Inventory routing in a warehouse: The storage replenishment routing problem

Melih Çelik[1]

School of Management, University of Bath, Claverton Down Rd, Combe Down, Bath BA2 7AY, United Kingdom

Claudia Archetti

ESSEC Business School, Av. Bernard Hirsch, B.P. 50105 95021, Cergy Pontoise Cedex, France

Haldun Süral

Department of Industrial Engineering, Middle East Technical University, Universiteler Mh, Dumlupinar Bl. No:1, 06800 Cankaya, Ankara, Turkey

Abstract

In warehouses, storage replenishment operations involve the transportation of items to capacitated item slots in the forward storage area from reserve storage. These items are later picked from these slots as demand arises. While order picking constitutes the majority of warehouse operating costs, efficient management of replenishment operations is important to ensure the availability of the items for picking and to decrease the operating costs due to replenishment, which might be particularly higher in warehouses with fast-moving items (e.g., e-commerce warehouses or retail distribution centers).

In this paper, we define the storage replenishment routing problem in a parallel-aisle warehouse, where replenishment and order picking operations are carried out in successive cycles with time limits. The aim is to determine the item slots that will be replenished and the route of the replenishment worker in each replenishment cycle, so as to minimize the total travel time and ensure the availability of items at the start of the cycle they will be picked. We present complexity results on different variants of the problem and show that the problem is $\mathcal{NP}$-hard in general. Consequently, we adapt a heuristic approach based on a priori routing and inspired by the literature on the inventory routing problem. We use randomly generated warehouse instances to analyze the effects of different a priori routing methods and demand skewness patterns on replenishment performance, and to compare the proposed approach to benchmarks that mimic practice.

Keywords: Routing, warehouse management, storage replenishment, order picking, inventory routing, heuristics

1. Introduction

Warehouses play an increasingly important role in supply chains to meet the challenges arising from such factors as the reduced delivery times, higher number of orders, and fewer items per order. Among the operations performed within a warehouse, the most labor-intensive and costly activity is order picking, which refers to the

[1]Corresponding author, e-mail: M.Celik@bath.ac.uk

retrieval of items from storage locations in response to customer demand. It is estimated that order picking constitutes around 55% of the warehouse operating costs (Bartholdi & Hackman, 2019). The productivity of order picking depends on decisions at the tactical and operational level including the warehouse layout, assignment of items to storage locations (*slots*) within the warehouse, batching of multiple orders to be picked together, and routing of order pickers to reduce travel time (de Koster et al., 2007).

To address the increased need for material handling due to the aforementioned challenges, many warehouses make use of separate *reserve* and *forward storage areas* in order to increase the efficiency of order picking. In such cases, items are put away into the reserve storage area upon receipt, where they are stored in bulk amounts. Upon need based on the picking schedule, items are broken down into smaller loads, such as cartons or cases, and transported into the forward storage area (a process known as *storage replenishment*), where storage is in smaller quantities. The motivation to have a forward storage area is to sacrifice space efficiency and replenishment time in order to provide better accessibility of slots in this area for picking.

Storage replenishment and order picking activities may be performed in conjunction or in sequence, depending on the intensity of the picking operations and available space in the forward storage area. In both cases, the replenishment process aims to ensure that items are available for picking in a timely manner, so that the order pickers do not incur additional waiting times for urgent replenishment. In this regard, storage replenishment plays an important role in the adherence to delivery times. When the sizes and quantities of the items to be replenished are small (e.g., when items are picked in cases or pieces), multiple slots can be replenished by a single replenishment trip into the forward storage area. This implies that by designing an efficient replenishment route, not only can replenishment be finished in shorter time, but also more slots can be replenished without significant time added. Motivated by the importance of routing on the efficiency of replenishment, we define the *storage replenishment routing problem* (SRRP), which aims to determine (i) the slots to be replenished and (ii) the sequence of slots in the replenishment route in successive replenishment cycles, subject to a replenishment cycle time limit and load capacity. The objective of the SRRP is to minimize the total replenishment time while ensuring no stockouts during order picking.

We consider the SRRP in a *single-block, parallel-aisle* forward storage area, an example of which is shown in Figure 1. Such a storage area consists of equal-length *aisles* parallel to each other, with the *front* and *back cross aisles* that intersect them orthogonally at each end. A feasible SRRP route starts at the *depot*, which constitutes the entry point to the forward storage area, replenishes a number of items, and returns to the depot within the pre-determined replenishment cycle time limit.

The SRRP is considered in a warehouse environment where storage replenishment and order picking are performed in successive cycles. Since it takes into account multiple upcoming order picking cycles in making the replenishment decisions, the SRRP resembles the well-studied inventory routing problem (IRP, e.g., Coelho et al., 2013). On the other hand, given the special structure of the graph corresponding to the forward storage area, the routing problem is much easier to solve for the SRRP than for the IRP (Ratliff & Rosenthal, 1983). Making use of this fact, we modify a heuristic approach based on an a priori routing scheme for the IRP to obtain an efficient solution of the SRRP. In computational experiments on instances based on the IRP literature,

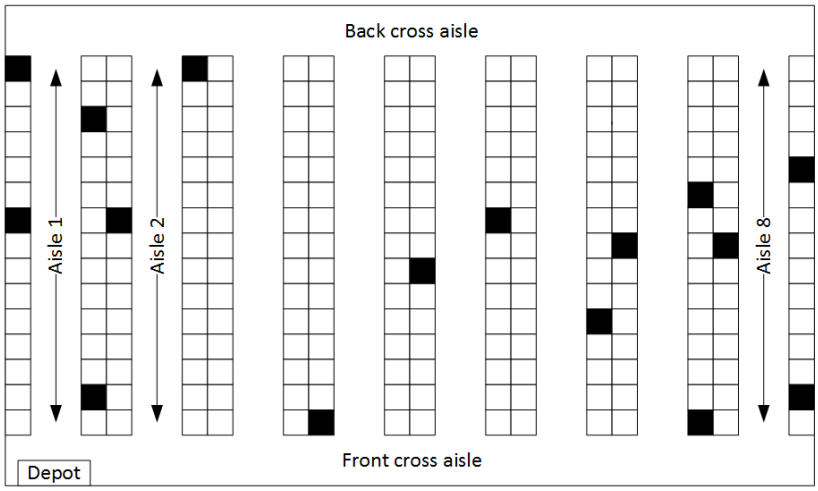

Figure 1: A single-block parallel-aisle forward storage area with eight aisles and 16 item slots in the replenishment route

we show that this approach not only finds near-optimal replenishment routes within short computational times, but also outperforms various benchmark replenishment schemes that mimic practice.

The remainder of this paper is organized as follows. In the next section, we discuss the relevant literature on storage replenishment, order picking and the IRP. This is followed by a formal definition of the SRRP in Section 3, where a mathematical model is formulated and the complexity of the problem is analysed. Section 4 presents the a priori routing heuristic to solve the SRRP. We discuss our computational experiments in Section 5 and conclude the paper in Section 6.

2. Literature Review

From a practical perspective, the SRRP involves the interaction between storage replenishment and order picking in warehouses. Furthermore, the inclusion of inventory replenishment and routing decisions generate similarities between the SRRP and the IRP. In what follows, we review the extant literature on these three problems and underline our contributions.

2.1. Warehouse Storage Replenishment

At the tactical level, the existence of forward and reserve storage areas in a warehouse gives way to the *forward-reserve allocation problem*, where the main decisions involve the size of the forward storage area, which items to store in the forward area, and how much to store of each item, with the objective to minimize the total cost (or time) for replenishment. The studies in this stream are distinguished by whether the forward and reserve areas are in separate locations in the warehouse (e.g., Hackman et al., 1990; van den Berg et al., 1998; Gu et al., 2010) or in the same rack (e.g., Thomas & Meller, 2015; Wu et al., 2020). In the SRRP, we assume that decisions related to forward-reserve allocation have already been made, which deems this problem outside the scope of our work.

A limited number of papers consider storage replenishment at the operational level. Here, the main consideration is to address the timing and quantities of replenishments to avoid any stockouts or delays during order

picking. The studies in this stream can be classified by whether replenishment is carried out in conjunction or sequentially with order picking.

The main challenge with simultaneous replenishment and order picking is that even when the list of items to be picked in each cycle are known, estimating the actual timing of each pick may be impossible or computationally very difficult. Gong et al. (2008) consider such a case with multiple carts, where each cart either replenishes or picks items from multiple stock locations in each trip. The authors propose a simulated annealing approach to schedule the routes of these carts to minimize the total makespan, which includes travel, loading/unloading and wait times. This is extended by Takano et al. (2011) to include rearrangement of items from one stock location to another and multiple job types (picking, replenishment and rearrangement) in the same trip, where a metaheuristic approach based on simulated annealing that uses local search moves and dispatching rules is put forward. Gagliardi et al. (2008) address the problem in a single-aisle pick-to-belt system with a single picker and replenishment technician, and propose four different replenishment heuristics to minimize stockouts. Here, each replenishment trip involves transporting a pallet of a single item (*unit-load replenishment*). Carrasco-Gallego & Ponce-Cueto (2009) and de Vries et al. (2014) also consider unit-load replenishment under an (s, S) replenishment policy for each item. Based on different assumptions, three heuristic replenishment policies are put forward in these studies.

The SRRP considers the case where replenishment and order picking are conducted in consecutive intervals. The number of papers addressing this case is even more limited. Of these, van den Berg et al. (1998) combine forward-reserve allocation and unit-load storage replenishment. A number of heuristics are proposed to minimize the total expected labor time. Kim et al. (2003) also focus on unit-load replenishment in a specific layout where the forward and reserve areas for each item are located in the same zone. The authors develop a heuristic approach to minimize the total number of replenishments over the planning horizon. Haverhals (2019) includes the possibility of picking from reserve storage in case of a stockout (*bulk picking*) and develops approaches to find the optimal inventory replenishment policies to minimize the total expected order picking, bulk picking and replenishment cost. To the best of our knowledge, none of the papers in this stream take into account the routing decisions during replenishment.

2.2. Warehouse Picker Routing

Given the importance of order picking in warehouse operating costs, there exists a vast amount of literature on order picking. de Koster et al. (2007) provide a framework for the decision problems at the strategic, tactical and operational levels relevant to order picking, as well as a comprehensive review of the literature. Recent studies on order picking aim to address combined subsets of these problems (e.g., order batching, picker routing, storage assignment, zoning). A review of such studies is given by van Gils et al. (2018). On average, more than half of the order picking time is spent during travel within the warehouse (Tompkins et al., 2010). Hence, a significant portion of the order picking literature is devoted to problems addressing efficient picker routing. Masae et al. (2020) present a detailed literature review of this stream.

Despite being closely related to the $\mathcal{NP}$-hard Traveling Salesman Problem (TSP), the picker routing problem

in a single-block parallel-aisle warehouse (as in Figure 1) can be solved in polynomial time using dynamic programming (Ratliff & Rosenthal, 1983). A mixed integer programming formulation for this problem is also put forward by Scholz et al. (2016). Although this formulation is stronger than the corresponding TSP formulations, its run time far exceeds that of the dynamic programming approach. Exact approaches exist for different versions of the problem including turn penalties (Çelik & Süral, 2016), weight, fragility and category constraints (Chabot et al., 2017), precedence constraints (Žulj et al., 2018), non-traditional warehouse layouts (Çelik & Süral, 2014), and multiple blocks (Roodbergen & de Koster, 2001b; Ruberg & Scholz, 2016; Scholz & Wäscher, 2017; Pansart et al., 2018; Glock et al., 2019).

In practice, the complicated nature of the optimal picker routes and the difficulty of their implementation have led to the development of simple-to-apply heuristics. For a single-block parallel-aisle warehouse, Hall (1993) proposes the *S-shape* (traversal), *midpoint* and *largest gap* heuristics, and evaluates their expected travel distances. Similar rules-of-thumb, called *return* and *composite*, are put forward by Petersen (1997). Makris & Giakoumakis (2003) modify the k-interchange heuristic for the TSP and Menéndez et al. (2017) propose a variable neighborhood search approach. Heuristic methods have also been proposed to address picker routing when items are stored in multiple scattered locations within the warehouse (Weidinger, 2018; Weidinger et al., 2019) and when multiple blocks exist (Vaughan & Petersen, 1999; Roodbergen & de Koster, 2001a; Theys et al., 2010; Dijkstra & Roodbergen, 2017; De Santis et al., 2018; Çelik & Süral, 2019).

2.3. Inventory Routing

As explained in more detail in Section 3, the SRRP shares many common characteristics with the inventory routing problem (IRP), which can be defined as a problem where a single commodity has to be distributed from a central supplier to a set of retailers over a given planning horizon. Retailers face a given per-period consumption of the commodity and the replenishment policy has to be such that no stock-out is incurred, i.e., retailers have a sufficient amount of stock to face the consumption in each period. Moreover, a maximum inventory level is established at each retailer (which might correspond to the capacity of the retailer's warehouse). A fleet of capacitated vehicles is available to perform the distribution and might be used in each period of the planning horizon. Finally, costs are associated with the routing of the vehicles that perform the distribution and with the holding of inventories both at the supplier and retailers. The objective is to determine the distribution plan that minimizes the total routing and inventory costs and satisfies inventory capacity constraints, vehicle capacity constraints and no stock-out constraints.

The IRP has attracted a lot of attention in the research community, due to the challenges raised by the intrinsic complexity of the problem, to the many practical applications it is related to, and to the economical advantages gained when managing the routing and the inventory management operations in an integrated way (as done in the IRP), as opposed to handling them separately (see Archetti & Speranza (2016) for a computational study of the gains).

The literature on the IRP is wide. Recent tutorials and surveys on the IRP are available in Bertazzi & Speranza (2012, 2013), Coelho et al. (2013) and Roldán et al. (2017), and the reader is referred to these works

for an exhaustive overview of contributions on the IRP. In what follows, we restrict our review to the most recent contributions on exact and heuristic algorithms for the IRP.

Despite the complexity of the problem, the recent literature is much richer in exact approaches than in heuristics. Exact approaches are mainly based on the branch-and-cut scheme and handle the case of a fleet of multiple vehicles. Archetti et al. (2014) present different formulations and branch-and-cut algorithms for the problem. The multi-product case is studied in Coelho & Laporte (2013a), while branch-and-cut algorithms for different variants of the problem are proposed in Coelho & Laporte (2013b). Adulyasak et al. (2014) study the Production Routing Problem, which is a generalization of the IRP where production decisions at the supplier are taken into account. They propose a branch-and-cut algorithm which can solve the IRP as well. Currently, the state-of-the-art branch-and-cut algorithm for the IRP is proposed in Coelho & Laporte (2014), where different classes of valid inequalities strengthen the formulation. To the best of our knowledge, the only branch-and-price algorithm for the IRP is due to Desaulniers et al. (2016), where it is shown that the performance is comparable to that of the branch-and-cut algorithm in Coelho & Laporte (2014).

Heuristic approaches for the IRP are mainly based on matheuristic and metaheuristic schemes. Coelho et al. (2012) propose an Adaptive Large Neighborhood Search where the MIP formulations of subproblems are iteratively solved to optimality. Archetti et al. (2017) propose a matheuristic where first a tabu search is used to find a good quality solution and, then, a MILP is solved in the attempt to improve it. Chitsaz et al. (2019) propose a three-phase decomposition matheuristic for the Assembly Routing Problem and use it to solve the IRP as well. Metaheuristics have been proposed in Santos et al. (2016) and Alvarez et al. (2018). Santos et al. (2016) propose a multi-start iterated local search algorithm while Alvarez et al. (2018) propose an iterated local search and a simulated annealing algorithm. The latter algorithm shows a good trade-off between solution quality and computing time.

2.4. Our Contributions

The contributions of this paper are mainly three-fold:

1. We define a novel problem that has important implications in practice. In particular, our models, solution approaches, and managerial insights from the computational experiments are useful in warehouses with separate forward and reserve storage areas, sequential replenishment and picking cycles, and piece or carton picking in the forward storage area.

2. We exploit the special graph structure of the warehouse layout and the tractability of routing on this graph by adapting and modifying models and solution approaches from the IRP literature to incorporate this structure. To the best of our knowledge, this is the first study to define an extension of the IRP in a warehouse environment.

3. We present the first complexity results for the problem. These results not only show that the problem is unlikely to be polynomially solvable for practical cases, but also shed light into the complexity of similar problems in the order picking literature.

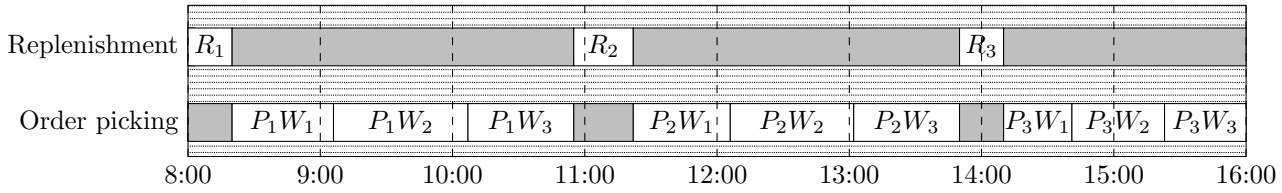

Figure 2: An example 8-hour shift consisting of 3 replenishment and order picking cycles, with each picking cycle consisting of 3 waves; R_i and P_iW_j denote the i^{th} replenishment cycle and the j^{th} wave of the i^{th} picking cycle, respectively

3. Problem Definition and Mathematical Model

In this section, we first define the problem environment for the SRRP, followed by our modeling assumptions and mixed integer programming model. In the last part, we present a number of complexity results for various cases of the SRRP.

3.1. Problem Definition

The SRRP is motivated by the operations of a retail distribution center (DC) that serves 76 retail stores of a large chain in the south west region of the United Kingdom. The DC applies a zone picking policy, with the zones determined by the types of products. Each zone consists of a forward storage area that applies manual picker-to-parts order picking using carts. These zones are replenished from a common reserve storage area. With a few exceptions, almost all zones apply *wave picking*, where the orders for a specific subset of stores are picked in pre-determined time intervals (called *waves*, lasting between 30 minutes and an hour). Due to the intensity of activities in a pick wave, replenishment and order picking are carried out at separate time intervals to avoid congestion. Usually, 3 to 5 successive pick waves (which we will call a *pick cycle*) are preceded by a replenishment cycle, which lasts between 15 and 30 minutes. An 8-hour shift generally consists of three replenishment-order picking cycles, as exemplified in Figure 2.

Each pick zone in the DC applies a continuous inventory review of the item slots by means of a warehouse management system (WMS). A single replenishment worker is assigned to each zone, who receives the list and quantities of items to be replenished in the next cycle from the WMS, and loads these items onto a material handling train while the previous pick cycle is in progress. Once the pick cycle is over, the replenishment worker visits the slots for these items and performs the replenishment within the allotted time limit. The current strategy is to replenish items whose quantity in the forward storage area falls below a predetermined minimum level. In case the allotted time is not sufficient to carry out all necessary replenishments, an additional replenishment cycle is set up at the end of the shift to replenish remaining item slots. On the other hand, such an additional cycle requires overtime, and hence the management would like to avoid this as much as possible. In this study, we ensure that stockouts can be avoided by including constraints in the mathematical model that all demand should be satisfied within a regular replenishment cycle prior to the corresponding pick cycle. Furthermore, using computational experiments, we compare our proposed replenishment approach to that implemented by the DC and quantify the potential improvements in replenishment efficiency with the proposed approach.

Given the limited time for replenishment, devising an efficient route for the replenishment worker results in the opportunity to replenish more items within the cycle, as well as to finish the cycle in shorter time, possibly allowing more time to perform the picking activities. Motivated by this, we define the *storage replenishment routing problem*, which aims to address the decisions on (i) when to replenish each item slot in the forward storage area from reserve storage to guarantee availability when demand arises and (ii) the replenishment route in each cycle, subject to the load capacity of the replenishment vehicle, time limit for the replenishment cycles, quantities of item arrivals to the reserve storage area, and the upcoming demand throughout the planning horizon. The objective of the SRRP is to minimize the total replenishment time.

Under the given problem settings, the SRRP resembles the well-known inventory routing problem, where the suppliers and retailers in the IRP correspond to the items in the reserve storage area and item slots in the forward storage area, respectively. The demand in the IRP is represented by the list of items to be picked in each order picking cycle in the SRRP. There are three main differences between the two problems. The first is the existence of multiple items in the SRRP, as opposed to a single item in the IRP, whereas this added complexity is offset by the fact that each item is demanded by a single item slot. Second, the inventory holding cost, which is an important component of the IRP, does not exist in the SRRP. The third difference is that as mentioned before, whereas the main challenge in solving the IRP arises from the routing aspect, this is an easy component of the SRRP, due to the special structure of the warehouse layout.

3.2. A Mixed Integer Programming Model for the SRRP

The mathematical model and solution approaches for the SRRP rely on a number of assumptions. First, we assume that the forward storage area has a single-block parallel-aisle layout, as in Figure 1, and that a single worker performs the replenishment cycle. These two assumptions are in line with the operations of the DC by which this study is motivated. Furthermore, our work can be extended to the case of multiple pickers, multiple blocks and non-traditional layout designs for which exact or heuristic approaches are available (see Section 2.2 for such approaches). The forward storage area is assumed to apply a dedicated storage policy, in that items are always stored in the same slots. Daily arrivals and demand of items for the planning horizon are assumed to be known in advance, which is applicable to cases where the demand from customers is frozen for a fixed period of time. The storage capacities, initial item inventories, and the replenishment cycle time limits are also assumed to be given. An item slot is replenished up to the capacity level of the slot, which is in line with practice and called *top-off replenishment* in the industry. Lastly, we assume that the replenishment time of an item is not dependent on the replenishment amount.

The notation used throughout the mathematical model and the solution approach for the SRRP is provided in Table 1. The set M denotes the item slots in the forward storage area, whereas M' also includes the depot. Each slot $i \in M$ has a capacity of U_i item units and an initial inventory of I'_{i1} and I_{i1} in the reserve and forward storage areas, respectively. We use t_{ij} to denote the travel time between locations $i \in M'$ and $j \in M'$. If $j \in M$, then this includes the replenishment time of j as well. Replenishment and picking cycles are denoted by the set T. Before cycle $t \in T$ starts, p_{it} units of item $i \in M$ arrives at the reserve storage area. Demand for item

Table 1: Notation used throughout the mathematical model for the SRRP

Index sets	
M	Item slots
M'	All locations in the forward storage area, including the item slots and the depot; $M' = M \cup \{0\}$
T	Replenishment and picking cycles
Parameters	
t_{ij}	Travel time between $i \in M'$ and $j \in M'$ (includes replenishment time of j if $j \in M$)
p_{it}	Amount of item $i \in M$ arrival at the reserve storage before cycle $t \in T$ starts
r_{it}	Amount of item $i \in M$ to be picked in cycle $t \in T$
U_i	Capacity of item slot $i \in M$
L	Replenishment cycle time
Q	Load capacity of the replenishment vehicle
I_{i1}	Initial inventory of item slot $i \in M$ in the forward storage area
I'_{i1}	Initial inventory of item $i \in M$ in the reserve storage area
Decision Variables	
x_{it}	Amount of item $i \in M$ replenished in cycle $t \in T$
z_{it}	$= \begin{cases} 1, & \text{if } i \in M' \text{ is visited in replenishment cycle } t \in T \\ 0, & \text{otherwise} \end{cases}$
y^t_{ij}	$= \begin{cases} 1, & \text{if item slots } i \in M \text{ and } j \in M : j > i \text{ are visited consecutively in the route in cycle } t \in T \\ 0, & \text{otherwise} \end{cases}$
y^t_{0i}	Number of times the replenishment route travels between the depot and item slot $i \in M$
I_{it}	Inventory of item slot $i \in M$ in forward storage at the beginning of cycle $t \in T \cup \{\|T\| + 1\} : t \geq 2$
I'_{it}	Inventory of item $i \in M$ in reserve storage at the beginning of cycle $t \in T \cup \{\|T\| + 1\} : t \geq 2$

$i \in M$ during pick cycle $t \in T$ is r_{it}.

To represent routing decisions, variables y^t_{ij} indicate the number of times the link between depot/item slot $i \in M'$ and item slot $j \in M$, $j > i$, is traversed by the replenishment vehicle. More specifically, y^t_{0j} determine the number of times an item slot $j \in M$ is visited before or after the depot. These variables can take values 0, 1 or 2, the latter representing the case where the vehicle only replenishes item slot $j \in M'$ on its route. Variables y^t_{ij}, $i \geq 1$ can only take values of 0 or 1, as the link between two item slots never needs to be traversed more than once. Variables z_{it} denote whether location $i \in M'$ is visited in cycle $t \in T$, whereas replenishment amount for item $i \in M$ in cycle $t \in T$ is denoted by x_{it}. Starting from the second cycle, I'_{it} and I_{it} represent the inventory amount of item $i \in M$ in the reserve and forward storage areas at the beginning of cycle $t \in T$, respectively. Similarly, $I'_{i,|T|+1}$ and $I_{i,|T|+1}$ denote the inventory of each item in these areas at the end of the planning horizon.

Based on these sets, parameters and decision variables, the SRRP can be modeled using the following mixed integer program, which is a modification of the model for the IRP in Archetti et al. (2007):

$$\min \quad \sum_{i \in M'} \sum_{j \in M',\, j > i} \sum_{t \in T} t_{ij} y^t_{ij} \tag{1}$$

$$s.t. \quad I'_{i,t+1} = I'_{it} + p_{it} - x_{it} \qquad \forall i \in M,\, t \in T, \tag{2}$$

$$I_{i,t+1} = I_{it} + x_{it} - r_{it} \qquad \forall i \in M,\, t \in T, \tag{3}$$

$$x_{it} \geq U_i\, z_{it} - I_{it} \qquad \forall i \in M,\, t \in T, \tag{4}$$

$$x_{it} \leq U_i - I_{it} \qquad \forall i \in M,\, t \in T, \tag{5}$$

$$x_{it} \leq U_i\, z_{it} \qquad \forall i \in M,\, t \in T, \tag{6}$$

$$\sum_{i \in M'} \sum_{j \in M',\, j > i} t_{ij} y_{ij}^t \leq L \qquad \forall t \in T, \tag{7}$$

$$\sum_{i \in M'} x_{it} \leq Q \qquad \forall t \in T, \tag{8}$$

$$\sum_{j \in M',\, j > i} y_{ij}^t + \sum_{j \in M',\, j < i} y_{ji}^t = 2 z_{it} \qquad \forall i \in M',\, t \in T, \tag{9}$$

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S},\, j > i} y_{ij}^t \leq \sum_{i \in \mathcal{S}} z_{it} - z_{kt} \qquad \forall \mathcal{S} \subseteq M,\, t \in T,\, k \in \mathcal{S}, \tag{10}$$

$$I_{it} \geq (1 - z_{it})\, r_{it} \qquad \forall i \in M,\, t \in T, \tag{11}$$

$$I_{i,t-k} \geq \left(\sum_{j=0}^{k} r_{i,t-j} \right) \left(1 - \sum_{j=0}^{k} z_{i,t-j} \right) \qquad \forall i \in M,\, t \in T,\, k \in \{0, 1, \ldots, t-1\}, \tag{12}$$

$$z_{it} \leq z_{0t} \qquad \forall i \in M,\, t \in T, \tag{13}$$

$$y_{ij}^t \leq z_{it} \qquad \forall i \in M,\, j \in M,\, i < j,\, t \in T, \tag{14}$$

$$y_{0j}^t \leq 2 z_{jt} \qquad \forall j \in M,\, t \in T, \tag{15}$$

$$y_{ij}^t \in \{0, 1\} \qquad \forall i \in M,\, j \in M,\, i < j,\, t \in T, \tag{16}$$

$$y_{0j}^t \in \{0, 1, 2\} \qquad \forall j \in M,\, t \in T, \tag{17}$$

$$z_{it} \in \{0, 1\} \qquad \forall i \in M',\, t \in T, \tag{18}$$

$$I_{it},\, I'_{it} \geq 0 \qquad \forall i \in M,\, t \in T \cup \{|T| + 1\} : t \geq 2, \tag{19}$$

$$x_{it} \geq 0 \qquad \forall i \in M',\, t \in T. \tag{20}$$

Objective function (1) of the model defines the total replenishment time to be minimized. Constraints (2) define inventory balance at the reserve area and constraints (3) determine the inventory balance at each item slot. Constraints (4)-(6) ensure that whenever a slot is replenished, it is replenished up to its capacity. Constraints (7) and (8) stipulate the replenishment time limit and load capacity of the replenishment vehicle in each cycle, respectively. Constraints (9) ensure that if an item is replenished, it has a predecessor and a successor on the route. Constraints (10) eliminate subtours and constraints (11)-(15) are valid inequalities to strengthen the formulation (Archetti et al., 2007). Constraints (11) enforce sufficient inventory to satisfy picking demand of an item in a cycle where there is no replenishment for it. With constraints (12), it is ensured that if an item is not replenished in cycles $t-k, t-k+1, \ldots, t$, then its inventory at the beginning of cycle $t-k$ should be at least equal to the total picking demand in these cycles. Constraints (13) imply that an item cannot be replenished in a cycle unless the vehicle starts from the depot, whereas constraints (14) and (15) require the item slots and the depot to be visited if they are included in the replenishment route. Constraints (16)-(20) define the domain of the decision variables.

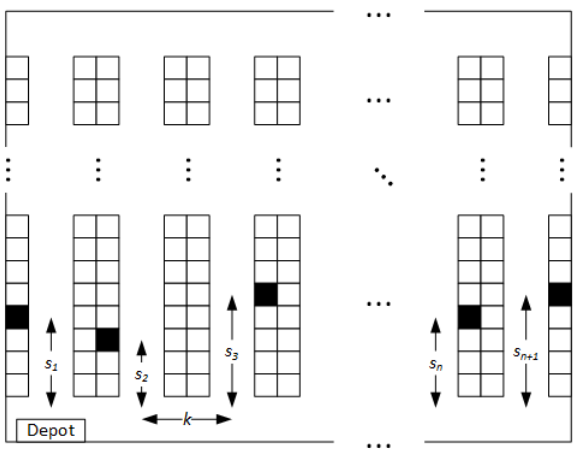

Figure 3: The SRRP instance required for transformation from the PARTITION problem

3.3. Computational Complexity of the SRRP

In this section, we analyze the computational complexity of the SRRP for varying number of replenishment cycles and present the first complexity results on the problem. In addition to establishing the complexity of various cases of the SRRP, corollaries on our results lead to conclusions on the complexity of the picker routing problem with multiple pickers in single-block parallel-aisle warehouses. We start with the case where $|T| = 1$.

Theorem 1. *Whenever feasible, the SRRP is polynomially solvable for a single replenishment cycle.*

Proof. The optimal solution of the SRRP with $|T| = 1$ is to replenish the items whose initial inventories in the forward storage area are insufficient to fulfill the amounts to be picked in the upcoming cycle. Once these items are determined, the problem is identical to solving the picker routing problem in a single-block parallel-aisle warehouse, which is polynomially solvable (Ratliff & Rosenthal, 1983). □

Next, we establish the complexity of the case where $|T| = 2$.

Theorem 2. *The SRRP is $\mathcal{NP}$-hard for two replenishment cycles, even with no load capacity.*

Proof. We first show the $\mathcal{NP}$-hardness of this case by transformation from PARTITION (a weakly $\mathcal{NP}$-hard problem due to Garey & Johnson, 1979), which, given a set A and an integer size s_i for each $i \in A$, seeks a subset $A' \subseteq A$ such that the total size in A' equals the total size in $A \setminus A'$, i.e., $\sum_{i \in A'} s_i = \sum_{i \in A \setminus A'} s_i$.

The obtain the SRRP instance that yields a PARTITION instance, we use the warehouse instance in Figure 3. Here, all item slots can store a single unit of the item and all initial inventories at the forward storage area are zero. There are $n+1$ aisles, with one item on each of aisles $1, 2, \ldots, n$ to be picked in the second cycle, and one item on aisle $n+1$ to be picked in both cycles. Each item i is located s_i time units from the front end of the aisle, and the distance between two consecutive aisles is k time units. The replenishment cycle has a time limit of $2(s_{n+1} + nk) + \sum_{i=1}^{n} s_i$ time units.

Given that item $n+1$ will be replenished on both cycles, this results in a travel time of $2s_{n+1}+nk$ time units on each cycle. Each remaining item i takes $2s_i$ time units to be replenished (to traverse the aisle until the item and return the same distance). Hence, the resulting SRRP instance has a feasible solution only if we can allocate the remaining $2\sum_{i=1}^{n} s_i$ time units into two equal-length cycles T_1 and T_2, so that $\sum_{i\in T_1} 2s_i = \sum_{i\in T_2} 2s_i = \sum_{i=1}^{n} s_i$. This is equivalent to solving the PARTITION instance. Hence, unless $\mathcal{P} = \mathcal{NP}$, the SRRP can be solved in pseudo-polynomial time in the best case. □

With the next theorem, we show that when the replenishment vehicle has no load capacity, the SRRP with two cycles can be solved in pseudo-polynomial time.

Theorem 3. *Without vehicle load capacity, the SRRP is weakly $\mathcal{NP}$-hard for $|T| = 2$.*

Proof. Theorem 2 establishes the $\mathcal{NP}$-hardness of this case. To show weak $\mathcal{NP}$-hardness, we provide a pseudo-polynomial-time algorithm to solve the uncapacitated SRRP with $|T| = 2$ in Appendix A. This algorithm extends the dynamic programming approach by Ratliff & Rosenthal (1983) for the picker routing problem.

The number of states in the algorithm does not depend on the number of item slots, but is linear in terms of the replenishment time limit. Due to this component, it is polynomial only in terms of the unary representation of the inputs, which implies that it can be solved in pseudo-polynomial time, proving the weak $\mathcal{NP}$-hardness of the SRRP with $|T| = 2$ and no replenishment vehicle load capacity. □

It should be noted here that with load capacities, the algorithm in Theorem 3 would no longer run in pseudo-polynomial-time. This is because in this case, the move types in Ratliff & Rosenthal (1983) would not be applicable, requiring to check all possible combinations of the items in each aisle, deeming the algorithm to be exponential-time in terms of the number of items.

Theorems 2 and 3 lead to an important result for the picker routing problem with two pickers.

Corollary 1. *The picker routing problem with two capacitated pickers on a single-block parallel-aisle warehouse is $\mathcal{NP}$-hard. It is weakly $\mathcal{NP}$-hard when the pickers do not have a load capacity.*

Next, we show the complexity result for the remaining cases.

Theorem 4. *The SRRP is strongly $\mathcal{NP}$-hard for $|T| \geq 3$, even without a load capacity.*

Proof. We show the $\mathcal{NP}$-hardness of this case by transformation from 3-PARTITION (a strongly $\mathcal{NP}$-hard problem due to Garey & Johnson, 1979), where there exist a set A of $3m$ elements, an integer bound B, a size s_i for all items $i \in A$ such that $\frac{B}{4} < s_i < \frac{B}{2}$ for all $i \in A$, and $\sum_{i\in A} s_i = mB$. The 3-PARTITION problem has a solution if there exists a partition of A into m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{i\in A_j} s_i = B$ for all $j = 1, 2, \ldots, m$.

The required SRRP instance for transformation is similar to the one in Figure 3, where each item slot has unit capacity and no initial inventory is available. There are m replenishment cycles and the forward storage area has $3m + 1$ aisles. Aisles 1 through $3m$ have one item to be picked in the last cycle, whereas aisle $3m + 1$ has one item to be picked in all m cycles. Each item on aisle i is located s_i time units from the front cross aisle,

where $\frac{B}{8} < s_i < \frac{B}{4}$ for all $i \in M$ and $\sum_{i=1}^{3m} s_i = \frac{B}{2}$. Travel time between two consecutive aisles is k units. The replenishment cycle time limit is $2\left((3m+1)\,k + s_{3m+1}\right) + B$.

Item $3m+1$ needs to be replenished in each cycle, taking $2\left((3m+1)\,k + s_{3m+1}\right)$ time units. This leaves B time units on each cycle to replenish the remaining $3m$ items, each of which takes $2s_i$ time units. The SRRP instance has a solution if and only if we can allocate the remaining $2\sum_{i=1}^{3m} s_i = mB$ time units into cycles $T_1, T_2, \ldots T_m$, where $\frac{B}{4} \leq 2s_i \leq \frac{B}{2}$ for all $i \in M$, so that $\sum_{i \in T_1} 2s_i = \sum_{i \in T_2} 2s_i = \ldots = \sum_{i \in T_m} 2s_i = B$. Since this yields the equivalent 3-PARTITION instance, a feasible SRRP solution can be found if and only if the corresponding 3-PARTITION instance has a feasible solution. Since the 3-PARTITION problem is strongly $\mathcal{NP}$-hard, so is the SRRP with at least three replenishment cycles. □

Theorem 3 results in an important corollary for picker routing with at least three pickers.

Corollary 2. *The picker routing problem with more than two pickers is strongly $\mathcal{NP}$-hard, even when the pickers have no load capacity.*

4. An A Priori Route-Based Heuristic

The $\mathcal{NP}$-completeness of the SRRP suggests that as the instance size increases, the computational burden to optimally solve it will substantially increase. This is also justified by the computational results in Section 5, where the mathematical model can be solved to optimality for only small instances up to 50 items and 15 replenishment cycles. To overcome this, we propose an a priori route-based heuristic in this section, which modifies the a priori routing heuristic for the IRP by Solyalı & Süral (2011). The motivation behind this is that once the item slots to replenish are fixed, the routing problem is tractable.

In the first step of the heuristic, the routing problem corresponding to all the items to be picked throughout the planning horizon is solved. To do so, one may use the exact approach by Ratliff & Rosenthal (1983), or any of the heuristics for the single-block picker routing problem in Section 2.2. The ability to solve the a priori routing problem in a fast manner differentiates our work from that by Solyalı & Süral (2011), where the routing subproblem involves solving the TSP on a general graph using a specialized solver. Once the a priori route is determined, the sequence of items to be visited in each replenishment trip is fixed. For any subset of items to replenish in any cycle, the items are sequenced in the replenishment cycle by maintaining their order in the a priori route. The main aim in fixing the sequences of item replenishment is to simplify the routing decisions in the next step. An example of using the a priori route for determining the replenishment sequence is given in Figure 4(a), which provides the optimal tour that visits all the items in Figure 1. If the next replenishment cycle involves the seven items shown in black, Figure 4(b) shows the resulting route that is formed by fixing the sequence of item visits in Figure 4(a).

The first step of the heuristic determines the replenishment route that fixes the sequence with which items slots will be replenished. This leaves the decision of which item slots to replenish. It should be noted here that due to the order-up-to replenishment policy, the amounts to replenish are endogenously determined by which

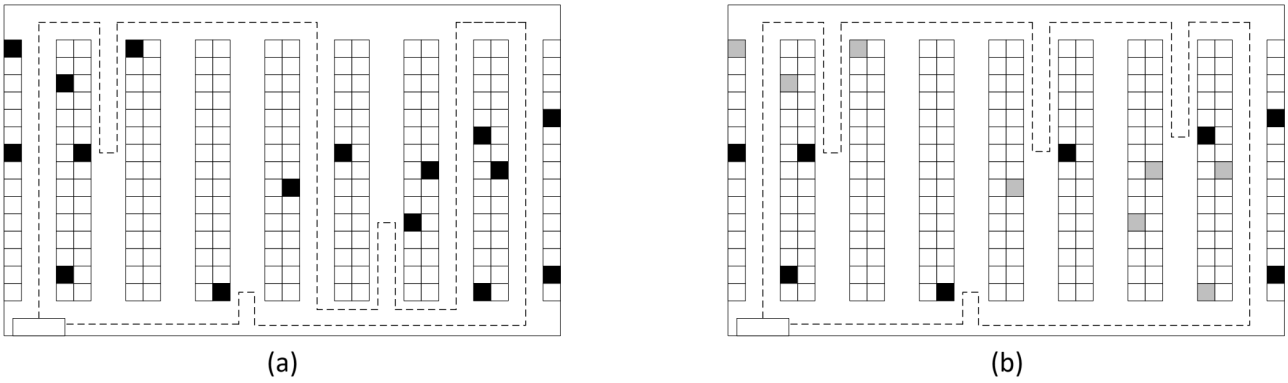

Figure 4: (a) The optimal route for the example in Figure 1, and (b) the resulting route when a subset of eight items (shown in black) are to be replenished

Table 2: Additional notation for the a priori routing heuristic

Index sets	
T'	$T \cup \{\|T\| + 1\}$
T^*	$T \cup \{0\}$
α_i	All predecessors of $i \in M'$ on the a priori route
β_i	All successors of $i \in M'$ on the a priori route
Parameters	
b_{ikt}	Amount to be replenished for item slot $i \in M$ in cycle $t \in T'$ if the last replenishment was made in cycle $k \in T^*$
π_{it}	The earliest cycle for item $i \in M$ in which a replenishment can satisfy the demand from cycle π_{it} to cycle $t \in T$
μ_{it}	The latest cycle for item $i \in M$ for which a replenishment in cycle $k \in T$ can satisfy the demand up to cycle μ_{it}
Decision variables	
w_{ikt}	Binary variable indicating whether stock location $i \in M$ is replenished in cycle $t \in T'$, given that the last replenishment was made in cycle k, $\pi_{it} \leq k \leq t-1$
$\hat{y}_{ij}^t$	$= \begin{cases} 1, & \text{if } j \in M' \text{ follows } i \in M' \text{ in the replenishment route in cycle } t \in T \\ 0, & \text{otherwise} \end{cases}$

cycles the item slot will be replenished. To determine the cycles in which each item slot will be replenished, we extend the strong formulation for the IRP, which was first put forward by Pınar & Süral (2006).

The additional notation we use for the a priori routing heuristic is given in Table 2. The index sets T' and T^* represent extended sets $T \cup \{|T| + 1\}$ and $T \cup \{0\}$, respectively. The predecessor and successor sets for location $i \in M'$ are denoted by α_i and β_i, respectively. To incorporate the strong formulation, we make use of additional parameters. The parameter b_{ikt} denotes the amount to be sent to item slot $i \in M$ in cycle $t \in T'$ if the last replenishment was made in cycle $k \in T^*$. More formally,

$$b_{i0t} = U_i - I_{i1} + \sum_{j=1}^{t-1} r_{ij} \text{ and } b_{ikt} = \sum_{j=k}^{t-1} r_{ij} \text{ for all } k \in T.$$

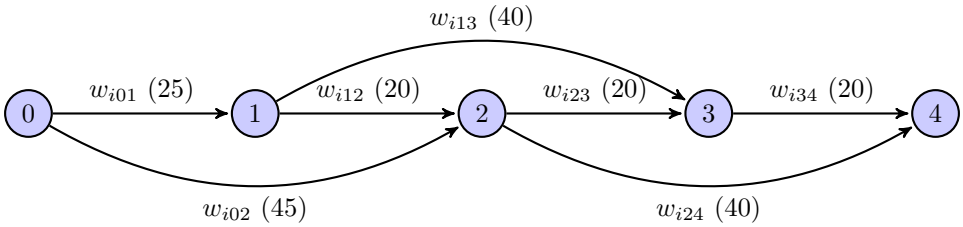

Figure 5: The shortest path network corresponding to the given example, with the b_{ikt} values given in brackets

For $k = 0$, the amount b_{ikt} should increase the initial inventory level I_{i1} for item slot $i \in M$ to the capacity level U_i and needs to satisfy the demand for the first $t-1$ pick cycles. For $k \geq 1$, b_{ikt} needs to satisfy the demand of pick cycles between k and $t-1$.

To guarantee that none of the item slots will stock out during the planning horizon, two additional parameters are used: (i) π_{it} is the earliest cycle for item slot $i \in M$ for which a replenishment in cycle $\pi_{it} \in T^*$ can satisfy the demand of order picking cycles $\pi_{it}+1, \pi_{it}+2, \ldots, t-1$ until the next replenishment in cycle $t \in T$, (ii) μ_{it} is the latest cycle for item slot $i \in M$ for which a replenishment in cycle $t \in T$ can satisfy the demand in picking cycles $t+1, t+2, \ldots, \mu_{it}-1$ until the next replenishment in cycle $\mu_{it} \in T$. In mathematical terms:

$$\pi_{it} = \min_{0 \leq k \leq t-1} \{k : b_{ikt} \leq U_i\} \text{ and } \mu_{it} = \max_{t+1 \leq k \leq |T|+1} \{k : b_{itk} \leq U_i\}.$$

We use decision variables w_{ikt} to denote whether item slot $i \in M$ is replenished in cycle $t \in T'$, after the last replenishment was made in cycle $k \in T$, where $\pi_{it} \leq k \leq t-1$. Here, w_{i0t} denotes whether the first replenishment for item slot $i \in M$ is in cycle $t \in T$ and $w_{ik,|T|+1}$ indicates whether the last replenishment for item $i \in M$ is in cycle $k \in T$. The replenishment scheme for item slot $i \in M$, when modeled with decision variables w_{ikt}, constitutes a shortest path network. As an example, consider an item with an initial inventory of 20 units, item slot capacity of 45 units, and a constant amount of 20 units is picked in every cycle. For a horizon of three replenishment-picking cycles, this yields the network given in Figure 5.

Lastly, we change the definition of the routing variables y_{ij}^t to a directed version by defining a binary variable $\hat{y}_{ij}^t$, which takes a value of 1 only if the replenishment vehicle visits location $j \in M'$ immediately after visiting $i \in M'$ on its route in cycle $t \in T$.

Using the given index sets, parameters, and decision variables, the following mixed integer programming model determines the set of item slots to replenish in each cycle:

$$\min \quad \sum_{i \in M'} \sum_{j \in M'} \sum_{t \in T} t_{ij} \hat{y}_{ij}^t \tag{21}$$

$$s.t. \quad I'_{i,t+1} = I'_{it} + p_{it} - \sum_{k=\pi_{it}}^{t-1} b_{ikt}\, w_{ikt} \quad \forall i \in M,\, t \in T, \tag{22}$$

$$I'_{it} \geq \sum_{k=\pi_{it}}^{t-1} b_{ikt}\, w_{ikt} \quad \forall i \in M,\, t \in T, \tag{23}$$

$$I_{i,t+1} = I_{it} + \sum_{k=\pi_{it}}^{t-1} b_{ikt}\, w_{ikt} - r_{it} \quad \forall i \in M,\, t \in T, \tag{24}$$

$$\sum_{k=1}^{\mu_{i,0}} w_{i0k} = 1 \quad \forall i \in M, \tag{25}$$

$$\sum_{k=t+1}^{\mu_{it}} w_{itk} - \sum_{k=\pi_{it}}^{t-1} w_{ikt} = 0 \quad \forall i \in M,\, t \in T, \tag{26}$$

$$\sum_{k=\pi_{it}}^{|T|} w_{i,k,|T|+1} = 1 \quad \forall i \in M, \tag{27}$$

$$\sum_{k=\pi_{it}}^{t-1} w_{ikt} = z_{it} \quad \forall i \in M,\, t \in T, \tag{28}$$

$$z_{it} \leq z_{0t} \quad \forall i \in M,\, t \in T. \tag{29}$$

$$\sum_{j \in \beta_i} \hat{y}_{ij}^t = z_{it} \quad \forall i \in M',\, t \in T, \tag{30}$$

$$\sum_{j \in \alpha_i} \hat{y}_{ji}^t = z_{it} \quad \forall i \in M',\, t \in T, \tag{31}$$

$$\sum_{i \in M'} \sum_{j \in M'} t_{ij} \hat{y}_{ij}^t \leq L \quad \forall t \in T, \tag{32}$$

$$\sum_{i \in M} \sum_{k=\pi_{it}}^{t-1} b_{ikt} w_{ikt} \leq Q \quad \forall t \in T, \tag{33}$$

$$w_{ikt} \in \{0,1\} \quad \forall i \in M,\, t \in T', \pi_{it} \leq k \leq t-1, \tag{34}$$

$$\hat{y}_{ij}^t \in \{0,1\} \quad \forall i \in M',\, j \in M', t \in T, \tag{35}$$

$$z_{it} \in \{0,1\} \quad \forall i \in M',\, t \in T, \tag{36}$$

$$I_{it},\, I'_{it} \geq 0 \quad \forall i \in M,\, t \in T' : t \geq 2. \tag{37}$$

Objective function (21) minimizes the total replenishment time. Constraints (22) and (23) impose inventory balance at the reserve storage area, whereas constraints (24) stipulate inventory balance at the item slots. Constraints (25)-(27) are the flow balance constraints for the resulting shortest path network, whereas constraints (28) ensure that an outgoing arc in the shortest path network results in the corresponding binary replenishment variable to be equal to one. Using constraints (29), we impose the inclusion of the depot in each replenishment tour, while constraints (30) and (31) connect the routing and binary replenishment variables, and impose the tour precedence relations. Replenishment cycle time and load capacity limits are set by constraints (32) and (33), respectively. Constraints (34) through (37) indicate the domains for the decision variables.

Using the results of the model, the set of item slots to be replenished, the amounts to replenish, and the replenishment route can be determined. The total replenishment time of this heuristic is denoted by H throughout our computational experiments. Given the items to be replenished in each cycle, the routes can be further improved by re-solving the picker routing algorithm by Ratliff & Rosenthal (1983), which we call as the *a posteriori routing step*. This improved replenishment time will be denoted by H^+.

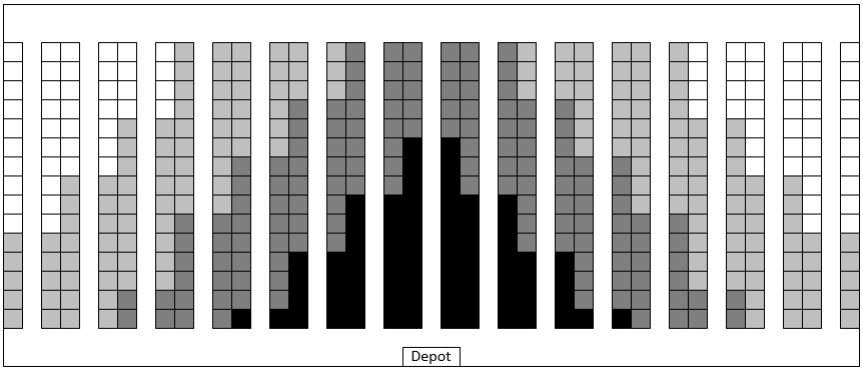

Figure 6: Relative convenience of item slots in a single-block parallel-aisle warehouse with a depot in the middle, where darker colors represent more convenience

5. Computational Experiments

To assess the performance of the heuristics, effect of problem parameters, and practical impact of our approaches, we perform computational experiments on two sets of instances, distinguished by their size and objectives. These instances are available online at https://researchdata.bath.ac.uk/id/eprint/976 and the details of the instance settings are provided in Appendices B.1 and B.2 in the Online Supplement.

With the smaller instances, our objectives are three-fold: First, we would like to compare the results of the a priori routing heuristic to optimal solutions and quantify the trade-off between the sacrifice in objective values and the savings in computational times. Second, we aim to compare various a priori routing approaches among each other to assess if there is any statistically significant difference between these methods in terms of the resulting replenishment times. Lastly, we would like to analyze the extent of improvement in the objective function brought about by the a posteriori routing step.

In most warehouses, items are of varying popularity in terms of demand. In such cases, more popular items may be stored in more convenient slots (i.e., those closer to the depot) to improve the efficiencies in picking and replenishment routes in the existence of skewed demand (an example showing the relative convenience of item slots in a single-block parallel-aisle warehouse is provided in Figure 6). With the larger instances, we vary the demand skewness and aim to assess whether changes in skewness levels have any significant effect on the replenishment times. Our second objective with the larger instances is to compare the integrated replenishment and routing approach to that implemented by the DC, as well as to benchmarks in the industry and literature that ignore the routing decisions.

Our heuristics are coded in C++ and implemented on a personal computer with Intel Core i7-6600U 2.60 GHz processor and 8 GB RAM. To obtain the optimal solutions, we solve the formulation presented in Section 2.3 using the branch-and-cut algorithm presented in Archetti et al. (2007). The algorithm is implemented in C++ CPLEX 12.5 was used as MIP solver. The experiments on the branch-and-cut algorithm are run on a laptop with Intel Core i7-3687U 2.10 GHz processor and 8 GB RAM. We impose a 2-hour CPU time limit on each run of CPLEX and the heuristics.

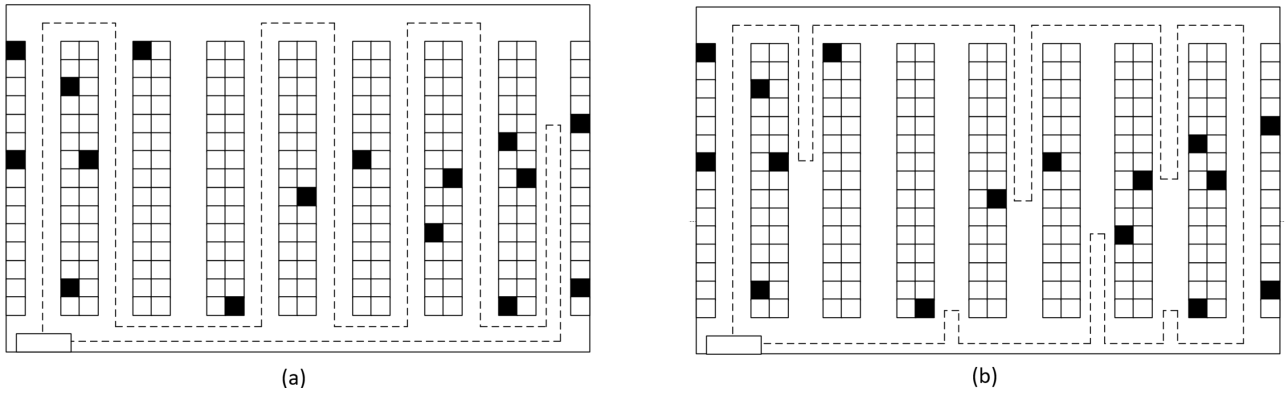

Figure 7: (a) The S-shape and (b) largest gap heuristic solutions for the instance in Figure 1

5.1. Experiments with Smaller Instances

To evaluate heuristic performance, we generate a smaller set of instances derived from Archetti et al. (2007) and Solyalı & Süral (2011). The warehouse parameters mainly depend on Roodbergen & de Koster (2001a). We consider a single-block forward storage area consisting of 30 parallel-aisles, with 30 equal-sized item slots on each side of an aisle. The lengths between two consecutive item slots and two consecutive aisles are 1 and 2.5 meters, respectively. As in Roodbergen & de Koster (2001a), we assume no additional time for acceleration, deceleration, turning, or entering/exiting an aisle. Travel speed of the replenishment vehicle is 2 meters per second on average and it takes 30 seconds to replenish an item slot. The depot (entrance and exit to the forward storage area) is located on the left end of the front cross aisle.

Each day consists of three replenishment picking cycles. We generate instances ranging from 1 to 5 days, hence from $|T| = 3$ to 15 replenishment cycles in increments of 3. We also vary the number of items $|M|$ from 25 to 150, in increments of 25. The locations of these items in the warehouse are randomly generated according to a uniform distribution. With 10 instances under five possible number of cycles and six possible number of items, we experiment on a total of 300 instances. Our instances are labeled as i##-t##-##, where the dash-separated numbers represent the number of items, number of cycles, and instance number, respectively.

We assume that an item is demanded in every picking cycle in varying amounts over the planning horizon. The first-cycle demand r_{i1} of item i is uniformly distributed from 10 to 90. For the remaining cycles, the demand r_{it} is uniformly distributed between 0 and $2r_{i1}$. Capacity of item slot U_i for item i is given by $\delta_i r_{i1}$, where δ_i is randomly generated from 2, 3, or 4. The initial inventory I_{i1} is given by $U_i - r_{i1}$. New items arrive at the reserve storage area at the beginning of each day (hence once every three cycles) at a quantity of $5r_{i1}$. Assuming one unit of each item weighs equally, load capacity of the replenishment vehicle is $60|M|$ units. Travel time limit for each cycle is 500 seconds for $|M| = \{25, 50\}$, 750 seconds for $|M| = \{75, 100\}$, and 1000 seconds for $|M| = \{125, 150\}$.

We use three alternative approaches to generate the a priori route for each instance. The first of these is the optimal route, determined using the dynamic programming-based algorithm by Ratliff & Rosenthal (1983), which will be abbreviated as *RR* throughout this section. The remaining two methods are the *S-shape* and *largest gap* heuristics by Hall (1993). The S-shape route starts from the left-most non-empty aisle and visits

each non-empty aisle in sequence, traversing each aisle completely, and returning to the depot using the front cross aisle. When there are an odd number of non-empty aisles, it traverses the last non-empty aisle until the last item. For each non-empty aisle, the largest gap heuristic compares the maximum of (1) the maximum travel time between two consecutive items in the list, (2) the travel time between the front-most item and the front end of the aisle, and (3) that between the back-most item and the back end of the aisle. After traversing the left-most non-empty aisle completely, aisles for which (1) or (2) are maximum are entered from the back end and traversed until the largest gap, in the order from left to right. The right-most non-empty aisle is traversed completely, and aisles for which (1) or (3) are maximum are entered from the front and traversed until the largest gap, in reverse order. The S-shape and largest gap heuristic routes for the example in Figure 1 are given in Figure 7(a) and (b), respectively. We run the a priori routing heuristic with these three approaches and report the results with and without the a posteriori routing step.

To evaluate the performance of each heuristic, we use a number of gap values. In cases where the optimal solution can be found, the *gap to optimum* ($\%Opt(.)$) is calculated as:

$$\%Opt(.) = \frac{z(.) - z_{opt}}{z_{opt}} \times 100\%,$$

where $z(.)$ denotes the heuristic replenishment time (we use $z(H)$ and $z(H^+)$ for the cases without and with improvement, respectively) and z_{opt} refers to the optimal replenishment time. When CPLEX cannot find the optimal solution within the time limit, we use *CPLEX gap* ($\%C_{gap}(.)$) as the performance measure, where we replace z_{opt} with the CPLEX upper bound. When CPLEX cannot find a feasible solution, the RR^+ *gap* ($RR^+(.)$), where z_{opt} is replaced by $z(H^+)$ for RR with a posteriori routing, is used to evaluate heuristic performance.

Within a time limit of two hours, CPLEX can solve 40 out of 300 instances to optimality. A comparison of the CPU times and optimality gaps for these instances, which involve 3 to 12 cycles with 25 items and 3 cycles with 50 items, is given in Table 3. On average, CPLEX requires around 25 minutes to find the optimal replenishment tours. For the heuristics, the average CPU times are 1.34, 1.58, and 1.81 seconds for RR, S-shape, and largest gap a priori tours, respectively (it should be noted here that since the a posteriori routing step takes less than 0.05 seconds for each instance, we report a single CPU time for the heuristics). With a posteriori routing, the average optimality gaps for the three routing approaches are 1.22%, 3.74%, and 1.05%, respectively. This shows that regardless of the a priori routing approach chosen, the proposed approach provides near-optimal results with more than 99.8% savings in computational time, justifying its efficiency.

When the analysis is extended to all instances for which CPLEX can find a feasible solution, the average CPLEX gaps and the CPU times are presented in Table 4. These instances include all 25-item instances and 19 of the 50-item instances. For these, improved versions of the RR and largest gap heuristics both result in a gap of 0.35% each. Furthermore, of the 29 instances for which CPLEX cannot find an optimal solution, these heuristics find a better solution in eight instances each. A paired t-test between the CPLEX results and the heuristics (presented in Appendix C.1 in the Supplementary Material) shows that for a confidence level of 95%,

Table 3: Percent gaps and CPU times (in seconds) of CPLEX and a priori heuristics for instances that can be solved to optimality, with the best gap for each instance shown in bold

	CPLEX	RR			S-shape			Largest gap		
Instance	CPU time	$\%Opt(H)$	$\%Opt(H^+)$	CPU time	$\%Opt(H)$	$\%Opt(H^+)$	CPU time	$\%Opt(H)$	$\%Opt(H^+)$	CPU time
i25-t03-01	15.44	2.19	2.19	0.86	14.58	**1.70**	0.25	2.19	2.19	0.87
i25-t03-02	6.55	2.72	0.54	0.24	7.83	**0.00**	0.17	2.17	**0.00**	0.16
i25-t03-03	27.43	2.25	**0.00**	0.63	11.64	4.76	0.44	1.98	1.59	0.64
i25-t03-04	20.10	**0.38**	**0.38**	0.22	6.01	3.25	0.14	0.75	0.75	0.66
i25-t03-05	10.77	2.02	**1.30**	0.45	18.59	5.04	0.38	3.89	2.31	0.78
i25-t03-06	5.13	1.57	**0.00**	0.45	10.85	4.47	0.31	7.38	2.24	0.22
i25-t03-07	6.94	2.01	**0.00**	0.21	17.55	2.88	0.12	5.32	0.43	0.27
i25-t03-08	3.12	1.56	**0.00**	0.17	14.35	1.25	0.17	1.56	**0.00**	0.19
i25-t03-09	2.85	3.80	1.77	0.14	12.17	5.20	0.17	1.65	**0.51**	0.13
i25-t03-10	15.46	4.53	4.25	0.44	8.09	3.43	0.13	5.21	**1.65**	0.19
i25-t06-01	288.68	0.66	**0.06**	0.50	7.30	1.05	1.22	1.99	0.33	1.45
i25-t06-02	72.74	0.06	**0.00**	0.27	10.54	0.75	0.61	0.06	**0.00**	0.29
i25-t06-03	104.43	0.65	**0.26**	0.52	13.43	2.42	0.50	2.62	0.65	0.52
i25-t06-04	298.93	4.23	1.66	2.19	5.84	3.37	1.95	2.25	**0.32**	1.76
i25-t06-05	155.12	2.23	**0.36**	0.38	16.36	2.91	0.70	7.84	**0.36**	1.09
i25-t06-06	1210.55	1.18	**0.56**	5.44	10.99	1.43	0.77	3.66	**0.56**	7.41
i25-t06-07	59.56	**1.35**	**1.35**	1.42	13.13	1.50	0.81	2.55	1.73	0.98
i25-t06-08	315.95	4.88	**0.11**	0.97	7.64	1.41	1.86	4.60	2.22	1.57
i25-t06-09	733.35	3.24	**1.44**	0.98	21.21	3.88	0.57	3.40	**1.44**	0.72
i25-t06-10	159.61	1.93	**1.55**	2.64	8.90	2.84	0.22	7.74	3.22	1.36
i25-t09-01	4114.76	1.39	0.45	1.55	7.43	1.88	1.24	3.45	**0.19**	2.39
i25-t09-02	5174.85	2.12	1.33	0.91	9.95	1.62	1.13	2.12	**0.90**	0.90
i25-t09-03	1883.73	**0.00**	**0.00**	2.11	13.40	1.40	2.89	4.28	0.37	3.46
i25-t09-04	2103.55	5.62	**1.08**	1.59	17.74	4.65	1.71	4.83	1.93	2.25
i25-t09-05	4568.44	3.14	**1.72**	3.85	16.58	2.93	1.84	3.39	1.97	2.38
i25-t09-06	1300.24	**0.80**	**0.80**	0.45	14.94	4.53	2.07	**0.80**	**0.80**	0.63
i25-t09-07	5080.23	2.11	0.62	5.70	11.00	2.02	17.66	2.42	**0.48**	3.44
i25-t09-08	2398.74	6.38	1.94	1.81	6.98	2.80	0.35	3.66	**1.51**	0.54
i25-t09-09	1706.13	0.80	**0.20**	2.35	23.18	7.08	1.46	1.20	0.40	2.25
i25-t09-10	486.48	1.01	0.54	1.58	4.04	0.70	1.67	0.39	**0.00**	1.13
i25-t12-05	3956.71	1.04	**0.00**	3.28	13.89	1.85	8.90	6.91	0.81	17.20
i25-t12-06	2122.71	0.75	**0.14**	0.72	9.98	3.10	2.87	6.74	1.29	7.61
i25-t12-10	6856.82	0.36	**0.00**	0.71	9.14	1.07	0.77	0.36	**0.00**	0.93
i50-t03-03	892.88	5.00	2.36	0.97	20.19	9.62	1.56	2.26	**0.00**	0.56
i50-t03-04	2362.81	5.41	4.39	0.99	24.39	8.27	0.73	2.76	**0.71**	1.00
i50-t03-06	3895.02	1.82	**0.57**	1.15	18.16	6.12	0.88	5.45	1.05	1.81
i50-t03-07	942.33	8.69	5.79	1.49	18.41	11.48	0.75	8.27	**2.69**	0.72
i50-t03-08	495.78	8.61	5.09	1.48	24.72	10.46	0.48	6.94	**3.24**	0.76
i50-t03-09	3643.22	2.57	0.77	0.80		Infeasible		6.78	**0.43**	0.85
i50-t03-10	2790.10	3.71	3.16	0.98	16.00	10.67	1.03	5.70	**0.90**	0.30
Average	1507.21	2.62	1.22	1.34	13.26	3.74	1.58	3.69	1.05	1.81

Table 4: Average percent CPLEX gaps and CPU times (in seconds) of the CPLEX and a priori routing heuristic solutions for instances where CPLEX can find a feasible solution (10 instances for each setting unless otherwise stated)

		CPLEX	RR		S-shape		Largest gap	
$\|M\|$	$\|T\|$	CPU time	$\%C_{gap}(H^+)$	CPU time	$\%C_{gap}(H^+)$	CPU time	$\%C_{gap}(H^+)$	CPU time
25	3	10.9	1.03	0.4	3.17	0.2	1.18	0.4
	6	339.3	0.73	1.5	2.20	0.9	1.06	1.7
	9	2247.2	0.83	2.2	2.83	3.2	0.81	1.9
	12	7204.7	-0.30	35.3	1.44	61.6	0.32	80.0
	15	7204.9	-0.40	43.7	1.67	102.5	-0.01	92.1
50	3	2354.1	2.54	1.3	6.72	0.9	1.14	0.9
	6[1]	7208.7	-1.88	4.2	0.85	5.5	-1.45	4.9
	9[2]	7222.5	-3.55	13.9	3.88	36.6	-5.14	7.5
Average		5630.8	0.35	13.1	2.81	26.2	0.35	26.4

[1] Average of 7 instances
[2] Average of 2 instances

there is no significant difference between the average objective values of the CPLEX results and those of the two heuristics. The S-shape heuristic yields an average gap of 2.81%, and the t-test result in Appendix C.1 shows that the objective value difference from CPLEX is statistically significant. On the other hand, this heuristic still finds better solutions than CPLEX in three instances. The results in Table 4 underline the computational burden faced by CPLEX as instance sizes increase, and point to the RR and largest gap a priori routes obtaining comparable objective values to CPLEX, further justifying their strength.

To compare the a priori heuristics among each other and to quantify the effect of improvement using the a posteriori routing, Table 5 shows a comparison of the heuristics in terms of RR^+ gaps and CPU times for all 300 smaller instances. On average, all heuristics terminate within a minute per instance, and the maximum average run time for any setting is less than six minutes. Without improvement, we observe a similar pattern in the gaps for all heuristics to those in Table 3. The RR heuristic outperforms largest gap, with average RR^+ gaps of 3.61% and 4.75%, respectively, and S-shape heuristic exhibits a gap of 10.44%. Appendix C.2 in the Supplementary Material shows that for a 95% confidence level, the differences between the average objective values of these three approaches are statistically significant. For smaller number of items, the difference between RR and largest gap is smaller, whereas for larger number of items, the difference increases. This is due to the structure of the largest gap heuristic, where dense replenishment lists (larger instances) result in excessive double-traversal of aisles. The main reason behind the inferior performance of the S-shape heuristic is the excessive travel time in the a priori route.

When the improved versions of the three approaches using a posteriori routing are considered, there are three main observations. First, we observe that for all three heuristics, the a posteriori routing step significantly improves the routes, as evidenced by the paired t-test results in Appendix C.3. Second, the improved version of the largest gap heuristic outperforms that of RR, with an average improvement of 0.52%. Of the 300 instances, RR and largest gap find the best solution in 101 and 184 cases, respectively, with equal objective values for eight instances. Whenever feasible, the improved S-shape heuristic performs worse than the other two improved approaches. Appendix C.2 shows that the difference between the improved versions of the three heuristics are statistically significant. Lastly, we assess the solution quality of the improved versions of the heuristics using a lower bounding scheme, for which a pseudocode is provided in Appendix D. The lower bound is obtained by

Table 5: Average percent RR^+ gaps, lower bound gaps, and CPU times (in seconds) of the a priori routing heuristics for all small instances (10 instances for each setting), with numbers in brackets indicating number of infeasible solutions

		RR			S-shape				Largest gap			
$\|M\|$	$\|T\|$	$RR^+(H)$	$\%L_{gap}(H^+)$	CPU time	$RR^+(H)$	$RR^+(H^+)$	$\%L_{gap}(H^+)$	CPU time	$RR^+(H)$	$RR^+(H^+)$	$\%L_{gap}(H^+)$	CPU time
25	3	1.30	2.66	0.4	10.78	2.11	4.81	0.2	2.17	0.14	2.80	0.4
	6	1.41	3.65	1.5	10.76	1.46	5.14	0.9	2.98	0.33	3.98	1.7
	9	1.49	4.28	2.2	11.28	1.98	6.33	3.2	1.77	-0.02	4.26	1.9
	12	0.94	4.97	35.3	10.92	1.74	6.78	61.6	3.53	0.62	5.61	80.0
	15	2.12	5.36	43.7	10.22	2.00	7.45	102.5	2.82	0.31	5.68	92.1
50	3	2.07	3.26	1.3	13.81 (1)	4.67 (1)	8.04 (1)	0.9 (1)	2.16	-1.37	1.86	0.9
	6	2.65	3.46	4.8	13.46 (7)	3.46 (7)	7.00 (7)	5.5 (7)	4.43	0.52	3.99	5.1
	9	3.01	3.95	17.1	12.46 (2)	2.38 (2)	6.40 (2)	85.9 (2)	2.57	-0.79	3.14	19.8
	12	3.30	5.04	68.8	11.15 (1)	1.76 (1)	6.87 (1)	158.2 (1)	3.11	-0.08	4.95	31.9
	15	3.81	5.28	328.4	9.58 (3)	0.88 (3)	6.20 (3)	248.5 (3)	2.05	-1.16	4.07	153.2
75	3	3.24	3.36	1.6	11.50	1.42	4.81	1.2	2.50	-1.94	1.37	1.5
	6	5.74	3.60	9.8	13.87	2.94	6.62	13.0	5.58	1.04	4.67	13.1
	9	4.63	4.06	16.0	12.57	0.49	4.56	20.5	3.93	-0.84	3.19	8.5
	12	4.16	5.13	27.4	10.88	1.21	6.39	21.5	4.83	-0.11	5.01	35.9
	15	3.71	5.84	188.3	All instances infeasible				2.67	-0.64	5.17	70.9
100	3	4.00	2.98	3.3	11.39	2.02	5.04	3.1	4.80	-1.14	1.82	3.0
	6	3.63	3.10	20.5	10.79	1.16	4.29	17.0	5.19	-0.23	2.87	22.6
	9	3.57	3.77	31.6	10.98	1.49	5.30	47.7	5.87	-0.14	3.62	22.5
	12	4.77	5.36	48.3	9.91	0.64	6.03	43.1	4.07	-0.65	4.63	43.0
	15	4.05	5.93	129.8	9.34	0.55	6.50	100.2	4.10	-1.11	4.76	78.2
125	3	5.75	3.17	3.7	8.93	1.60	4.80	3.1	7.62	-0.58	2.58	3.2
	6	4.12	3.56	27.8	9.47	0.69	4.27	43.3	6.81	-0.62	2.92	42.6
	9	4.06	3.90	63.0	10.14	0.89	4.81	55.7	6.85	-0.63	3.25	38.2
	12	4.24	5.30	109.8	9.67	0.54	5.86	125.4	5.85	-0.77	4.50	77.5
	15	4.22	5.67	121.9	8.99	0.64	6.34	100.6	5.84	-1.06	4.56	80.4
150	3	3.63	2.80	5.1	5.96	0.76	3.58	5.0	8.61	-1.17	1.61	5.3
	6	5.09	3.34	66.6	8.55	0.68	4.04	55.6	8.11	-0.20	3.14	46.3
	9	4.73	4.05	74.4	8.37	0.42	4.48	76.6	7.39	-1.15	2.87	50.9
	12	4.33	4.91	94.3	9.15	0.18	5.10	119.5	6.79	-1.07	3.80	120.1
	15	4.41	5.64	150.0	7.85	0.25	5.90	142.1	7.61	-1.00	4.60	110.2
Average		3.61	4.25	56.6	10.44	1.41	5.65	57.3	4.75	-0.52	3.71	42.0

ignoring the time and load capacity limits, and solving the remaining problem optimally. We denote the average relative gaps of the heuristic solutions from the lower bound as $\%L_{gap}(H^+)$ in Table 5. The average gaps show that the three a priori routing heuristics are within 4.25%, 5.65%, and 3.71% of the lower bounds (and hence optimal solutions), which underlines the high solution quality of these approaches even for larger instances for which CPLEX cannot find a feasible solution.

To obtain a better understanding of the structures of the replenishment routes resulting from the heuristics, Appendix E provides each heuristic route for three replenishment cycles of the $i50 - t03 - 07$ instance. The routes representation is not reported here for the sake of conciseness. Also, the solution of this instance is representative of the remaining instances. The inferiority of the S-shape routes can be immediately observed, where the replenished items are scattered throughout the warehouse for all three cycles. This leads to excessive inter-aisle movement, as evidenced by the long horizontal lines. Improvement by an a posteriori routing step removes part of the excessive inter-aisle moves, but the inefficient assignment of item slots to replenishment cycles implies that the improved routes are still inferior to those under the RR and largest gap heuristics. With both RR and largest gap a priori routes, the resulting assignment of items to replenishment cycles is substantially better, but the fixed ordering of replenishment visits may lead to the route crossing itself at times. The improvement step is able to alleviate this issue and produce near-optimal routes in both cases.

To ensure that the dimensions of the warehouse do not have a significant effect on our results, we repeat our experiments with the same settings, only changing the number of aisles to 20 and the number of item slots in each aisle to 45. The results in Appendix F of the Supplementary Material show that the relative performance of

the heuristics and the gap values do not exhibit significant change. Consequently, we conclude that our results and insights are not specific to the original warehouse dimensions.

Overall, our experiments with smaller instances show that the a priori routing heuristics perform particularly well with the RR and largest gap routes. The average gaps to optimum are around 1% for smaller instances and the solutions are obtained much more quickly than CPLEX. With up to 150 items and 15 cycles, solutions can be obtained within six minutes on average, owing to the tractability of routing for the SRRP. A posteriori routing provides statistically significant improvement in the solutions, and the lower bound gaps indicate that the obtained solutions are very close to optimal.

5.2. Experiments with Larger Instances

With a set of larger instances, our aim is to analyse the effect of demand patterns and compare our heuristics to various benchmark approaches. The warehouse settings are identical to those with smaller instances, except that each aisle includes 15 item slots, and the depot is in the middle of the front cross aisle. With 30 aisles and 15 slots on each, there are a total of 450 item slots. We assume that any of the 450 items can be demanded in any of the 15 cycles we consider, but a fixed total number of items are demanded in each picking cycle. We vary this number as 25, 50, 75, 100, 150, 200, 250, and 300.

To determine which items will be demanded in each cycle, we consider four different demand skewness patterns. Under uniform demand, the items demanded in each cycle are generated randomly, regardless of location within the forward storage area. We also consider 20-40, 20-60, and 20-80 demand skewness patterns, where top 20% of the most popular items constitute 40%, 60%, and 80% of the demand, respectively. Under these patterns, we incorporate a turnover-based storage policy, in which more popular items are stored at a closer distance to the depot. To do so, we sort the items in terms of decreasing popularity, sort the slots in terms of closeness to the depot, and assign items to slots accordingly. In this sorted list, probability p_i of the demand for the i^{th} most popular item is given by:

$$p_i = F\left(\frac{i}{|M|}\right) - F\left(\frac{i-1}{|M|}\right), \text{ where } F(i) = \frac{(1+A)i}{A+i}.$$

Here, A is a shape parameter whose value is 3/5, 1/5, and 1/16 for 20-40, 20-60, and 20-80 demand patterns, respectively. If a item i is demanded in cycle t, its demand is randomly generated between 0 and $2\gamma_i$, where $\gamma_i \sim U[10, 90]$ is the base demand for each item i.

Item arrivals at the reserve storage, slot capacities, and initial inventory levels are identically determined to those in the preceding section, except that base demand is used to determine these as opposed to the first-cycle demand. Travel time limit is 500 seconds, and load limit is 1500 item units.

To compare against our heuristic, we consider four benchmark approaches based on similar ones in the literature and industry that do not incorporate routing into the replenishment decisions (Richards, 2014; APS Fulfillment, 2021). The first is based on *demand (wave) replenishment*, where the total demand amount of each item in the upcoming order picking cycle is replenished in the preceding replenishment cycle. The next three are based on *triggered replenishment*, where an item is replenished whenever its stock level hits a pre-determined

minimum level. We combine this with *top-off replenishment*, where the item is replenished up to its capacity. The three variants of this combination, which we refer to as *Triggered-1/2*, *Triggered-1/4*, and *Triggered-0*, replenish an item up to its capacity level whenever the inventory level would go below half its capacity, a quarter of its capacity, and zero in the upcoming picking cycle, respectively.

With eight possible number of items, four demand skewness levels and 10 instance for each setting, we run a total of 320 instances. We use improved largest gap for a priori routing, due to its superior relative performance to the other methods in the preceding section. The CPU time for the heuristic ranges from 221.6 to 4215.2 seconds, with an average of 1002.3 seconds. All benchmark heuristics run within 1 second for each instance.

Figure 8 shows the results of our experiments with larger instances, which are grouped by demand skewness levels. As expected, we observe that for any demand skewness level and number of items, the worst performance is obtained by Demand Replenishment, followed by Triggered-1/2, 1/4, and 0, respectively. This is due to the decreased frequency of replenishments, and thus the total replenishment time, in this order. Another expected observation is that for any policy and number of items, total replenishment time decreases as demand skewness increases, since the turnover-based storage can provide more travel-time savings as demand is more skewed.

The a priori routing heuristic outperforms its counterparts in every setting. Under any demand skewness level, the relative difference is more prominent when fewer items are demanded per cycle. For example, under uniform storage, replenishment time savings of $LG(H^+)$ over Triggered-0 is 53.7% with 25 items per cycle, which decreases to 13.4% with 300 items. Similarly, for demand replenishment, the savings are 77.4% and 22.8%, respectively. As the number of items increases, so does the replenishment frequency, whereby a significant portion of the forward storage area is traversed in any case, offsetting the potential advantages from routing. The average improvement over Triggered-0 ranges from 29.9% to 32.1% over the four skewness levels, whereas the improvement ranges are 43.8%-46.2% over Triggered-1/4, 49.9%-52.2% over Triggered-1/2, and 50.7%-53.1% over Demand Replenishment. These substantial improvement levels underline the importance of incorporating routing into storage replenishment decisions.

Under the proposed heuristic, the effect of demand skewness also varies depending on the number of items. With 25 items per cycle, compared to 20-80 demand skewness, total replenishment time increases by 14.7%, 19.4%, and 36.6% under 20-60, 20-40, and uniform demand skewness, respectively. The corresponding increases are 0.8%, 3.2%, and 6.8% with 300 items. On average over all possible number of items, the objective increases by 4.0%, 9.4%, and 17.3% for 20-60, 20-40, and uniform demand skewness, respectively. The benchmark heuristics behave similarly to each other, with average increases of 7.2%, 14.4%, and 18.8% for the same skewness levels. The differences in these averages between the proposed heuristic and the benchmarks imply that the former is more robust to changes in demand skewness.

To ensure that the a priori heuristic results for the larger instances are not far from optimal, we apply the lower bounding scheme discussed in Section 5.1 for this set of instances. Appendix G in the Supplementary Material provides the average percent lower bound gaps for each number of items per cycle and demand skewness. We deduce from these gaps that although the gap increases with increasing number of items per cycle, the maximum gap is around 9%. Given that this is an upper bound on the gap from optimal, it is reasonable to

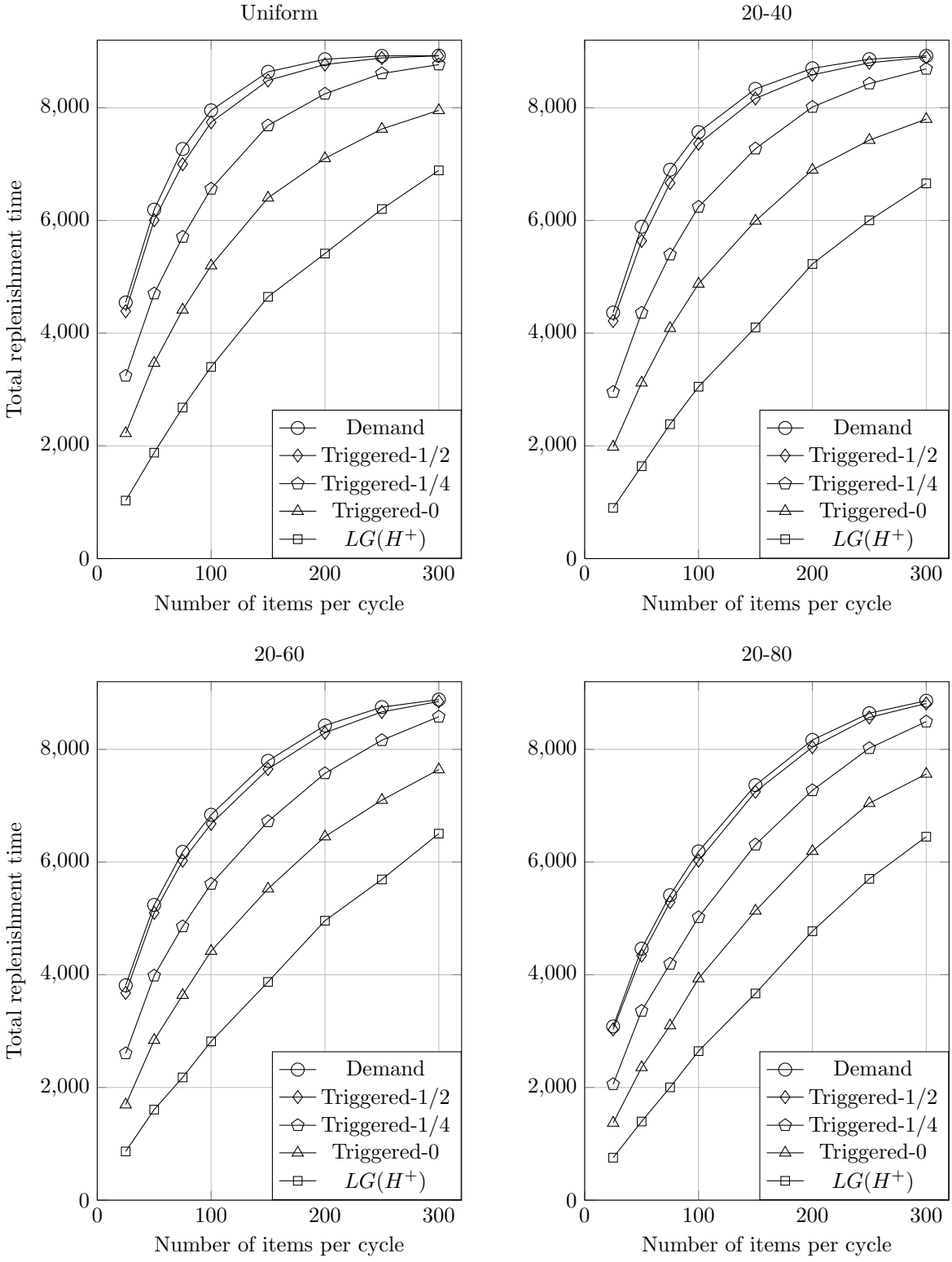

Figure 8: Average replenishment time for the a priori routing heuristic ($LG(H^+)$) and four benchmark methods for varying demand skewness levels (each data point is an average of 10 instances)

conclude that the heuristic produces high-quality routes, even for larger instances.

6. Conclusions and Further Research Directions

In warehouses with separate forward and reserve storage areas, integration of storage replenishment and order picking operations plays an important role in determining the timeliness of shipments. When multiple item slots need to be replenished in limited time intervals, an efficient routing of the replenishment worker is important to ensure the availability of items for order picking. For this end, this paper has introduced the storage replenishment routing problem, where the decisions involve which item slots to replenish in which cycle and sequence subject to a time and capacity constraint for each cycle. We have established the complexity of the SRRP for various cases and developed a heuristic approach. Our computational experiments show that the proposed approach not only finds near-optimal solutions within very short computational time, but it is also quite robust in terms of demand skewness and substantially outperforms different approaches where routing is not part of the replenishment decisions.

Given the important theoretical and practical implications of integrating storage replenishment and routing decisions, the SRRP gives way to a number of interesting further research directions. An immediate extension of this work is to adapt the SRRP and the solution approaches to different warehouse layouts. Exact and heuristic routing algorithms exist for the case of multiple blocks (e.g. Roodbergen & de Koster, 2001a,b), as well as fishbone and flying-V layouts (e.g. Çelik & Süral, 2014), which could be used as the a priori routing scheme. Extension to multiple replenishment workers may capture a wider set of applications. With multiple blocks and multiple replenishment workers, the heuristics in Valle et al. (2017) can be used to extend the a priori routing approach to involve batching decisions as well. Another extension would be to consider the case where replenishment and order picking are simultaneous. This also involves the possibility of order pickers having to wait for replenishment during the pick waves, which exacerbates the difficulty of routing. On the other hand, focusing on the case with unit-load replenishments under this setting leads to a more tractable problem, which may enable some of the approaches for the SRRP to be adapted.

With the advent of automation in the recent years, many warehouses store items in multiple slots at the same time. Our approaches are directly applicable in such an environment if order picking and replenishment plans are sequentially made, i.e., picker routes are determined irrespective of replenishment, and replenishment routing takes picking schedules as inputs. An interesting extension here is to determine the order picking and replenishment routes in an integrated manner. The main challenge in this case is that the picker routing problem by itself is $\mathcal{NP}$-hard, even for a single cycle (Daniels et al., 1998), which leads to an even harder problem if replenishment is also considered. However, devising efficient heuristics for the picker routing problem with multiple cycles and multiple slots for the same item may allow for the a priori routing approach to be applicable for the integrated problem.

References

Adulyasak, Y., Cordeau, J.-F., & Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, *26*, 103–120. doi:10.1287/ijoc.2013.0550.

Alvarez, A., Munari, P., & Morabito, R. (2018). Iterated local search and simulated annealing algorithms for the inventory routing problem. *International Transactions in Operational Research*, *25*, 1785–1809. doi:10.1111/itor.12547.

APS Fulfillment (2021). How to optimize your warehouse replenishment. https://www.apsfulfillment.com/warehouse-fulfillment/how-to-optimize-your-warehouse-replenishment/. Accessed: August 2021.

Archetti, C., Bertazzi, L., Laporte, G., & Speranza, M. G. (2007). A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation Science*, *41*, 382–391. doi:10.1287/trsc.1060.0188.

Archetti, C., Bianchessi, N., Irnich, S., & Speranza, M. G. (2014). Formulations for an inventory routing problem. *International Transactions in Operational Research*, *21*, 353–374. doi:10.1111/itor.12076.

Archetti, C., Boland, N., & Speranza, M. G. (2017). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, *29*, 377–387. doi:10.1287/ijoc.2016.0737.

Archetti, C., & Speranza, M. G. (2016). The inventory routing problem: The value of integration. *International Transactions in Operational Research*, *23*, 393–407. doi:10.1111/itor.12226.

Bartholdi, J. J., & Hackman, S. (2019). Warehouse and Distribution Science. Release 0.98.1. https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf. Accessed: June 2020.

Bertazzi, L., & Speranza, M. G. (2012). Inventory routing problems: An introduction. *EURO Journal on Transportation and Logistics*, *1*, 307–326. doi:10.1007/s13676-012-0016-7.

Bertazzi, L., & Speranza, M. G. (2013). Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, *2*, 255–275. doi:10.1007/s13676-013-0027-z.

Carrasco-Gallego, R., & Ponce-Cueto, E. (2009). Redesigning a piece picking area replenishment process supported by a WMS. In *2009 International Conference on Computers & Industrial Engineering* (pp. 748–753). doi:10.1109/ICCIE.2009.5223856.

Çelik, M., & Süral, H. (2014). Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE Transactions*, *46*, 283–300. doi:10.1080/0740817X.2013.768871.

Çelik, M., & Süral, H. (2016). Order picking in a parallel-aisle warehouse with turn penalties. *International Journal of Production Research*, *54*, 4340–4355. doi:10.1080/00207543.2016.1154624.

Çelik, M., & Süral, H. (2019). Order picking in parallel-aisle warehouses with multiple blocks: Complexity and a graph theory-based heuristic. *International Journal of Production Research*, *57*, 888–906. doi:10.1080/00207543.2018.1489154.

Chabot, T., Lahyani, R., Coelho, L. C., & Renaud, J. (2017). Order picking problems under weight, fragility and category constraints. *International Journal of Production Research*, *55*, 6361–6379. doi:10.1080/00207543.2016.1251625.

Chitsaz, M., Cordeau, J.-F., & Jans, R. (2019). A unified decomposition matheuristic for assembly, production, and inventory routing. *INFORMS Journal on Computing*, *31*, 134–152. doi:10.1287/ijoc.2018.0817.

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, *24*, 270–287. doi:10.1016/j.trc.2012.03.007.

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, *48*, 1–19. doi:10.1287/trsc.2013.0472.

Coelho, L. C., & Laporte, G. (2013a). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, *51*, 7156–7169. doi:10.1080/00207543.2012.757668.

Coelho, L. C., & Laporte, G. (2013b). The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, *40*, 558–565. doi:10.1016/j.cor.2012.08.012.

Coelho, L. C., & Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, *155*, 391 – 397. doi:10.1016/j.ijpe.2013.11.019.

Daniels, R. L., Rummel, J. L., & Schantz, R. (1998). A model for warehouse order picking. *European Journal of Operational Research*, *105*, 1–17. doi:10.1016/S0377-2217(97)00043-X.

de Koster, R. B. M., Le-Duc, R., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, *182*, 481–501. doi:10.1016/j.ejor.2006.07.009.

De Santis, R., Montanari, R., Vignali, G., & Bottani, E. (2018). An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research*, *267*, 120–137. doi:10.1016/j.ejor.2017.11.017.

de Vries, H., Carrasco-Gallego, R., Farenhorst-Yuan, T., & Dekker, R. (2014). Prioritizing replenishments of the piece picking area. *European Journal of Operational Research*, *236*, 126–134. doi:10.1016/j.ejor.2013.12.045.

Desaulniers, G., Rakke, J. G., & Coelho, L. C. (2016). A branch-price-and-cut algorithm for the inventory routing problem. *Transportation Science*, *50*, 1060–1076. doi:10.1287/trsc.2015.0635.

Dijkstra, A. S., & Roodbergen, K. J. (2017). Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses. *Transportation Research Part E: Logistics and Transportation Review*, *102*, 38–59. doi:10.1016/j.tre.2017.04.003.

Gagliardi, J.-P., Ruiz, A., & Renaud, J. (2008). Space allocation and stock replenishment synchronization in a distribution center. *International Journal of Production Economics*, *115*, 19–27. doi:10.1016/j.ijpe.2008.04.006.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability*. New York, NY: W. H. Freeman and Company.

Glock, C. H., Grosse, E. H., Abedinnia, H., & Emde, S. (2019). An integrated model to improve ergonomic and economic performance in order picking by rotating pallets. *European Journal of Operational Research*, *273*, 516–534. doi:10.1016/j.ejor.2018.08.015.

Gong, J., Tamura, H., Higashi, T., & Ota, J. (2008). Scheduling of order-picking with replenishment in a warehouse environment. *SICE Transactions on Industrial Application*, *7*, 33–39.

Gu, J., Goetschalckx, M., & McGinnis, L. (2010). Solving the forward-reserve allocation problem in warehouse order picking systems. *Journal of the Operational Research Society*, *61*, 1013–1021. doi:10.1057/jors.2009.39.

Hackman, S. T., Rosenblatt, M. J., & Olin, J. M. (1990). Allocating items to an automated storage and retrieval system. *IIE Transactions*, *22*, 7–14. doi:10.1080/07408179008964152.

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, *25*, 76–87. doi:10.1080/07408179308964306.

Haverhals, J. A. (2019). Approximation of the optimal inventory replenishment policy for multi-items used in a warehouse. Eindhoven Institute of Technology. Unpublished thesis.

Kim, B.-I., Heragu, S. S., Graves, R. J., & St. Onge, A. (2003). Realization of a short cycle time in warehouse replenishment and order picking. *International Journal of Production Research*, *41*, 349–364. doi:10.1080/00207540210166321.

Makris, P. A., & Giakoumakis, I. G. (2003). k-interchange heuristic as an optimization procedure for material handling applications. *Applied Mathematical Modelling*, *27*, 345–358. doi:10.1016/S0307-904X(02)00137-3.

Masae, M., Glock, C. H., & Grosse, E. H. (2020). Order picker routing in warehouses: A systematic literature review. *International Journal of Production Economics*, *224*, 107564. doi:10.1016/j.ijpe.2019.107564.

Menéndez, B., Pardo, E. G., Alonso-Ayuso, A., Molina, E., & Duarte, A. (2017). Variable neighborhood search strategies for the order batching problem. *Computers & Operations Research*, *78*, 500–512. doi:10.1016/j.cor.2016.01.020.

Pansart, L., Catusse, N., & Cambazard, H. (2018). Exact algorithms for the order picking problem. *Computers & Operations Research*, *100*, 117–127. doi:10.1016/j.cor.2018.07.002.

Petersen, C. G. (1997). An evaluation of order picking routeing policies. *International Journal of Production & Operations Management*, *17*, 1098–1111. doi:10.1108/01443579710177860.

Pınar, Ö.., & Süral, H. (2006). Coordinating inventory and transportation in vendor managed systems. In R. Meller, M. K. Ogle, B. A. Peters, G. D. Taylor, & J. Usher (Eds.), *Proceedings of the 9th International Material Handling Colloquium (IMHRC)* (pp. 459–474). Material Handling Institute of America, Charlotte, NC.

Ratliff, H. D., & Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, *31*, 507–521. doi:10.1287/opre.31.3.507.

Richards, G. (2014). *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. (2nd ed.). London, United Kingdom: Kogan Page.

Roldán, R., Basagoiti, R., & Coelho, L. C. (2017). A survey on the inventory-routing problem with stochastic lead times and demands. *Journal of Applied Logic*, *24*, 15–24. doi:10.1016/j.jal.2016.11.010.

Roodbergen, K. J., & de Koster, R. B. M. (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, *39*, 1865–1883. doi:10.1080/00207540110028128.

Roodbergen, K. J., & de Koster, R. B. M. (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, *133*, 32–43. doi:10.1016/S0377-2217(00)00177-6.

Ruberg, Y., & Scholz, A. (2016). *A mathematical programming formulation for the single-picker routing problem in a multi-block layout*. Working Paper 2/2016 Faculty of Economics and Management, Universität Magdeburg, Germany.

Santos, E., Ochi, L., Simonetti, L., & González, P. (2016). A hybrid heuristic based on iterated local search for multivehicle inventory routing problem. *Electronic Notes in Discrete Mathematics*, *52*, 197–204. doi:10.1016/j.endm.2016.03.026.

Scholz, A., Henn, S., Stuhlmann, M., & Wäscher, G. (2016). A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, *253*, 68–84. doi:10.1016/j.ejor.2016.02.018.

Scholz, A., & Wäscher, G. (2017). Order batching and picker routing in manual order picking systems: The benefits of integrated routing. *Central European Journal of Operations Research*, *25*, 491–520. doi:10.1007/s10100-017-0467-x.

Solyalı, O., & Süral, H. (2011). A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. *Transportation Science*, *45*, 335–345. doi:10.1287/trsc.1100.0354.

Takano, R., Higashi, T., Tamura, H., Sugi, M., & Ota, J. (2011). Mixed-load transportation scheduling of multiple agents in a warehouse environment. *Advanced Robotics*, *25*, 1557–1576. doi:10.1163/016918611X579538.

Theys, C., Bräysy, O., Dullaert, W., & Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, *200*, 755–763. doi:10.1016/j.ejor.2009.01.036.

Thomas, L. M., & Meller, R. D. (2015). Developing design guidelines for a case-picking warehouse. *International Journal of Production Economics*, *170*, 741–762. doi:10.1016/j.ijpe.2015.02.011.

Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities Planning*. Hoboken, NJ: John Wiley & Sons.

Valle, C. A., Beasley, J. E., & da Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, *262*, 817–834. doi:10.1016/j.ejor.2017.03.069.

van den Berg, J. P., Sharp, G. P., Gademann, A., & Pochet, Y. (1998). Forward-reserve allocation in a warehouse with unit-load replenishments. *European Journal of Operational Research*, *111*, 98–113. doi:10.1016/S0377-2217(98)80013-1.

van Gils, T., Ramaekers, K., Caris, A., & de Koster, R. B. M. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, *267*, 1–15. doi:10.1016/j.ejor.2017.09.002.

Vaughan, T. S., & Petersen, C. G. (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, *37*, 881–897. doi:10.1080/002075499191580.

Žulj, I., Glock, C. H., Grosse, E. H., & Schneider, M. (2018). Picker routing and storage-assignment strategies for precedence-constrained order picking. *Computers & Industrial Engineering*, *123*, 338–347. doi:10.1016/j.cie.2018.06.015.

Weidinger, F. (2018). Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, *95*, 139–150. doi:10.1016/j.cor.2018.03.012.

Weidinger, F., Boysen, N., & Schneider, M. (2019). Picker routing in the mixed-shelves warehouses of e-commerce retailers. *European Journal of Operational Research*, *274*, 501–515. doi:10.1016/j.ejor.2018.10.021.

Wu, W., de Koster, R. B. M., & Yu, Y. (2020). Forward-reserve storage strategies with order picking: When do they pay off? *IISE Transactions*, *52*, 961–976. doi:10.1080/24725854.2019.1699979.