results, e.g., $\alpha = 10$) and for relatively small warehouses and short pick lists, the positive effect of a cross-aisle is always significant. Overall, the highest average cost reduction is 36.6% for the group $(\alpha, m, C, n) = (1, 25, 180, 30)$.

4. Conclusions and Outlook

We have introduced a new approach to exactly solve the SPRP-SS which is a two-level optimization problem. The higher-level decision is the selection of pick positions for requested articles that are collectible from, in general, more than one pick position per article. The modeling of the lower-level picker routing decisions is inspired by the dynamic program of Ratliff and Rosenthal (1983). To properly cope with the selection aspect, we extended the state space of Ratliff and Rosenthal. Since solving a dynamic program over the extended state space does not result in a feasible solution, we add consistency constraints and propose to solve the resulting picker-routing problems with the help of a MIP solver.

Our work shares some similarities with recent approaches that exploit the fact that pseudo-polynomial formulations exist for several fundamental problems (de Lima *et al.*, 2022). For example, in cutting and packing, a feasible structure, i.e., a cutting pattern or packing of items, is in one-to-one correspondence with a path in the underlying state space over which the pseudo-polynomial formulation is defined. Solving these formulations with a MIP solver has become possible even for large instances, either directly or by nesting optimal solutions with (a sequence of) restrictions and relaxations of the respective pseudo-polynomial formulation. Our approach is, however, different as we do not directly use the original state space. We first extend the state space, then add consistency constraints, and finally solve the resulting pure binary formulation using a MIP solver.

The new modeling and solution approach has several advantages:

- We have presented a new binary formulation for the SPRP-SS which is simple and elegant in the sense that only very few types of variables and constraints are needed to entirely capture the respective problem (one may compare with the models of Goeke and Schneider, 2021; Su *et al.*, 2023).
- Our modeling approach is generic, since it applies to variants of the picker routing problem whenever a dynamic-programming approach is known for this variant. This includes different warehouse layouts, multiple possible endpoints of a picker tour, and other extensions (Masae *et al.*, 2020b; Löffler *et al.*, 2022). To prove this statement, we have adapted the new solution approach from the standard single-block parallel-aisle warehouse layout (used as an example in the majority of the warehousing publications) to the two-block variant (cf. Section 2.5).
- To date, all exact solution approaches for the SPRP-SS use integer programming formulations solved directly with a MIP solver. The most competitive one is the recent and remarkably well-performing formulation of Goeke and Schneider (2021), for which the associated article received the *Meritorious Paper Award* of the journal *INFORMS Journal on Computing*. Our new network flow-based formulation for the SPRP-SS mostly outperforms it with an overall average speedup ratio of 3.6.

Other integrated operational planning problems in warehousing that include a picker routing component (see van Gils *et al.*, 2018) may benefit from our modeling and solution method as well. Examples of two- or multi-level optimization problems are storage assignment (Petersen and Schmenner, 1999), order batching and sequencing (Menéndez *et al.*, 2017), and *joint order batching and picker routing* (JOBPRP, Wahlen and Gschwind, 2022) with scattered storage. This latter JOBPRP extension gives rise to an integrated three-level optimization problem of grouping orders, selecting pick positions, and routing pickers.

Acknowledgement

The authors thank Laura Korbacher for fruitful discussions and Siska Langsdorf for her support to produce the figures of the Appendix.

References

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

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.

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.

Corberán, Á. and Laporte, G., editors (2014). *Arc Routing.* Society for Industrial and Applied Mathematics, Philadelphia, PA.

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 Lima, V. L., Iori, M., and Miyazawa, F. K. (2022). Exact solution of network flow models with strong relaxations. *Mathematical Programming*. doi: 10.1007/s10107-022-01785-9.

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.

Glock, C. H. and Grosse, E. H. (2012). Storage policies and order picking strategies in U-shaped order-picking systems with a movable base. *International Journal of Production Research*, **50**(16), 4344–4357.

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.

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.

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.

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.

Korbacher, L., Heßler, K., and Irnich, S. (2023). The single picker routing problem with scattered storage: Modeling and evaluation of routing and storage policies. Technical Report LM-2023-01, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany. URL https://logistik.bwl.uni-mainz.de/files/2023/02/LM-2023-01.pdf.

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. (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.

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

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, O., 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.

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.

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

Santini, A. (2022). Fork of the concorde TSP solver with an easier build procedure. GitHub. https://github.com/alberto-santini/concorde-easy-build.

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., and Frazelle, E.H.and Tanchoco, J. (2003). *Facilities Planning*. John Wiley & Sons, Hoboken, NJ.

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.

Wahlen, J. and Gschwind, T. (2022). Branch-price-and-cut-based solution of order batching problems. *Transportation Science*. Forthcoming. Available as Technical Report L-2022-01, RPTU, Germany, https://logistik.wiwi.uni-kl.de/fileadmin/logistik.wiwi.uni-kl.de/publikationen/Artikel/L-2022-01.pdf.

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.

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

Appendix

A. Instance Generation Procedure for SPRP-SS Instances

First, the generation procedure described by Weidinger and Goeke and Schneider completely fills the warehouse with articles. The assumption is that one article is stored in each of the $m \cdot C$ possible pick positions. In particular, opposite cells in an aisle are considered as one pick position. The scatter factor α can be realized by defining a set of $\xi = \lceil mC/\alpha \rceil \geq n$ different articles stored in the warehouse (not all of them are in the final pick list). To ensure that exactly ξ different articles are available, each of them is randomly assigned to a unique pick position.

Second, all ξ different articles are divided into the three classes A, B, and C (in practice depending on the turnover rate) with 20% in class A, 30% in class B, and 50% in class C. The ABC class-based addition of further SKUs must now ensure that (on average) 80% of the positions are stocked with A-class articles, 15% with B-class articles, and 5% with C-class articles. One by one, articles are assigned to positions as follows: According to the 80-15-5 distribution, a class is chosen. Then, a random article of that class is selected. Finally, this article is assigned randomly to one of the (up to this point) free storage positions in the warehouse.

Third, supply values are then determined with a further distinction between instances with

- a *sufficient number of items* or *unit demand*: Here, the demand and supply values are set to $q_s = b_{sp} = 1$ for all $s \in S$ and $p \in P_s$;
- *general supply and demand*: Here, the supply available at a position is randomly drawn as $b_{sp} \in \{1, 2, 3\}$. Moreover, demands are randomly drawn as $q_s \in \{1, \ldots, \min(6, \sum_p b_{sp})\}$.

Finally, the pick list is generated so that it contains exactly n different articles. Iteratively, a pick position is randomly drawn. If the SKU at this position is not yet an article in the pick list, this article s is chosen to be included in the pick list.

B. Results for SPRP-SS Instances with General Supply and Demand

For the sake of brevity, Tables 7 and 8 have been moved to the Appendix. The former is structurally identical to Table 2 but for general supply and demand. The latter reports average computation times for the NF model, again for general supply and demand.

Table 7: Comparison of NF and GS with computation times (in milliseconds) and speedup factor comparing GS and NF for the SPRP-SS, a single-block parallel-aisle warehouse, and general supply and demand.

Scatter factor	Warehouse dimension	Number of SKUs in pick list							
		$n = 3$		$n = 7$		$n = 15$		$n = 30$	
	(m, C)	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$	t_{NF}	$\frac{t_{\text{GS}}}{t_{\text{NF}}}$
$\alpha = 2$	$(5, 30)$	46.8	4.5	67.0	2.0	87.9	1.1	107.9	0.9
	$(5, 60)$	47.4	5.1	56.1	2.7	99.0	1.2	127.2	0.9
	$(5, 180)$	59.4	6.6	71.8	2.3	106.7	1.0	167.1	0.8
	$(10, 30)$	70.8	5.4	92.7	2.3	189.7	1.4	231.5	1.0
	$(10, 60)$	56.7	9.6	78.9	2.9	159.4	1.6	309.2	0.9
	$(10, 180)$	71.4	6.9	87.8	2.7	147.4	1.3	344.5	0.8
	$(25, 30)$	78.6	5.3	122.6	3.4	296.5	2.0	712.3	1.4
	$(25, 60)$	87.1	8.6	134.1	3.9	268.7	2.1	589.2	1.3
	$(25, 180)$	111.7	8.8	113.4	6.2	300.0	2.1	643.4	1.5
	$(50, 30)$	162.9	4.6	226.6	5.8	415.8	3.1	847.3	1.9
	$(50, 60)$	146.6	7.2	228.4	7.1	387.1	3.3	895.1	2.2
	$(50, 180)$	199.1	16.1	214.5	12.5	416.0	4.5	931.3	2.5
$\alpha = 5$	$(10, 30)$	226.4	3.4	505.8	1.5	794.4	0.8	1021.3	0.6
	$(10, 60)$	173.6	4.8	448.6	2.4	859.4	1.0	2183.4	0.5
	$(10, 180)$	134.0	5.5	396.8	3.0	1070.6	1.1	2980.6	0.5
	$(25, 30)$	558.2	4.5	1190.9	2.8	1913.9	1.1	4141.6	0.6
	$(25, 60)$	552.2	4.7	1229.1	2.7	2210.2	1.4	4174.6	0.6
	$(25, 180)$	439.6	7.7	912.3	4.5	1372.5	1.8	4862.5	0.7
	$(50, 30)$	667.3	4.2	1186.1	2.9	1369.5	2.8	4398.7	1.3
	$(50, 60)$	887.7	6.3	1021.8	5.1	1865.1	3.4	5889.4	1.3
	$(50, 180)$	461.9	8.6	964.9	9.3	1755.8	4.4	5620.5	1.6
$\alpha = 10$	$(25, 30)$	2022.4	4.0	3635.7	1.5	6265.0	0.7	8669.6	0.6
	$(25, 60)$	1657.2	3.2	6333.0	1.7	9376.2	0.9	10 845.0	0.6
	$(25, 180)$	914.0	5.2	3534.5	2.5	9228.1	1.5	13 239.4	0.8
	$(50, 30)$	3643.5	1.4	3959.6	2.3	10 193.5	1.1	17 616.0	0.8
	$(50, 60)$	2307.5	3.5	3425.4	3.3	9315.7	1.3	17 902.1	0.9
	$(50, 180)$	1599.1	4.8	4260.0	6.1	11 015.0	2.7	20 612.2	1.7

Table 8: Computation times (in milliseconds) of NF for the SPRP-SS, two-block parallel-aisle warehouse, and general supply and demand.

Scatter factor	Warehouse dimension	Number of articles in pick list			
		$n = 3$	$n = 7$	$n = 15$	$n = 30$
	(m, C)	t_{NF}	t_{NF}	t_{NF}	t_{NF}
$\alpha = 2$	$(5, 30)$	21.6	44.9	78.3	119.1
	$(5, 60)$	24.7	46.4	103.7	177.6
	$(5, 180)$	34.3	64.8	105.1	258.9
	$(10, 30)$	52.3	81.0	173.4	276.3
	$(10, 60)$	55.4	86.8	163.8	408.9
	$(10, 180)$	75.6	85.0	128.4	385.8
	$(25, 30)$	82.1	135.5	278.5	678.4
	$(25, 60)$	63.9	119.9	285.5	680.0
	$(25, 180)$	72.5	144.7	322.8	722.1
	$(50, 30)$	156.3	579.3	453.4	945.6
	$(50, 60)$	110.7	299.2	419.2	1269.0
	$(50, 180)$	211.6	215.2	480.0	2295.1
$\alpha = 5$	$(10, 30)$	537.5	1147.4	1453.7	2237.4
	$(10, 60)$	836.2	1448.4	1882.0	4056.2
	$(10, 180)$	891.9	1383.6	3353.7	6011.4
	$(25, 30)$	1697.3	3076.7	4161.3	6685.3
	$(25, 60)$	1871.9	4331.6	4074.5	8855.6
	$(25, 180)$	2336.7	2144.2	4066.7	12 807.7
	$(50, 30)$	8172.5	6566.0	9222.6	24 512.2
	$(50, 60)$	3376.1	4007.4	12 136.5	60 624.2
	$(50, 180)$	3350.2	7916.7	28 560.3	77 918.1
$\alpha = 10$	$(25, 30)$	8774.1	9433.0	12 728.5	25 128.6
	$(25, 60)$	12 120.0	28 881.6	28 692.2	32 253.0
	$(25, 180)$	11 025.8	25 136.3	33 230.2	32 516.8
	$(50, 30)$	15 735.8	56 414.1	179 451.1	110 293.4
	$(50, 60)$	25 184.1	57 259.9	78 620.4	437 665.8
	$(50, 180)$	18 090.0	39 523.2	147 424.7	350 290.7

C. Dynamic-Programming State Space for Two-block Parallel-Aisle Warehouse Layout

The dynamic program of Roodbergen and de Koster (2001b) solving the SPRP for two-block parallel-aisle warehouse has $3m+1$ stages. For each aisle $j \in J = \{1, 2, \dots, m\}$, the stage j^- describes a PTS before aisle j is traversed, the stage j^{+x} when the back ('x') block in aisle j is traversed, and the stage j^{+y} when the front ('y') block in aisle j is traversed. An artificial stage $(m+1)^-$ is added as in the one-block case.

The states of the DP describe vertex degrees and the connectivity of the PTS. Since there are three crossing points per aisle now (in the back cross-aisle, middle cross-aisle, front cross-aisle), the vertices a_j, b_j, and c_j are introduced for this purpose. As a result, there are 25 states relevant to construct an optimal picker tour:

$$\begin{aligned}\mathscr{S} \;=\; & \{\texttt{0000c}, \texttt{0001c}, \texttt{E001c}, \texttt{0E01c}, \texttt{00E1c}, \texttt{EE01c}, \texttt{E0E1c}, \texttt{0EE1c}, \texttt{EEE1c}, \texttt{UU01c}, \texttt{U0U1c}, \texttt{0UU1c}, \texttt{EUU1c}, \\ & \texttt{UEU1c}, \texttt{UUE1c}, \texttt{EE02c}, \texttt{E0E2c}, \texttt{0EE2c}, \texttt{EEE2c.a-bc}, \texttt{EEE2c.b-ac}, \texttt{EEE2c.c-ab}, \texttt{EUU2c}, \texttt{UEU2c}, \texttt{UUE2c}, \texttt{EEE3c}\}\end{aligned}$$

(the first three symbols again describe vertex degrees for the back/middle/front cross aisle). The connectivity information is more detailed compared to the DP of Ratliff and Rosenthal: there can be one (`1c`), two (`2c`), or three (`3c`) connected components. Moreover, if all three vertices a_j, b_j, and c_j are even (EEE), two connected components can result from having middle and front cross-aisle connected (`a-bc`), back and front cross-aisle connected (`b-ac`), or back and middle cross-aisle connected (`c-ab`).

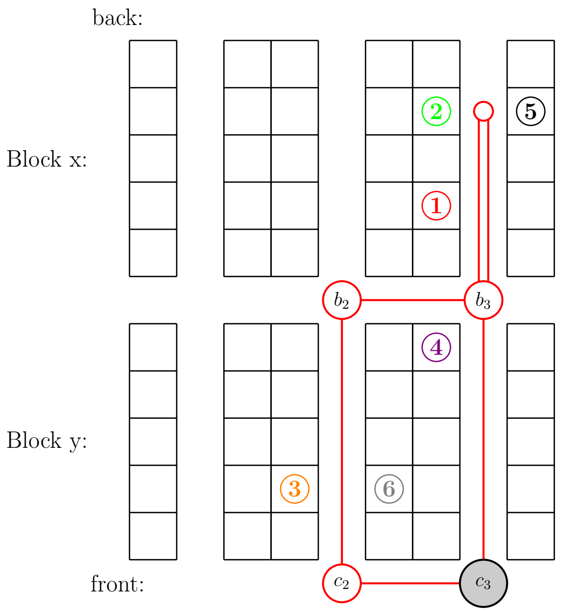


Figure 3: Instance of the SPRP for a 2-block parallel-aisle warehouse. The pick list contains six SKUs $S = \{1, 2, 3, 4, 5, 6\}$ (unit demand); pick operations are encircled.

An instance of the SPRP for a 2-block parallel-aisle warehouse and the optimal picker tour are depicted in Figure 3. The associated state space is shown in Figure 4. Moreover, the highlighted o-d-path from state `0000c` at stage 1^- to state `0001c` at stage 4^- (drawn in red/thick) describes the optimal picker tour.

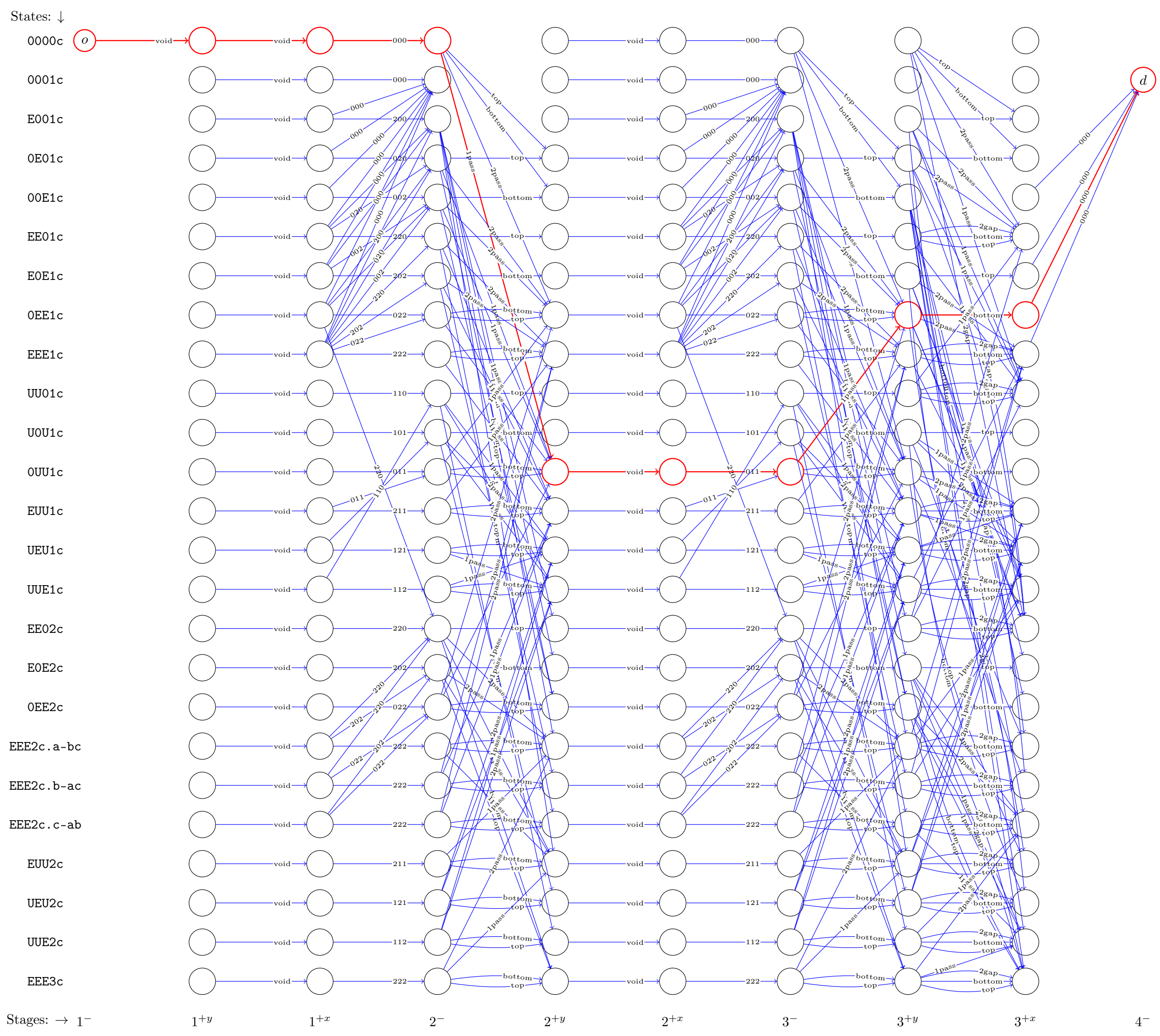


Figure 4: State space (V, E) of the dynamic program for a 2-block parallel-aisle warehouse and optimal sequence of states and actions (in red/thick).

D. GTSP Formulation and Branch-and-Cut Algorithm

For describing the GTSP formulation and branch-and-cut algorithm of Fischetti *et al.* (2002), we make the following assumptions. An instance of the GTSP is given by an undirected graph (V, E) with vertex set V and edge set E. The vertices V are partitioned into m clusters $C_1, C_2, \ldots, C_m$, i.e., $V = \bigcup_{h \in H} C_h$ and $C_h \cap C_{h'} = \varnothing$ for all $h, h' \in H = \{1, 2, \ldots, m\}$ with $h \neq h'$. Let $h(i) \in H$ be the index of the cluster that contains $i \in V$, i.e., $i \in C_{h(i)}$. Edges $e = \{i, j\} \in e$ exist between every two i and j vertices with $h(i) \neq h(j)$. The cost of edge $e = \{i, j\} \in E$ is given by an integer number $c_e = c_{ij} = c_{ji}$. The GTSP is the problem of finding a cycle $T \subset E$ of minimal length $\sum_{e \in T} c_e$ such that T goes through every cluster at least once.

For $S \subseteq V$, the following sets and number are defined:

$$\begin{aligned} E(S) &= \{\{i,j\} \in E : i, j \in S\} \\ \delta(S) &= \{\{i,j\} \in E : i \in S, j \notin S \text{ or } i \notin S, j \in S\} \\ \mu(S) &= |\{h \in H : C_h \subseteq S\}| \end{aligned}$$

For $i \in V$, we write $\delta(i)$ as a shorthand notation for $\delta(\{i\})$.

Two types of decision variables are present in the GTSP formulation of Fischetti *et al.*: Binary variables x_e indicate whether edge $e \in E$ is present in the selected cycle and binary variables y_i indicate whether vertex $i \in V$ is included in the selected cycle. The binary formulation reads as follows:

$$z_{GTSP} = \sum_{e \in E} c_e \tag{4a}$$

$$\text{subject to} \sum_{e \in \delta(i)} x_e - 2y_i = 0 \qquad \forall i \in V \tag{4b}$$

$$\sum_{i \in C_h} y_i \geq 1 \qquad \forall h \in H \tag{4c}$$

$$\sum_{e \in \delta(S)} x_e - 2y_i - 2y_j \geq -2 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, i \in S, j \in V \setminus S \tag{GSEC}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4d}$$

$$y_i \in \{0, 1\} \qquad \forall i \in V \tag{4e}$$

The objective (4a) is to minimize the length of the cycle. Constraints (4b) couple the x- and y-variables guaranteeing that a selected vertex has degree two. Constraints (4c) ensure that at least one vertex per cluster is visited. The *generalized subtour elimination constraints* (GSEC) forbid subtours over the selected vertices. Last, the domain of the variables is specified in (4d) and (4e).

There exist strengthened versions of the GSEC when complete clusters are included in S or its complement:

$$\sum_{e \in \delta(S)} x_e - 2y_i \geq 0 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, i \in S, \mu(V \setminus S) > 0 \tag{GSEC'}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad \forall S \subset V, 2 \leq |S| \leq |V| - 2, \mu(S) > 0, \mu(V \setminus S) > 0 \tag{GSEC''}$$

Our re-implementation of the branch-and-cut algorithm described by Fischetti *et al.* uses the `lazy` and `user` callback functions of the `C++` API of CPLEX.

- A method like `GSEC-BUILD` (Fischetti *et al.*, 2002, p. 382) is used to find, for a given subset S, a most violated facet-defining inequality of type (GSEC), (GSEC'), or (GSEC''). The given set S is (heuristically) modified into S' in order to arrive at $\mu(S') > 0$ and/or $\mu(V \setminus S') > 0$ and to select better vertices i (and j).
- Integer solutions (x^*, y^*) (in the `lazy` cut callback) are inspected using a union-find algorithm (Tarjan, 1975) to very quickly identify subsets S.

- For the separation of the strongest inequalities (GSEC”), we only rely on `GSEC-BUILD` and do not use a dedicated method like `GSEC-H1`, since we did not see a computational advantage in pre-tests.
- For fractional solutions (x^*, y^*), a spanning tree-based method like `GSEC-H2` (Fischetti *et al.*, 2002, p. 382) is used as a fast heuristic separation procedure.
- Moreover, a method like `GSEC-SEP` (Fischetti *et al.*, 2002, p. 382) is used (Fischetti *et al.*, 2002, p. 383) for exact GSEC separation (to be precise, exact only if $\max_{i \in V} y_i^* = 1$, which is very often fulfilled).
- The overall strategy for calling the above heuristic and exact separation procedures follows a similar logic as described by Fischetti *et al.* However, we use a simplified management of selecting a subset of violated GSECs compared to the method `SEPARATION` (Fischetti *et al.*, 2002, p. 386) (no loops with different minimum thresholds for violation), since the recent versions of CPLEX are excellent in handling cut pools itself.