European Journal of Operational Research 308 (2023) 696–708

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

Scheduling shuttles in deep-lane shuttle-based storage systems

Jingjing Yang[a], René B. M. de Koster[b], Xiaolong Guo[a], Yugang Yu[a,*]

[a] Anhui Province Key Laboratory of Contemporary Logistics and Supply Chain, International Institute of Finance, School of Management, University of Science and Technology of China, 96 Jinzhai Road, Hefei 230026, Anhui, P.R. China
[b] Department of Technology and Operations Management, Rotterdam School of Management, Erasmus University of Rotterdam, Rotterdam 3062 PA, The Netherlands

ARTICLE INFO

Article history:
Received 14 March 2022
Accepted 18 November 2022
Available online 1 December 2022

Keywords:
Logistics
Scheduling shuttles
Transfer shuttles
Deep-lane storage
Shuttle-based system

ABSTRACT

Deep-lane shuttle-based storage systems with forklifts are widely used in bulk storage and finished-goods warehouses that require dense storage paired with high operational efficiency. In such a system, automated shuttles take care of the movements of loads within the storage lanes, and forklifts take care of the horizontal and vertical movements of loads. Since shuttles are expensive, the number of shuttles in the system is typically smaller than the number of storage lanes, so shuttle transfers are required with the help of the forklift. This paper focuses on optimizing the schedule: Given a set of retrieval requests and shuttles, how can the shuttle transfer sequence and the retrieval request sequence be arranged to minimize the makespan? A mathematical model is formulated. Since the problem is NP-hard, an efficient two-stage heuristic is proposed to compute near-optimal solutions. This study's numerical results show that the two-stage heuristic can provide high-quality solutions in reasonable time. Compared to straightforward heuristics used in practice and in the literature, the makespan can be reduced considerably. The proposed two-stage heuristic can also be used to decide the optimal shuttle fleet size.

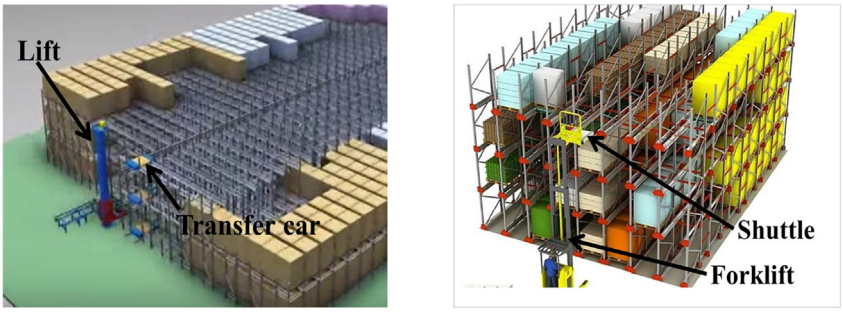
1. Introduction

In many areas of the world, land and warehouse buildings are expensive. Particularly for products that are produced in large quantities or which have to be refrigerated, it is important to have compact warehouse storage areas and buildings (Goetschalckx & Ratliff, 1991; Yu & De Koster, 2012). Compact storage systems can lead to less space consumption and lower energy consumption (Azadeh, De Koster, & Roy, 2019). A commonly used solution is deep-lane storage. Deep-lane warehouses can improve both order-picking efficiency (due to reduced travel time) and warehouse floor space. Both conventional and automated solutions exist. Fig. 1(a) shows an automated shuttle-based system consisting of multiple tiers of multi-deep storage lanes serviced by multiple shuttles, transfer cars (to move shuttles between the lane), and lifts (to move shuttles with pallets between the tiers). Other automated deep-lane systems include rail-mounted cranes moving shuttles between lanes at different tiers (Zaerpour, Yu, & de Koster, 2015). Note that these systems require last-in-first-out (LIFO) operations per storage lane. Therefore, they are used when multiple loads of a single product and production lot can be stored in the same storage lane. It is possible to store multiple products in a single deep lane, but this requires extensive reshuffling operations, lowering the throughput capacity.

Also, conventional systems exist (see Fig. 1(b)) in which shuttles are moved between lanes by manually operated forklifts. Such systems are more flexible than crane-based or automated shuttle-based systems, as multiple forklifts may work in parallel in the same space. In such a forklift-based system, a forklift fetches a shuttle and drives to a particular lane where a load needs to be retrieved. The forklift (see Fig. 2(a)) also takes care of load or shuttle movements between the tiers. At the retrieval lane, the forklift drops off the shuttle, which drives autonomously on a rail underneath the loads. It lifts the load in the storage lane and brings it to the front end of the storage lane. There, the forklift can pick up the load with the shuttle, or the load without the shuttle, so that the shuttle is available to retrieve another load in that lane. If a system has fewer shuttles than storage lanes, the forklift moves the shuttles between the lanes (Stadtler, 1996). Shuttle-based systems with transfer cars (see Figs. 1(a) and 2(b)) are similar to shuttle-based systems with forklifts (although forklifts can also serve multiple levels). The main difference between these systems is that a forklift can transfer any of the three units: loaded shuttles, only loads, or empty shuttles. Transfer cars, however, cannot pick up a load without a shuttle. In recent years, the application of autonomous shuttles, operating together with forklifts or transfer cars, has become

* Corresponding author.
E-mail addresses: yjj2015@mail.ustc.edu.cn (J. Yang), rkoster@rsm.nl (R.B.M. de Koster), gxl@ustc.edu.cn (X. Guo), ygyu@ustc.edu.cn (Y. Yu).

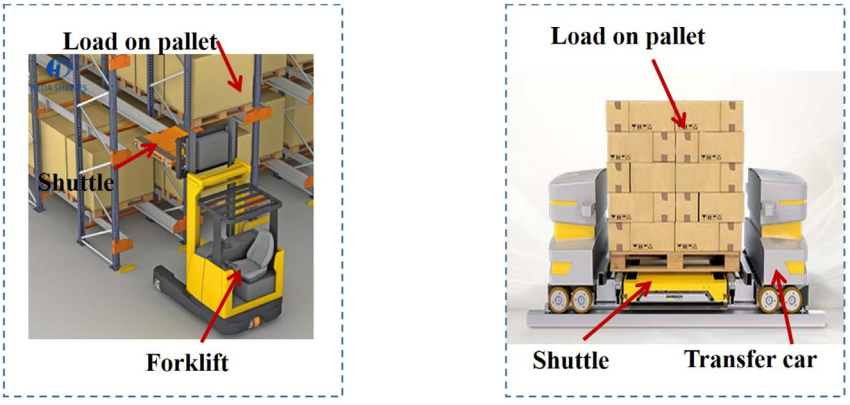
https://doi.org/10.1016/j.ejor.2022.11.037
0377-2217/



(a) A shuttle-based system with a lift and transfer cars (Tappia et al., 2017). *Source*: Total Solution Provider Group

(b) A shuttle-based system with a forklift. *Source*: Maxrac.com

Fig. 1. Shuttle-based deep-lane storage systems.



(a) A forklift in a shuttle-based system

(b) A transfer car in a shuttle-based system

Fig. 2. A forklift and a transfer car.

Table 1
Unit handling time.

Description of handling step	Time value
The shuttle picks up or drops off the unit load.	t_o
The forklift picks up or drops off the unit load.[a]	t_u
The forklift / transfer car picks up or drops off the shuttle.	t_s
The forklift / transfer car picks up or drops off both the shuttle and the unit load.	t_c

[a] Note: A transfer car cannot pick up a load without a shuttle.

more popular due to dropping prices of the shuttles, the flexibility offered, and a large choice of suppliers offering such systems (see the websites of various system suppliers, e.g., Automha.com or Toyota.com).

To fully benefit from such systems, it is important to schedule the forklift, transfer car, and shuttles to store or retrieve the loads. The throughput capacity depends not only on the number of shuttles but also on how they are deployed. Proper deployment may save investment in shuttles or lead to a reduction in makespan or throughput time, yielding higher throughput capacity. This scheduling problem was inspired by a visit to a warehouse designed by Toyota). In this warehouse, products are stored in ten pallet-deep channels in racks five levels high, with multiple shuttles transferred between the channels by a forklift. Each channel contains one batch of a product. Either the pallet with load or the shuttle with pallet can be picked up by the forklift. Toyota material handling provided detailed data about a similar warehouse, which we used in the numerical experiments (see Table 4). Surprisingly, this scheduling problem in multi-deep systems has received little academic attention, though the literature on scheduling decisions in single-deep automated systems is abundant (Carlo & Vis, 2012; Gharehgozli, Yu, Zhang, & De Koster, 2017; Han, McGinnis, Shieh, & White, 1987; Roodbergen & Vis, 2009; Zhao, Luo, & Lodewijks, 2018). We are aware of only one paper by Dong, Jin, Wang, & Kelle (2021) studying a scheduling problem in a multi-deep crane-based AS/RS. They assume that the crane picks up only the load from the shuttle and leaves the shuttle in the lane. This operation differs from our system, since the forklift can pick up only the load or both the load and the shuttle in our paper.

We study this important problem in the multi-deep shuttle system. To do this, we assume a dedicated storage policy in which every lane stores only a single product. This policy is particularly used in production and distribution warehouses storing bulk products, such as fresh produce, general ambient and frozen food

products, vaccines, and packaging materials. As the shuttles are battery-powered, they need charging. We assume this is done offline, outside regular working hours. We also assume a single storage level in our base model, but extending the model to multiple levels is straightforward. We focus on retrievals rather than storage requests because these job types are typically split in deep-lane systems. Storage requests result from receiving a production batch or inbound truckload, and the loads must be stored in a few deep lanes. Retrieval requests come with time pressure and require high responsiveness, so optimal shuttle scheduling is important (Boysen, De Koster, & Weidinger, 2019; Zou, Xu, De Koster et al., 2016).

Given a set of retrieval requests and a number of shuttles, we aim to schedule forklifts (or transfer cars) and shuttles such that the makespan is minimized. Specifically, the following research questions are addressed:

- In what ways can we determine the optimal choices for (1) the sequence in which the shuttles are released in the lanes of retrieval requests by the forklift (or transfer car), (2) the sequence in which the retrieval requests are processed, and (3) the decision whether a shuttle is dropped off at the I/O point or kept in the lane for another request?
- What improvements can be achieved by this paper's proposed method compared to straightforward heuristics used in practice and in the literature?

Note that we choose the makespan (the total completion time) as the objective of the scheduling problem. For the system studied in our paper, minimizing makespan is a particularly good objective choice for several reasons. (i) According to Toyota and EAB (a Toyota subsidiary, see https://www.youtube.com/watch?v=hdZ_vQIpnJo), the system is particularly used by manufacturers producing in batch, logistics service providers handling and storing pallet batches, and deep-freeze warehouses. In these environments, work can be planned for a whole batch of pallets. This means that these warehouses can plan ahead, and individual pallets do not need prioritization. The most appropriate measure for this situation is the makespan. (ii) In deep-freeze warehouses, a key shuttle application area, a typical objective is to minimize the total time people have to work in cold circumstances.

The scheduling problems with the forklift and the transfer car are denoted as the Forklift-Shuttle Scheduling (FSS) problem and the Transfer car-Shuttle-Scheduling (TSS) problem. Note that the case of scheduling transfer cars and shuttles is a special case of FSS since when using a transfer car, the shuttle must always be dropped off at the I/O point, as the transfer car cannot leave the shuttle in the lane and move only the load. Therefore, this paper studies scheduling retrievals with a focus on the FSS problem with the forklift, where the shuttle should be either kept in the lane or returned to the I/O point.

We define a cycle as the movements of the forklift (FL) between two successive visits at the I/O point. Four types of such cycles can be distinguished: (i) the FL picks up and drops off one shuttle (without requested load); (ii) the FL only retrieves a requested item, with or without a shuttle; (iii) the FL first drops off a shuttle in a lane with a request and then moves to another lane to retrieve a request, with or without a shuttle; and (iv) the FL transfers a shuttle to the lane of a request, waits, and then just retrieves this request, with or without a shuttle. The cycles are performed one by one until the final retrieval request is finished.

One possible heuristic solution is using cycle (iv) for all requests, but then the performance can be quite poor since only one shuttle is used, despite multiple shuttles being available, and the waiting time of the forklift can be long. Another solution is to drop off all the shuttles in lanes with requests to ensure that these requests each have a shuttle to service them. Then the requests are retrieved according to the shuttle transfer sequence, i.e., first using cycle (i) and then (ii). This solution heuristic is better than the previous solution, but a large number of empty travels can still occur. Instead, in this study, we propose a new heuristic policy that uses cycles (i), (iii), and (ii), in sequence. That is, some shuttles are dropped off first. Subsequently, a joint shuttle transfer and load retrieval are performed. Finally, the remaining requested loads are retrieved. The heuristic has two stages. In the first stage, priority rules are used to determine the load retrieval sequence, whereas the shuttle transfer sequence in the second stage is solved optimally by a dynamic programming approach. We show that this two-stage heuristic has near-optimal performance.

The contribution of this research is twofold. First, we formulate a novel mathematical model that can be used to obtain the optimal shuttle transfer and load retrieval sequence and the resulting makespan for a set of retrieval requests We show that the problem is NP-hard, and to solve it, we propose an efficient two-stage heuristic to produce near-optimal solutions, even for large instances. The model can also be used to decide on the optimal shuttle fleet size, minimizing the total cost. Second, we provide managerial insights into the use of these systems based on real-case data. We show that our two-stage heuristic can decrease the makespan substantially compared with straightforward heuristics used in practice and in the literature.

The remainder of this paper is structured as follows: Section 2 reviews the related literature. Section 3 defines and models our research problem for deep-lane warehouse systems in detail and investigates the computational complexity. The two-stage heuristic is developed in Section 4. Subsequently, the computational performance of our solution methods is explored based on computational experiments in Section 5. Section 6 further extends the model in Section 3 to a multi-level system. Finally, Section 7 concludes the paper.

2. Literature review

The paper is relevant to two streams of literature: (i) deep-lane storage systems and (ii) sequencing retrieval requests with shuttles or vehicles.

The literature on deep-lane storage systems can be classified into two categories based on the different types of equipment used: crane-based and shuttle-based compact storage systems. Crane-based compact systems can be seen as multi-deep Automated Storage and Retrieval (AS/R) systems; they are equipped with a storage and retrieval crane (S/R machine) and an orthogonal transport mechanism. In such a system, the S/R crane can simultaneously move in both the horizontal and vertical directions of the rack, and the orthogonal transport mechanism carries out the depth movements (Azadeh et al., 2019). Two different depth-movement transport mechanisms are used in crane-based compact systems: conveyor-based and satellite-(or shuttle-) based systems.

Most literature studying crane-based compact systems focuses on design-related decisions, where the objective is to maximize the throughput capacity with given constraints. In doing so, closed-form expected travel time expressions are proposed and used to optimize the design choices. De Koster, Le-Duc, & Yu (2008) develop expected travel time models for both single-command and dual-command systems under the random storage policy in a conveyor-based compact storage system, and they determine the optimal system dimension by minimizing the expected travel time of the crane. Yu & De Koster (2009a) further extend this work by considering a full turnover-based storage policy. As a follow-up, Yu & De Koster (2009b) investigate the optimal system dimensions under a two-class-based storage policy. Hao, Yu, & Zhang (2015) and Yang, Miao, Xue, & Qin (2015) investigate the optimal rack configuration by considering possible locations of the

input/output point and taking into account the acceleration and deceleration of the S/R machine, respectively.

Literature on operational problems in crane-based compact systems is limited. The objective varies but typically is to minimize the makespan or the cycle time of a task. Yu & De Koster (2012) propose a heuristic approach called PPR-SL to sequence both storage and retrieval requests in the conveyor-based compact system for dual-command cycles to minimize the average cycle time. Their numerical results show that the PPR-SL approach outperforms approaches commonly used in practice by more than 20%. Zaerpour et al. (2015) formulate a model for satellite-based deep-lane systems and show how to minimize the total retrieval time (makespan) of crane-with-satellite deep-storage systems with one-sided access for a given set of requests. In contrast to a shuttle, a satellite is connected to the crane; while the satellite operates in the deep lane, the connected crane must wait in front of the lane.

Shuttle-based compact storage systems are popular because of their high and flexible capacity compared to crane-based compact systems. Lifts take care of the vertical movements, whereas shuttles are responsible for the movements within the storage lanes, operating in parallel. A transfer car is responsible for lateral movements. Tappia, Roy, De Koster, & Melacini (2017) focus on throughput time and study shuttle-based compact systems with lifts, transfer cars, and shuttles. They develop queuing network models to estimate performance. Their numerical results show that a depth/width ratio of around 1.25 leads to a minimum expected throughput time. Deng, Chen, Zhao, & Wang (2021) extend this work to include parallel movements of the transfer car and shuttles. Zou, Koster, & Xu (2018) evaluate dedicated and shared storage policies in robot-based compact storage and retrieval systems. Although the dedicated storage policy outperforms the shared storage policy in terms of the dual command throughput time, the annualized costs of dedicated storage are up to twice as large as those of shared storage. Boysen, Boywitz, & Weidinger (2018) extend the storage assignment problem of Zaerpour et al. (2015) for the satellite-based system to a forklift-with-shuttle based compact system with two-sided access of storage lanes. They also focus on load-to-storage location assignment where load blocking (i.e., a pallet with a later retrieval time is placed in front of another pallet with higher priority in the same lane) is avoided. Different from the crane-with-satellite system, the forklift in the shuttle-based system can move somewhere else after dropping off the shuttle in a lane and does not have to wait until the shuttle returns.

We also review the literature on sequencing problems. Sequencing retrieval requests, i.e., determining the sequence in which a forklift (or any other handling equipment) should retrieve a set of requests, is an elementary and important problem in any warehouse system. Papers on scheduling retrievals and storage jobs in single-deep storage systems are (somewhat) more abundant. Han et al. (1987) are the first to study the retrieval sequencing request problem for dual command cycles in a traditional AS/R system. Carlo & Vis (2012) investigate the scheduling problem to sequence storage and retrieval requests with two lifts in a shuttle-based storage and retrieval system. They develop an integrated look-ahead strategy to solve the problem. Experimental results show that it can achieve reductions averaging 44.3% in terms of makespan compared to the warehouse system with a single lift. Furthermore, Zhao et al. (2018) extend this work by considering the acceleration and deceleration of the lifts.

However, the amount of literature focusing on sequencing retrieval requests in multi-deep systems is limited. We are aware of only one paper by Dong et al. (2021). They study a retrieval task scheduling problem for a crane-based AS/RS with multiple shuttles with the objective of minimizing the makespan of completing a set of retrievals. They assume that the crane only picks up the load from the shuttle when the shuttle moves the load to the foremost location. This differs from our system, where rather than moving just the load, the forklift can decide to pick up only the load (leaving the shuttle in the lane) or both the load and shuttle and bring them to the depot, after which the shuttle can be taken to any other lane. The two-stage heuristic in our paper can contribute to more than 5% reduction in terms of makespan, compared to their Lowest-Waiting-Time-First heuristic.

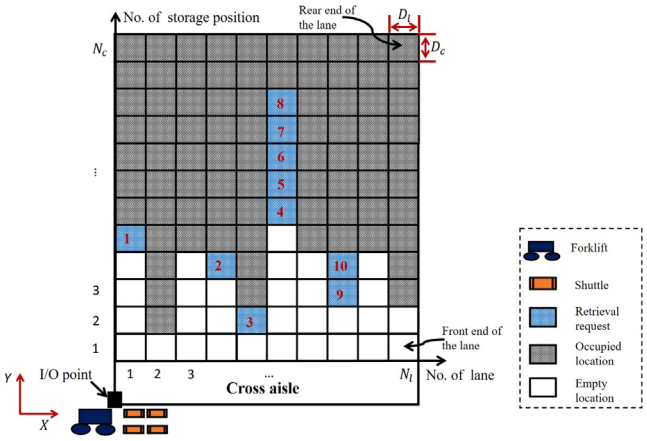


Fig. 3. Top view of a shuttle-based deep-lane warehouse system.

In summary, few papers focus on scheduling retrievals in shuttle-based deep-lane storage systems with forklifts. Also, the option that just the shuttle, just the load, or both can be picked up has not yet been studied.

3. Problem description and model formulation

In this section, we describe our research problem in Section 3.1, then define cycles and handling time in Section 3.2 and 3.3, respectively. Based on these, a mathematical model for minimizing the makespan is derived in Section 3.4.

3.1. Research problem

This subsection describes a deep-lane warehouse system (see Fig. 3). The warehouse is served by a forklift and multiple shuttles, operating in a single-command mode, i.e., one load is retrieved in each command. All the items enter and leave the warehouse via an input and output (I/O) point, which is located in the lower-left corner of the warehouse (see Fig. 3). A cross-aisle allows the forklift to travel horizontally to reach the lanes. The forklift can move shuttles between lanes. It operates parallel to the shuttles.

The shuttles carry out the movements within the storage lanes and move the loads to the front end of the lanes. The forklift transports the loads from the front end of the lanes to the I/O point. A retrieval request relies on the collaboration between a shuttle and the forklift. We assume that shuttles are parked at the I/O point initially. Hence, before retrieval, each request must reserve a shuttle. We call the process of the forklift moving a shuttle to the front end of a storage lane a *shuttle transfer* process, and the transport of the load to the I/O point a *load retrieval* process. This paper studies the optimal sequence of the joint shuttle transfer and load retrieval, such that the makespan of a set of requests is minimized. The assumptions are as follows:

(1) Each lane holds one type of product (similar to, e.g., Anken, Gagliardi, Renaud, & Ruiz, 2011; Manzini, Accorsi, Baruffaldi, Cennerazzo, & Gamberi, 2016; Zaerpour et al., 2015; Zou et al., 2018). When more than one load needs to be retrieved in the same lane, the loads are retrieved in last-in-first-out

(LIFO) sequence from the near location to the far location. This is typical in e.g., production warehouses, where products are produced in lots which have to be separated into different lanes. We do not consider reshuffle operations.
(2) All shuttles are located at the I/O point initially. Our model and method can also be applied in the setting where some shuttles are located in the lanes initially.
(3) The operations of the forklift start from and end at the I/O point.
(4) When multiple remaining requests exist in the same lane, the forklift leaves the shuttle in that lane (except for the last request) rather than bringing it back to the I/O point buffer. Note that this is not a strong assumption because, in practice, the drop-off and pick-up are time-consuming. Picking up a shuttle from another lane, moving it to the current lane, and dropping it off again will always take more time than just retrieving the load in the current lane. Any practical value of the pick-up or drop-off time leads to it being suboptimal to get another shuttle to the current lane.
(5) To simplify the request scheduling process, we assume that the shuttle must wait for the handover of a load by the forklift at the front end of a lane. A similar assumption is made by Zou et al. (2016). We later relax this assumption and assume the shuttle can continue retrieving loads in the same lane (see Extensions in Section 6.2).

3.2. Definition of cycles

Consider a set $R = \{1, 2, \dots n\}$ of requests. For each request $i \in R$, the lane (l_i), and storage position (c_i) are known. We define the I/O point as l_0. In this study, we define a cycle as the movements of the forklift between two successive visits at the I/O point. A cycle is a 5-tuple (l_i, l_j, h, l_h, k) or (l_i, l_j, h, l_0, k), where $l_i \neq l_j$, and $i \in R \cup \{0\}$, $j, h \in R$, indicating that a forklift starts from the I/O point, transfers the shuttle from the lane of request i, l_i, to the lane of request j, l_j, and then retrieves requested load h in cycle k. The only difference between the two cases of 5-tuples is the fourth element in the tuples, representing the shuttle's location after it has retrieved load h to the front end of its lane. If the forklift picks up only load h to bring it to the I/O point and leaves the shuttle in the lane of request h in cycle k, then the fourth element in the tuple is l_h (the shuttle is in the lane of request h); if the forklift picks up the load h and the shuttle in the lane of request h to bring them to the I/O point, then the fourth element in the tuple is l_0 (the shuttle is at the I/O point).

Further, the forklift may transfer a shuttle from the I/O point to the lane with requests directly in a shuttle transfer process. In the cycle (l_0, l_j, h, l_h, k), $j, h \in R$, the forklift picks up a shuttle at the I/O point, transfers the shuttle to the lane of request j, l_j, then retrieves the requested load h. It picks up only the load h and leaves the shuttle in the lane of request h, while bringing the load h to the I/O point. Since a cycle may consist of only a shuttle transfer or only a load retrieval, we define request "$n+1$" as a virtual request. Then the cycle with only shuttle transfer (when $l_i \neq l_j$) is denoted by the 5-tuple $(l_i, l_j, n+1, l_{n+1}, k)$, where $l_i \neq l_j, i \in R \cup \{0\}, j \in R$. Similarly, the cycle with only load transfer is denoted by the 5-tuple $(l_{n+1}, l_{n+1}, h, l_h, k)$ or $(l_{n+1}, l_{n+1}, h, l_0, k)$, where $h \in R$. Three types of cycles can be obtained based on whether a shuttle is transferred or a load is retrieved in a cycle. These cycles are denoted by cases A, B, and C.

(1) *Shuttle transfer and load retrieval in a cycle (Case A)*: This is the most general cycle. The forklift performs both a shuttle transfer and a load retrieval, that is (l_i, l_j, h, l_h, k) or (l_i, l_j, h, l_0, k), where $l_i \neq l_j$, $i \in R \cup \{0\}$, and $j, h \in R$.
(2) *Only shuttle transfer in a cycle (Case B)*: Only a shuttle transfer is performed in a cycle, that is $(l_i, l_j, n+1, l_{n+1}, k)$, where $l_i \neq l_j$, $i \in R \cup \{0\}$, and $j \in R$.
(3) *Only load retrieval in a cycle (Case C)*: Only a load retrieval is performed in a cycle, that is $(l_{n+1}, l_{n+1}, h, l_h, k)$ or $(l_{n+1}, l_{n+1}, h, l_0, k)$, where $h \in R$.

Whether the forklift leaves the shuttle in the lane or moves both the load and shuttle to the I/O point (the fourth element in a cycle is l_h or l_0), the location of that shuttle (the I/O point l_0 or the lane l_h) affects the movements of the forklift in a cycle. Thus, several subcases can be distinguished to describe unique movements of the forklift for completing cycle k, as described in Appendix F.

3.3. Defining the handling time

We assume constant times are required for the forklift to pick up or drop off the shuttle, the load, or both the shuttle and the load. The same holds for the time for the shuttle to pick up or drop off a load. This is not a major restriction, as the time is also close to constant in practice (see, e.g., Tappia et al. (2017)). The unit handling time for the forklift, transfer car, and shuttle are summarized in Table 1. The total handling time depends on the tasks performed in a cycle.

The total handling time for different types of cycles are summarized in Appendix G.

3.4. Model formulation

In this section, we formulate a mathematical programming model for the FSS problem. Then, we prove that problem FSS is NP-hard. The notation used in the mathematical model is presented in Table 2, including the sets, parameters, and decision variables.

Based on the parameters, the processing time of the shuttle to retrieve the load i, (denoted by p_i), consists of the following three components:

(1) Travel time of the shuttle to go from the front end of the lane (position 1) to the storage location (position c_i) of load i. This time equals $(c_i - 1) \times D_c / v_s$.
(2) Time for the shuttle to pick up the load i (t_o).
(3) Travel time of the shuttle to bring the load i from the storage location (position c_i) to the front end of the lane (position 1). This time also equals $(c_i - 1) \times D_c / v_s$.

Thus, the processing time of the shuttle to retrieve the load i can be calculated by

$$p_i = 2(c_i - 1) \times D_c / v_s + t_o, \qquad i \in R. \tag{1}$$

Since we consider only one storage level, the horizontal travel time for the forklift to go from the I/O point to the lane l_i of load i can be calculated by

$$d_i = (l_i - 1) \times D_l / v_f, \qquad i \in R. \tag{2}$$

The completion time of a cycle depends on the type of the cycle. We treat cycle (l_i, l_j, h, l_h, k) as an example, where $l_i \neq l_j$, $i \neq h$ and $j \neq h$. Here the forklift travels from the I/O point to the lane of request i, transfers the shuttle from the lane of request i to the lane of request j, and then moves to the lane of request h with arrival time $\sum_{q=1}^{k-1} f_q + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|$. The time when the shuttle is dropped off at the lane of request h is a_h. Since the shuttle processing time within the lane is p_h, the waiting time of the forklift is given by $\max\{0, a_h + p_h - (\sum_{q=1}^{k-1} f_q + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|)\}$. After that, the forklift goes back to the I/O point taking travel time d_h and handling time $2t_u$ since the fourth element in the cycle (l_i, l_j, h, l_h, k) is l_h (i.e., $m_h = 1$). Consequently, the required time of the cycle (l_i, l_j, h, l_h, k) is

if $x_{ijk} = 1, y_{hk} = 1, m_h = 1 \rightarrow$

Table 2
Notation.

Parameters	Description
N_s	Number of shuttles.
n	Number of requests.
R	Set of requests, with indices $i, j, h \in R$. In the following, we define $\bar{R} = R \cup \{0\} \cup \{n+1\}$, and $\underline{R} = R \cup \{n+1\}$, where request 0 and request $n+1$ represent the I/O point and the virtual request, respectively.
R_1, R_2, R_3, R_4	R_1 denotes the set of requests in all lanes with one request, R_2 denotes the set of leading requests in lanes with multiple requests, R_3 denotes the set of trailing requests in lanes with multiple requests (≥ 3) excluding the last request. When the number of requests equals 2, for all lanes, $R_3 = \emptyset$. R_4 denotes the set of last requests in lanes with multiple requests. We have $R = R_1 \cup R_2 \cup R_3 \cup R_4$ and $R_m \cap R_n = \emptyset$, for $\forall m \neq n, m, n \in \{1, 2, 3, 4\}$. As an example: there are 10 requests in Fig. 3, where $R = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $R_1 = \{1, 2, 3\}$, $R_2 = \{4, 9\}$, $R_3 = \{5, 6, 7\}$, and $R_4 = \{8, 10\}$.
N_r	Number of lanes with requests.
K	Set of cycles, with index $k \in K$.
N_c, N_l	Number of storage columns and lanes.
v_f, v_s, v_t	Velocity of forklift, transfer car, and shuttle. We assume that the forklift, transfer car, and shuttle move at constant speeds and ignore the acceleration and deceleration.
D_c, D_l	Width and length of a storage location, respectively.
c_i, l_i	The location of the column and the lane of request i.
e_h	The request stored in the location in front of request h. $e_h = 0$ means that h is the leading request in the lane.
α_{ij}	Parameter, = 1 if request i and j are in the same lane and the location of request i is in front of that of request j, 0 otherwise.
M	Big positive number.
f_k	Time to complete cycle k.
Decision variables	Description
x_{ijk}	= 1 if the forklift transfers the shuttle from request i's lane to request j's lane in cycle k, and 0 otherwise, $i \in \bar{R}, j \in \underline{R}, k \in K$. If $i = 0, j \in R$, the forklift transfers the shuttle from the I/O point to request j's lane in cycle k. If $i = j = n+1$, there is no shuttle transfer process in cycle k.
y_{hk}	= 1 if request h is completed in cycle k and 0 otherwise. If $h = n+1$, there is no load retrieval process in cycle k, $h \in \underline{R}, k \in K$.
m_h	= 1 if the shuttle is left in the lane of request h, and 0 if it is dropped off at the I/O point buffer after completing the request h, $h \in R$.
a_j	Time that the shuttle became available to retrieve load j in lane l_j, $j \in R$.
β_{jk}	Auxiliary binary variable, =1 if $y_{jk} = 1$ and $m_j = 0$.

$$f_k = d_i + |d_i - d_j| + 2t_s + |d_j - d_h| + \max\{0, a_h + p_h - (\sum_{q=1}^{k-1} f_q + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|)\} + d_h + 2t_u$$

$$= \max\{d_i + |d_i - d_j| + 2t_s + |d_j - d_h|, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u, i \in \bar{R}, j, h \in R, k \in K. \quad (3)$$

Constraints (3) are written as logical constraints, which can be directly used in CPLEX. Another possible way is to linearize them, as follows.

$$\max\{d_i + |d_i - d_j| + 2t_s + |d_j - d_h|, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u \leq f_k + M(3 - x_{ijk} - y_{hk} - m_h) \quad \forall i \in \bar{R}, j, h \in R, k \in K. \quad (4)$$

The time needed for the completion of other cycles can be calculated similarly. The objective (5) of model FSS is to minimize the makespan, i.e., the time taken to complete all retrieval requests. The model for FSS is formulated as follows.

$$\min \sum_{k \in K} f_k, \quad (5)$$

$$s.t. \quad \sum_{i \in \bar{R}} \sum_{k \in K} x_{ijk} = 1, \quad \forall j \in R, \quad (6)$$

$$\sum_{i \in \bar{R}} \sum_{j \in \underline{R}} x_{ijk} = 1, \quad \forall k \in K, \quad (7)$$

$$\sum_{j \in \underline{R}} \sum_{k \in K} x_{ijk} \leq 1, \quad \forall i \in R, \quad (8)$$

$$\sum_{k \in K} y_{hk} = 1, \quad \forall h \in R, \quad (9)$$

$$\sum_{h \in \underline{R}} y_{hk} = 1, \quad \forall k \in K, \quad (10)$$

$$\sum_{j \in R} x_{0j1} = 1 \quad (11)$$

$$x_{ijk} \leq \sum_{p=1}^{k-1} y_{ip}, \quad \forall i, j \in R, k \in \{2, \ldots, K\} \quad (12)$$

$$\sum_{i \in \bar{R}} \sum_{k \in K} x_{ijk} + 1 \leq \sum_{k \in K} y_{jk}, \quad \forall j \in R, \quad (13)$$

$$\sum_{k \in K} y_{ik} + 1 \leq \sum_{k \in K} y_{jk}, \quad \forall i, j \in R, \alpha_{ij} = 1, \quad (14)$$

$$\sum_{i \in \bar{R}} \sum_{k \in K} x_{ijk} \leq m_j, \quad \forall j \in R, \quad (15)$$

$$x_{n+1,n+1,k} + y_{n+1,k} \leq 1, \quad \forall k \in K, \quad (16)$$

$$y_{jk} - m_j \leq \beta_{jk}, \quad \forall j \in R, k \in K, \quad (17)$$

$$\beta_{jk} \leq y_{jk}, \quad \forall j \in R, k \in K, \quad (18)$$

$$\beta_{jk} \leq 1 - m_j, \quad \forall j \in R, k \in K, \quad (19)$$

$$\sum_{k=1}^{q} \sum_{j \in R} x_{0jk} - \sum_{k=1}^{q} \sum_{j \in R} \beta_{jk} \leq N_s, \quad \forall q \in \{N_s, \ldots, |K|\}, \quad (20)$$

$$\sum_{t=1}^{k-1} f_t + d_i + |d_i - d_j| + 2t_s \leq a_j + M(1 - x_{ijk}), \quad i \in \bar{R}, j \in R_1 \cup R_2, k \in K, \tag{21}$$

$$\sum_{t=1}^{m} f_t - (d_j + t_u) \leq a_j + M(1 - y_{qm}), j \in R_3 \cup R_4, m \in K, q = e_j, \tag{22}$$

$$2\max\{d_i, d_j\} + 2t_s \leq f_k + M(2 - x_{ijk} - y_{n+1,k}), \quad i \in \bar{R}, j \in R, k \in K, \tag{23}$$

$$\max\{d_h, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u \leq f_k + M(3 - x_{n+1,n+1,k} - y_{hk} - m_h), \quad h \in R, k \in K, \tag{24}$$

$$\max\{d_h, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u + 2t_c \leq f_k + M(2 - x_{n+1,n+1,k} - y_{hk} + m_h), \quad h \in R, k \in K, \tag{25}$$

$$\max\{d_i + |d_i - d_j| + 2t_s + |d_j - d_h|, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u \leq f_k + M(3 - x_{ijk} - y_{hk} - m_h), \quad i \in \bar{R}, j, h \in R, k \in K. \tag{26}$$

$$\max\{d_i + |d_i - d_j| + 2t_s + |d_j - d_h|, a_h + p_h - \sum_{q=1}^{k-1} f_q\} + d_h + 2t_u + 2t_c \leq f_k + M(2 - x_{ijk} - y_{hk} + m_h), \quad i \in \bar{R}, j, h \in R, k \in K. \tag{27}$$

$$x_{ijk}, y_{hk}, m_h \in \{0, 1\} \quad \forall i \in \bar{R}, j, h \in \underline{R}, k \in K. \tag{28}$$

Constraints (6) ensure that each request has a shuttle and is included in one cycle. Constraints (7) state that in each cycle, at most one shuttle is transferred and one request is retrieved. Constraints (8) imply that the shuttle in a lane can be transferred to another lane at most once. Similarly, Constraints (9) and (10) ensure that each request must be completed once, and at most one request is finished in each cycle, respectively. Constraints (11) ensure that in the first cycle, the forklift must transfer the shuttle from the I/O point. Constraints (12) make sure that before leaving a lane, a shuttle has already finished the request. Similarly, Constraints (13) guarantee that a request can be retrieved unless a shuttle is in the lane of this request. The precedence constraints among requests in the same lane are ensured in Constraints (14). In Constraints (15) ensures that only when the shuttle is kept in the lane can it be transferred to other lanes. An empty cycle, i.e., neither a shuttle transfer nor a load retrieval process, is avoided (Constraints (16)). Constraints (17)-(19) require that $\beta_{jk} = 1$ only when $y_{jk} = 1$ and $m_j = 0$. Constraints (20) imply that the number of shuttles in the system is limited by N_s. The first term on the left side is the number of shuttles released to the system in the first p cycles, whereas the second term represents the number of loads finished in the first p cycles. Since the shuttle either stays at that lane or returns to the I/O point, the upper bound of the number of shuttles in the systems is $\sum_{k=1}^{q}\sum_{j\in R} x_{0jk} - \sum_{k=1}^{q}\sum_{j\in R}\beta_{jk}$. Constraints (21–27) define the relationship between a_h and f_k. Constraints (28) define the domain of the decision variables. The complexity status of FSS is settled by the following theorem.

Table 3
Priority rules for determining the load retrieval sequence.

Name	Description	Priority value
SPT	Shortest processing time	$\min\{p_h, h \in R\}$
STT	Shortest horizontal travel time	$\min\{d_h, h \in R\}$
SDT	Shortest duration time	$\min\{p_h - d_h, h \in R\}$

Theorem 1. *The FSS problem is NP-hard.*

Proof. See Appendix I.1 □

4. Two-stage heuristic approach

In this section, we introduce a two-stage heuristic approach to solve the FSS problem for the case with multiple shuttles.

When there is one shuttle in the warehouse system, the optimal solution is provided in Appendix C. When the number of shuttles is larger than one, we propose a two-stage heuristic algorithm. In the first stage, the load retrieval sequence is determined according to rule-based methods. The load retrieval sequence is then handed over to the second stage, where the optimal shuttle transfer sequence is obtained by a dynamic programming approach.

We propose three priority rules to determine the load retrieval sequence of the forklift, as shown in Table 3. The retrieval sequence of request h is based on increasing priority values. Using rule SPT (shortest processing time), the requests with least shuttle processing time are retrieved by the forklift first. In rule STT (shortest horizontal travel time), the forklift retrieves the requests in lanes closer to the I/O point first. Rule SDT (shortest duration time) makes use of parallel movements of the forklift and shuttles. This reduces the waiting time of the forklift for the shuttle bringing the load to the front end of the lane. The requests with smaller duration times are retrieved first.

Then, for each cycle, we determine whether the shuttle should be kept in the lane or dropped off at the I/O point buffer.

When multiple remaining requests exist in the same lane, we assume that the forklift leaves the shuttle in that lane. In other words, we assume that the forklift picks up only the requested load h when $h \in R_2 \cup R_3$. We then propose a heuristic to determine whether the shuttle should be kept in the lane or dropped off at the I/O point buffer after completing request h when $h \in R_1 \cup R_4$. We define w_h as the number of requests located in lanes farther than the lane of request h, and u_h as the number of requests located in lanes closer to the I/O point than the lane of request h. The heuristic to determine whether the shuttle should be left in the lane when request h is retrieved is as follows. When request h is retrieved, if $w_h \geq u_h$, the forklift picks up only the requested load h, and the shuttle is left behind in the lane. Otherwise, when $w_h < u_h$, the forklift picks up both the requested load h and the shuttle and brings them to the I/O point. Once the load retrieval sequence of requests and the shuttle movement decisions of the forklift operation are given, Dynamic Programming (DP) can be applied to obtain the optimal shuttle transfer sequence.

Furthermore, we use DP to determine the transfer sequence in each cycle. Recall the cycle (l_i, l_j, h, l_h, k), where the forklift transfers the shuttle from the lane of request i to the lane of request j, then retrieves request h in the load retrieval of cycle k. The retrieval sequence of the loads and the decisions of whether the forklift leaves the shuttle in the lane are determined based on the rule-based heuristic.

We now determine the selection of the number of cycles $|K|$. When the forklift performs a shuttle transfer and a load retrieval in each cycle, the number of cycles equals the number of requests, i.e., $|K| = n$. However, when the forklift performs only a shuttle transfer or only a load retrieval in each cycle, the number of cycles equals two times the number of requests, i.e., $|K| = 2n$. Thus, the range of the number of cycles is given by $n \le |K| \le 2n$. We vary the number of cycles $|K|$ from n to $2n$. For each value of $|K|$, the DP is used to determine the shuttle transfer sequence by minimizing the makespan. Among all solutions, we find the minimum makespan, which is the optimal solution to the scheduling problem with the given retrieval sequence. We assume that request "$n+1$" represents a virtual retrieval (i.e., an empty shuttle transfer), which we add to the load retrieval sequence, so the number of elements in the "new" load retrieval sequence equals $|K| > n$. We allow virtual request $n+1$ to appear multiple times in the load retrieval sequence. For example, consider request set $R = \{1, 2, 3\}$, and $|K| = 5$ cycles. According to the priority rules, the load retrieval sequence is $(2, 3, 1)$. We then add virtual request 4 to the load retrieval sequence and allow virtual request 4 to appear twice. This virtual request can be added at arbitrary positions in the load retrieval sequence. In the new load retrieval sequence $(4, 2, 4, 3, 1)$, only shuttle transfers are performed (without load retrievals) in the first and third cycles. Such virtual requests can be added at arbitrary positions in the load retrieval sequence. However, from our pre-experiments, we found that the most effective way is to add the virtual requests at the beginning of the load retrieval sequence. This method is adopted in the following numerical experiments.

We now have to select a free shuttle to be transferred when the lane of request j does not contain a shuttle. A shuttle becomes free only after it has completed all the requests in its lane or when it is initially located at the I/O point. When a shuttle transfer is required, a free shuttle is selected. When multiple free shuttles are available, we give priority to a shuttle at the I/O point. If the shuttle transfer must be carried out in cycle k but no free shuttles are located at the I/O point, then we select the shuttle freed most recently in the previous cycles.

Given $|K|$ cycles, there are $|K|$ stages in total. State V_k at stage k represents a subset V_k of requests to whose lanes shuttles have been transferred in the first k cycles. The function $g(V_k)$ represents the completion time of the first k cycles. A transition from state V_{k-1} to state V_k is defined if $V_{k-1} \subset V_k$ and $V_k - V_{k-1} = \{j\}$, $j \in R \cup \{n+1\}$. The recursions can be formalized as follows:

$$g(V_k) = \min_{j \in V_k} \{g(V_k \backslash \{j\}) + c(j, V_k)\}, \tag{29}$$

with the initialization $g(\{\emptyset\}) = 0$. The objective is to find $g(V_{|K|})$.

The additional value associated with a transition from V_{k-1} to V_k, denoted by $c(j, V_k)$, can be decomposed into three cases based on types of cycles, denoted by cases A, B, and C. In case A, the forklift completes a cycle with both a shuttle transfer and a load retrieval; in case B, the forklift completes a cycle with only a shuttle transfer; in case C, the forklift completes a cycle with only a load retrieval. Several subcases can be distinguished, each corresponding to a unique formula for the completion time of cycle k. Here, we only present the additional value associated with a transition of subcase A1. The other subcases are given in Appendix H.

In subcase A1 with a cycle (l_i, l_j, h, l_0, k), $l_i \neq l_j$, $l_j \neq l_h$, the forklift first travels from the I/O point to the lane of request i, transfers the shuttle from the lane of request i to the lane of request j, then moves to the lane of request h, and arrives at the lane of request h at the arrival time $g(V_k \backslash \{j\}) + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|$. Let the time when the shuttle became available to retrieve load h in lane l_h be a_h. After this time, multiple loads have been retrieved by this shuttle in lane l_h. Since the shuttle processing time within the lane is p_h, the waiting time of the forklift is given by $\max\{0, a_h + p_h - (g(V_k \backslash \{j\}) + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|)\}$. After that, the forklift picks up both the load and shuttle and then goes back to the I/O point using travel time d_h and handling time $2t_u + 2t_c$. Consequently, the required time of cycle (l_i, l_j, h, l_0, k) is

$$\begin{aligned} c(j, V_k) &= d_i + |d_i - d_j| + 2t_s + |d_j - d_h| + \max\{0, a_h + p_h \\ &\quad - \big(g(V_k \backslash \{j\}) + d_i + |d_i - d_j| + 2t_s + |d_j - d_h|\big)\} \\ &\quad + d_h + 2t_u + 2t_c \\ &= \max\{d_i + |d_i - d_j| + 2t_s + |d_j - d_h|, a_h + p_h - + g(V_k \backslash \{j\})\} \\ &\quad + d_h + 2t_u + 2t_c. \end{aligned} \tag{30}$$

We still need to determine a_j, the time that the shuttle becomes available to retrieve load j in lane l_j. We find that, for cycle (l_i, l_j, h, l_0, k) or (l_i, l_j, h, l_h, k),

$$a_j = \begin{cases} g(V_k \backslash \{j\}) + d_j + 2t_s, & i = 0,\ j \in R_1 \cup R_2, \\ g(V_k \backslash \{j\}) + d_i + |d_j - d_i| + 2t_s, & i \neq 0,\ j \in R_1 \cup R_2, \\ g(V_b) - d_j - t_u, & j \in R_3 \cup R_4, \end{cases}$$

where $g(V_b)$ is the completion time of the first b cycles and where the request in front of request j is retrieved by the forklift in stage b (or cycle b). We provide an example of a DP solution in Appendix B. The complexity of the DP is described in the following.

Proposition 1. *The complexity of the DP procedure is $O({N_r}^2 \times 2^{N_r})$, where N_r is the number of lanes with requests.*

Proof. See Appendix I.2. □

5. Computational results

This section presents computational results and obtains insights from them. The experiments are run on a computer with an AMD Ryzen ThreadRipper 1950X 3.4 GHz processor and 64 GB of random access memory operating under Windows 10. The models are solved using CPLEX version 12.5. We first introduce the input data and evaluate the performance of the two-stage heuristic. Then we compare the computational results of the two-stage heuristic with some straightforward heuristics used in practice and in the literature. In addition to makespan, we study the cost minimization problem under all five policies in Section 5.3. The purpose is to find the optimal number of shuttles, lanes, and storage positions per lane, such that the total cost (consisting of the cost of shuttles and storage positions) is minimized. We also apply the two-stage heuristic to the deep-lane storage systems with transfer cars in Section 5.4.

5.1. Performance of the two-stage heuristic

We collected data from a real warehouse (courtesy of Toyota.com). The warehouse uses forklifts, but we also obtained corresponding data for a transfer car situation. We obtained data on the warehouse size, the velocity of the shuttle, forklift, and transfer car, and the unit handling time; see Table 4. Using these parameters, we can obtain the shuttle processing time p_i and travel time d_i based on Equations (1) and (2) in Section 3.

In the two-stage heuristic, we generate three load retrieval sequences based on the three priority rules described in Section 4 and then find the optimal shuttle transfer sequence based on these load retrieval sequences. We compare the solution values under these three load retrieval sequences and choose the best one as the solution result of the two-stage heuristic.

We investigate the computational performance for two problem instance sizes: small instances that are solvable to optimality by CPLEX with the computation time limit of 1800 seconds and large instances solved by the two-stage heuristic. If CPLEX cannot find the optimal solution within 1800 seconds,

Table 4
Parameters used in the computational experiments .

Parameter	Values
Small warehouse size (single level)	10 lanes × 17 storage positions per lane
Large warehouse size (single level)	20 lanes × 40 storage positions per lane
Width of a storage location (D_c)	1.4 m
Length of a storage location (D_l)	1.2 m
Velocity of the forklift (horizontal direction) (v_f)	1.7 m/s
Velocity of the shuttle (v_s)	0.9 m/s
Unit handling time of the shuttle (t_o)	3 s
Unit handling time of the forklift (t_s, t_u, and t_c)	15 s

The parameters are chosen based on real warehouse data provided by Toyota.com.

Table 5
Computational results for the FSS problem .

$\|R\|$	CPLEX		C_{LB} (s)	Two-stage heuristic		Gap	Gap_{LB} (s)
	C_{CPLEX} (s)	CPU time (s)		C (s)	CPU time (s)		
Small-size warehouse: 10 lanes × 17 locations per lane							
6	$412.6_{\pm 23.5}$	3.1	$386.4_{\pm 16.4}$	$422.8_{\pm 36.8}$	0.01	2.4%	8.6%
8	$546.0_{\pm 30.4}$	50.6	$514.1_{\pm 28.9.4}$	$560.6_{\pm 38.1}$	0.01	2.6%	8.3%
10	$689.8_{\pm 35.9}$	449.8	$637.1_{\pm 31.2}$	$701.7_{\pm 45.5}$	0.5	1.7%	9.2%
12	$809.6_{\pm 45.6}$	1246.8	$775.1_{\pm 39.7}$	$841.6_{\pm 51.9}$	1.6	3.8%	7.9%
15	$1125.8_{\pm 49.3}$	1800	$993.1_{\pm 42.6}$	$1054.2_{\pm 62.1}$	2.8	-6.8%	5.8%
18	$1395.6_{\pm 85.3}$	1800	$1189.2_{\pm 47.8}$	$1266.5_{\pm 79.0}$	5.9	-10.2%	6.1%
20	$1510.2_{\pm 90.4}$	1800	$1299_{\pm 61.5}$	$1394.5_{\pm 88.3}$	7.3	-8.3%	6.8%
25	$1868.3_{\pm 132.6}$	1800	$1640.4_{\pm 81.1}$	$1739.6_{\pm 114.2}$	12.5	-7.4%	5.7%
Large-size warehouse: 20 lanes × 40 locations per lane							
10	$809_{\pm 52.4}$	487.3	$749.2_{\pm 38.5}$	$818.9_{\pm 72.6}$	1.2	1.2%	8.5%
15	$1371.4_{\pm 84.5}$	1800	$1166.2_{\pm 80.3}$	$1259.4_{\pm 94.9}$	4.9	-8.9%	7.4%
20	$1824.9_{\pm 100.4}$	1800	$1524.5_{\pm 87.7}$	$1641.1_{\pm 108.8}$	6.7	-11.2%	7.1%
25	$2251.2_{\pm 125.7}$	1800	$1926.3_{\pm 108.9}$	$2055.9_{\pm 150.7}$	10.1	-9.5%	6.3%
30	$2750.6_{\pm 168.9}$	1800	$2322.1_{\pm 158.9}$	$2449.4_{\pm 189.8}$	27.4	-12.3%	5.2%
35	$3213.5_{\pm 200.4}$	1800	$2658.6_{\pm 156.3}$	$2831.3_{\pm 213.4}$	61.2	-13.5%	6.1%
40	$3674.4_{\pm 231.5}$	1800	$3073.9_{\pm 197.4}$	$3228.9_{\pm 240.6}$	113.5	-13.8%	4.8%
50	$4650.5_{\pm 289.4}$	1800	$3868.3_{\pm 186.3}$	$4033.4_{\pm 323.9}$	395.1	-15.3%	4.1%

*Note. Column C_{CPLEX} represents the average objective values obtained by truncated CPLEX. Column C_{LB} represents the average lower bound values. Column C represents the average objective values of the two-stage heuristic.

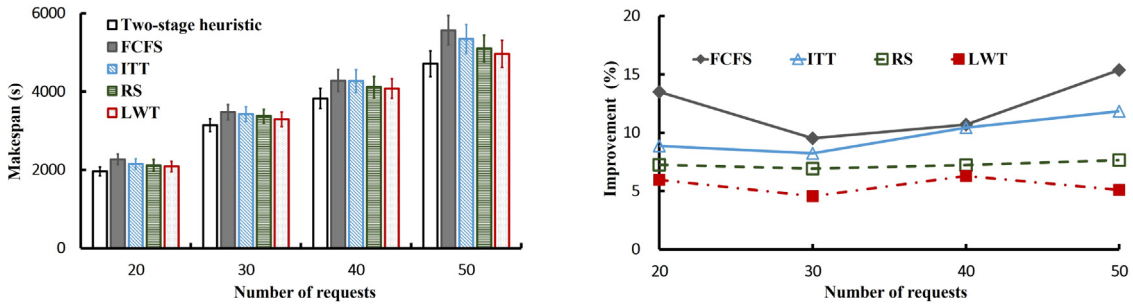
then we use the best CPLEX solution found within that limit. We set the number of requests for small and large warehouses to $\{6, 8, 10, 12, 15, 18, 20, 25\}$ and $\{10, 15, 20, 25, 30, 35, 40, 50\}$, respectively. Each instance with a given number of requests was solved ten times. It is interesting to have a lower bound, particularly for large problem instances. A straightforward lower bound that ignores the waiting time of the forklift in each cycle is given by Proposition 2 in Appendix E. We use that lower bound to evaluate the performance of the two-stage heuristic. We use Gap (or Gap_{LB}) to evaluate the relative difference between the objective values obtained by the two-stage heuristic and CPLEX (or a lower bound calculated by Equation (A1) in Appendix E). The value of Gap is obtained as $\frac{C-C_{CPLEX}}{C_{CPLEX}} \times 100\%$, averaged over all replications. Gap_{LB} is computed as $Gap_{LB} = \frac{C-C_{LB}}{C_{LB}} \times 100\%$. Table 5 gives the computational time and makespan for instances with four shuttles, including the 95% confidence intervals. CPLEX can find the optimal solutions in small instances (i.e., when the number of requests $|R|$ is at most 12). Our two-stage heuristic is within a 3.8% gap from the optimal solution obtained by CPLEX for these small instances. The average computation time by using the two-stage heuristic is less than 2 seconds, which is much smaller than that of CPLEX. When the number of requests is large (i.e., when $|R| > 12$), CPLEX fails to find an optimal solution within the time limit of 1800 seconds. We observe that the two-stage heuristic yields better solutions than CPLEX truncated after 1800 seconds, but these CPLEX solutions could be far from optimal, so we compare our solutions also to the lower bound. The relative difference Gap_{LB} in makespan between the two-stage heuristic and the lower bound of Equation (A1) in Appendix E is no more than 9%.

5.2. Comparison with other heuristics

This section focuses on four policies to solve the FSS (Forklift-Shuttle Scheduling) problem: the Retrieval Sequence (RS) policy, the Increasing Travel Time (ITT) policy, the Lowest-Waiting-Time-First (LWT) policy, and the First Come First Serve (FCFS) policy. We compare the makespan between the two-stage heuristic and these policies.

(1) Retrieval Sequence (RS) policy: The sequence of requests in the load retrieval process is determined by a priority rule (referred to Table 3). We now focus on how to produce the transfer shuttle sequence. Given a load retrieval sequence, instead of using the dynamic programming approach to obtain the optimal sequence of shuttle transfers, the RS sequence of transferring shuttles to the lane of requests is identical to the load retrieval sequence (without virtual requests). Consider an example with five requests $R = \{1, 2, 3, 4, 5\}$. Assume using a priority rule leads to the load retrieval sequence (2,5,4,3,1). Then, the RS sequence in the shuttle transfer process is also $(l_2, l_5, l_4, l_3, l_1)$. The cycles with only load retrievals are added to the end. That is, given seven cycles, the total transfer sequence under RS is $(l_2, l_5, l_4, l_3, l_1, l_6, l_6)$.

(2) Increasing-Travel-Time (ITT) policy: Similar to the RS policy, we also use a priority rule of Table 3 to determine the ITT sequence of load retrievals. The sequence in the shuttle transfer process is sorted based on increasing travel time d_i. For example, assume there are five requests with $R = \{1, 2, 3, 4, 5\}$, seven cycles, and we have $d_5 < d_2 < d_1 < d_4 <$



(a) Average makespan and 95% confidence intervals

(b) Average makespan improvement of the two-stage heuristic

Fig. 4. Effect of number of requests on makespan.

d_3. Then, the ITT sequence in the shuttle transfer process is $(l_5, l_2, l_1, l_4, l_3)$. Cycles with only load retrievals are added to the end, implying that the total shuttle transfer sequence is $(l_5, l_2, l_1, l_4, l_3, l_6, l_6)$.

(3) Lowest-Waiting-Time-First (LWT) policy: According to Dong et al. (2021), this policy can yield a better solution than a genetic algorithm in their setup. The key idea is to select the lane with the lowest forklift waiting time in each cycle so that the total waiting time is reduced. The authors give a shuttle transfer process a percentage priority α, meaning that a forklift performs a shuttle transfer with a probability α, or a load retrieval with a probability $1-\alpha$, for each cycle. Due to randomness, the results may differ each time for the same instance, so we repeat each instance ten times, and select the best result as the solution result.

(4) First-come-first-served (FCFS) policy: FCFS is the policy most frequently used in practice (Dong et al., 2021; Gharehgozli et al., 2017) and employed by the case warehouse (data courtesy of Toyota.com). It assumes that both the shuttle transfer and load retrieval are carried out in an FCFS sequence.

We use the three priority rules for determining the load retrieval sequence (see Section 4.1) in RS and ITT policies, selecting the one with the minimal makespan as the best solution. The makespan improvement of the two-stage heuristic compared to the policy A, A $\in$ {RS, ITT, LWT, FCFS}, is calculated by $\frac{C_A - C}{C_A} \times 100\%$, where C_A and C represent the makespan obtained by the policy A and by the two-stage heuristic, respectively. To identify the efficiency of the two-stage heuristic compared to the aforementioned heuristics (RS, ITT, LWT, and FCFS), three parameters are investigated: the number of requests, the number of shuttles, and the warehouse design. Note that all these experiments are based on the parameters specified in Table 4 with 40 requests and 4 shuttles in the system with 400 locations (10 lanes $\times$ 40 storage positions per lane), if not clearly pointed out otherwise. For each case, 10 instances (different in terms of locations of retrieval requests) are generated. We set the percentage priority α in the LWT policy as 0.7.

Fig. 4(a) and Fig. 4(b) present the computational results of cases with 20, 30, 40, and 50 requests under all five policies, and the improvement in makespan between the two-stage heuristic and the other four policies. The results, shown in Fig. 4, suggest that the two-stage heuristic outperforms the other policies considerably. Specifically, the average makespan of the two-stage heuristic is more than 6.9% shorter than RS, 8.2% than ITT, 4.6% than LWT, and 9.5% than FCFS. In addition, it can be observed that the makespan improvement achieved by the two-stage heuristic becomes larger when there are more requests, which indicates that the two-stage heuristic is well-suited for large-scale problems.

Next, we test the impact of the number of shuttles on the makespan. The number of shuttles is set to $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Fig. 5(a) shows the average makespan and the 95% confidence intervals (only of the two-stage heuristic and FCFS). The following observations can be made. With an increase in the number of shuttles, the average makespan for the two-stage heuristic is decreasing in a concave fashion. Compared to having only one shuttle in the system, using two shuttles can achieve a considerable makespan reduction (about $\frac{6606.3-5152.9}{6606.3} \times 100\% = 22\%$ for the two-stage heuristic). The effect of more shuttles decreases rapidly as the shuttle fleet size increases. For example, with six shuttles in our experiments, the reduction is less than 5% when the number of shuttles increases by 67% to 10 shuttles. Apparently, a small number of shuttles suffices to achieve a near-minimum possible makespan.

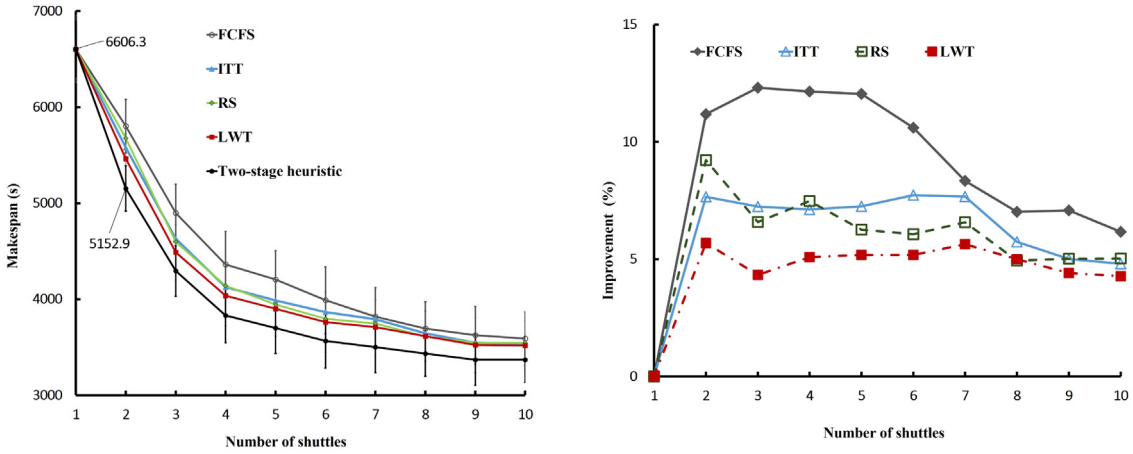
Fig. 5(b) shows that the two-stage heuristic outperforms the other four policies under a varying number of shuttles in the system: more than 7.4% improvement over ITT, 5% improvement over RS, 4.3% improvement over LWT, and 6.2% improvement over FCFS can be expected. When there is only one shuttle in the system, We assume the makespan of all five policies is identical to the optimal makespan provided in Appendix C.

We also validate the performance of the two-stage heuristic under different warehouse designs, assuming every load can be stored in any lane[1] We consider an experiment with 40 retrieval requests and a total of 400 storage positions. We vary the number of lanes as follows: $N_l = \{5, 8, 10, 20\}$. As shown in Fig. 6(a), the makespan increases with the number of lanes for all five policies. This is because a wide-shallow warehouse layout characterized by a large number of storage lanes leads to a long travel time for the forklift. This result suggests that warehouse managers adopt a narrow-deep layout with a small number of deep storage lanes rather than a wide-shallow layout. Moreover, Fig. 6(b) indicates that the proposed two-stage heuristic has an advantage over the other policies, and the improvement can be expected to be more than 5%. We also test a dynamic setting, where retrieval requests arrive block by block in a dynamic fashion. The results can be found in Appendix D.

5.3. Investment cost minimization problem

In addition to makespan, we also study the cost minimization problem under all five policies. The purpose, in this case, is to find the optimal number of shuttles N_s, lanes N_l, and storage positions

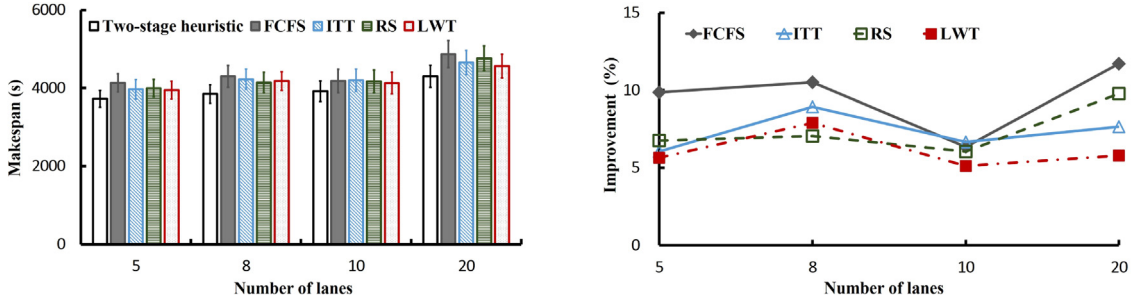
[1] This implicitly assumes only one product is to be stored.



(a) Average makespan and 95% confidence intervals

(b) Average makespan improvement of the two-stage heuristic

Fig. 5. Effect of number of shuttles on makespan (40 requests).



(a) Average makespan and 95% confidence intervals

(b) Average makespan improvement of the two-stage heuristic

Fig. 6. Effect of different warehouse designs on makespan (40 requests).

per lane N_c, such that the total cost (consisting of the cost of shuttles and storage positions) is minimized. This must be done while ensuring that the system makespan for 40 requests is smaller than a threshold specified by the warehouse manager.

Let c_s and c_u be the annual cost of a shuttle and a storage position, respectively. The number of storage positions in the warehouse system N_{total} is given. We have $N_{total} = N_c \times N_l$. The objective is to minimize $c_s N_s + c_u N_{total}$. Note that the makespan function is related to the number of shuttles (N_s), lanes(N_l), and storage positions per lane (N_c). The constraint $Makespan(N_s, N_l, N_c) \leq threshold$ must hold. We use a full enumeration approach to solve the problem.

We consider a deep-lane shuttle-based storage system with $N_{total} = 400$ storage locations. The purchase cost per shuttle is € 20,000, and the cost per storage position is € 60. We select the basic parameters from Table 4 and vary the makespan threshold as $\{1000, 1500, 2000, 2500, 3000, 3500\}$. For each threshold, we generate ten instances with different retrieval request locations and solve each using the five policies. Fig. 7 reports the total minimum annual costs. For a given makespan threshold, the two-stage heuristic approach has a lower total annual cost than ITT, RS, LWT, and FCFS. The reason is that to achieve a given makespan when completing 40 requests, the two-stage heuristic needs fewer shuttles than other policies. With the same total investment cost, the two-stage heuristic results in a lower makespan and, thus, higher throughput performance.

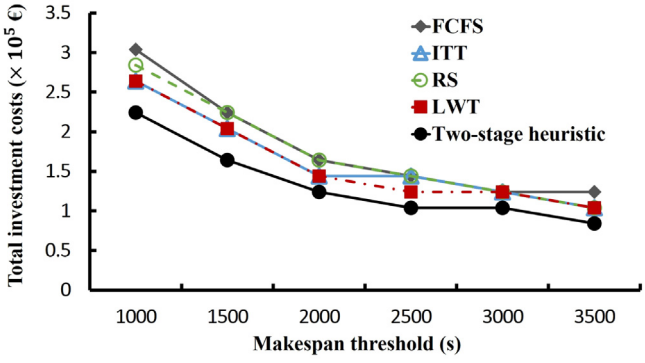
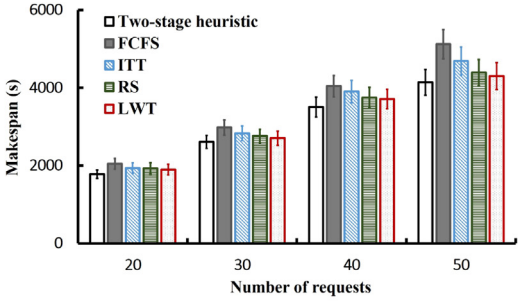


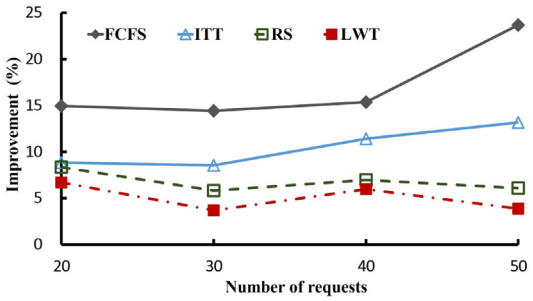
Fig. 7. Total investment costs with given makespan threshold and $N_{total} = 400$.

5.4. Results for deep-lane storage systems with transfer cars

The two-stage heuristic can also be applied in deep-lane storage systems with transfer cars. In such a system, the transfer car is responsible for transferring shuttles with loads between the storage lanes and the I/O point. We use the parameters in Table 4 with 40 requests and 4 shuttles in the system with 400 locations (10 lanes × 40 storage positions per lane). The handling time for the transfer car to pick up or drop off the shuttle or the load with the shuttle



(a) Average makespan and 95% confidence intervals



(b) Average makespan improvement of the two-stage heuristic

Fig. 8. Effect of number of requests on makespan in the system with a transfer car.

is 5 s ($t_s = t_c = 5$ s). Results show that the two-stage heuristic has a more than 8.3% improvement over the ITT policy, a more than 5.8% improvement over the RS policy, a more than 3.7% improvement over the LWT policy, and a more than 14.4% improvement over the FCFS policy. These results are similar to the results for the deep-lane storage system with a forklift.

6. Extensions

This section extends the request scheduling process within a single storage level to multiple levels. Then the impact of shuttles that can autonomously compact the retrieval requests within a lane is investigated.

6.1. Multi-level system with a forklift

A deep-lane storage system usually consists of multiple levels. All levels have the same number of lanes. The difference with a single-level storage system is the additional vertical movement for the forklift. Thus, the travel time of the forklift comprises not only the time to reach the lane with the request (denoted by the horizontal travel time) but also the time to reach the level with the request (denoted by the vertical travel time) in the multi-level system. For safety reasons, forklifts drive and lift sequentially: a horizontal move from the I/O point to the front end of the lane and a vertical movement from the ground to the specific level.

We take the sum of the horizontal travel time and vertical travel time as the travel time of the forklift in the multi-level system with a forklift. Accordingly, the travel time of a forklift from the I/O point to lane l_i of load i in level h_i equals

$$d_i = (l_i - 1) \times D_l / v_f + (h_i - 1) \times D_h / v_h, \qquad i \in R.$$

where D_h is the height of a level, and v_h is the velocity of a forklift in the vertical direction.

To study the impact of the number of levels on the system makespan, we consider the following numerical experiment: Given a total of 960 storage locations and 10 storage positions per lane, we vary the number of levels as $\{8, 6, 4\}$, so that the number of lanes per level is 12, 16, and 24, respectively. The other parameters D_h and v_h are set to 0.4 m and 0.3 m/s. The velocity of a forklift in the horizontal direction is 1.7 m/s (see Section 5.1). Table 6 shows the makespan against the number of levels for 20 requests and 50 requests. We find that with fewer levels, the makespan decreases. This result can be explained by the fact that the smaller the number of levels, the longer the cross-aisle becomes. Since the velocity of a forklift in the vertical direction is much lower than that in the horizontal direction, the vertical move becomes the bottleneck. Therefore, to minimize the makespan, It seems advisable to build a deep-lane system with a small number of levels. However, the fewer the levels, the lower the space utilization. This trade-off between space utilization and makespan will be investigated in future work.

Table 6
Impact of the number of levels on the system makespan (960 locations, 10 positions per lane) .

No. of levels	No. of lanes	Makespan of the two-stage heuristic	
		20 requests	50 requests
8	12	1309.9 s	3414.6 s
6	16	1123.2 s	2881.3 s
4	24	964.6 s	2472.1 s

6.2. Autonomously compacting retrieval requests

In our basic FSS problem, we assume that the shuttle must wait for the handover of the load at the front end of the lane. However, some shuttles can leave the load at the front end of the lane and move on to retrieve the next requested loads in the same lane. This potentially compacts the request retrieval process and reduces the makespan.

In this case, the proposed algorithm still works, as different modes of shuttles only affect the moment when the shuttle becomes available to retrieve load h, a_h, $h \in R$. For a lane with multiple requests, if two successive loads, q and h, are retrieved by the same shuttle and load q is first retrieved, then the equation $a_h = a_q + p_h$ holds. Note that the pick-up position (the front end of the lane) might still be occupied by a previous load, in which case, the load must be dropped off at the position adjacent to the pick-up position. As soon as the forklift has picked up the first load, the shuttle must move the load from the adjacent position to the pick-up position. We neglect any additional time to do this because, in practice, it will always be faster than the time needed by the forklift to return. A similar assumption is made in other papers, i.e., Zaerpour et al. (2015). For the data used in Table 5, the makespan using the two-stage heuristic will reduce by, on average, 5.09% (ranging between 4.54% at minimum and 6.91% at maximum).

7. Conclusions and future research

This paper studies the operational problem of how to transfer shuttles and schedule retrieval requests to minimize the makespan in a deep-lane storage system with a forklift and shuttles, given multiple retrieval requests. We develop an integer programming model for the problem and prove it is NP-hard. We propose a two-stage heuristic algorithm to solve it. In stage 1, a load retrieval

sequence is determined based on three priority rules. The optimal shuttle transfer sequence is obtained using a dynamic programming approach in stage 2. Numerical experiments show that the two-stage heuristic can provide high-quality solutions in a reasonable time. Compared with the methods used in practice and in the literature (ITT, RS, LWT, and FCFS policies), our heuristic can reduce both the makespan and the total cost. We also study the impact of the number of shuttles on the makespan. The makespan of a given set of requests decreases with an increasing number of shuttles in a convex fashion. Increasing the number of shuttles beyond a threshold is not helpful to reduce the makespan.

Our paper makes some assumptions that could be relaxed in future research. First, multiple forklifts or transfer cars can be considered. Incorporating multiple forklifts or transfer cars into the model, as well as possible congestion among these, would further increase the practical applicability. It is also possible to relax the assumption that there is only one type of item in a lane. In a system with multiple items in one lane, reshuffle operations are inevitable when a lane has more than one retrieval request. A third extension would be to consider storage policies other than dedicated storage, such as class-based storage and full-turnover storage. Other future research could examine the shuttle scheduling problem incorporating the use of real-time information in the scheduling problem.

Acknowledgments

The authors are grateful to the Editor and three anonymous referees for valuable comments and constructive suggestions. This research is supported by the National Key R&D Program of China (No. 2018YFB1601401); the National Natural Science Foundation of China (Nos. 71991464/71991460, 72091215/72091210, 72271225, 71921001); Anhui Provincial Natural Science Foundation (No. 2208085J06); and the USTC Research Funds of the Double First-Class Initiative (No. YD2040002017).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.11.037

References

Anken, N., Gagliardi, J. P., Renaud, J., & Ruiz, A. (2011). Space allocation and aisle positioning for an industrial pick-to-belt system. *Journal of the Operational Research Society, 62*(1), 38–49.

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

Boysen, N., Boywitz, D., & Weidinger, F. (2018). Deep-lane storage of time-critical items: One-sided versus two-sided access. *OR Spectrum, 40*(4), 1141–1170.

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

Carlo, H. J., & Vis, I. F. A. (2012). Sequencing dynamic storage systems with multiple lifts and shuttles. *International Journal of Production Economics, 140*(2), 844–853.

De Koster, R. B. M., Le-Duc, T., & Yu, Y. (2008). Optimal storage rack design for a 3-dimensional compact AS/RS. *International Journal of Production Research, 46*(6), 1495–1514.

Deng, L., Chen, L., Zhao, J., & Wang, R. (2021). Modeling and performance analysis of shuttle-based compact storage systems under parallel processing policy. *PloS One, 16*(11), e0259773.

Dong, W., Jin, M., Wang, Y., & Kelle, P. (2021). Retrieval scheduling in crane-based 3D automated retrieval and storage systems with shuttles. *Annals of Operations Research, 193*(1), 111–135.

Gharehgozli, A. H., Yu, Y., Zhang, X., & De Koster, R. B. M. (2017). Polynomial time algorithms to minimize total travel time in a two-depot automated storage/retrieval system. *Transportation Science, 51*(1), 19–33.

Goetschalckx, M., & Ratliff, H. D. (1991). Optimal lane depths for single and multiple products in block stacking storage systems. *IIE Transactions, 23*(3), 245–258.

Han, M.-H., McGinnis, L. F., Shieh, J. S., & White, J. A. (1987). On sequencing retrievals in an automated storage/retrieval system. *IIE Transactions, 19*(1), 56–66.

Hao, J., Yu, Y., & Zhang, L. L. (2015). Optimal design of a 3D compact storage system with the i/o port at the lower mid-point of the storage rack. *International Journal of Production Research, 53*(17), 5153–5173.

Manzini, R., Accorsi, R., Baruffaldi, G., Cennerazzo, T., & Gamberi, M. (2016). Travel time models for deep-lane unit-load autonomous vehicle storage and retrieval system (AVS/RS). *International Journal of Production Research, 54*(14), 4286–4304.

Roodbergen, K. J., & Vis, I. F. A. (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research, 194*(2), 343–362.

Stadtler, H. (1996). An operational planning concept for deep lane storage systems. *Production and Operations Management, 5*(3), 266–282.

Tappia, E., Roy, D., De Koster, R., & Melacini, M. (2017). Modeling, analysis, and design insights for shuttle-based compact storage systems. *Transportation Science, 51*(1), 269–295.

Yang, P., Miao, L., Xue, Z., & Qin, L. (2015). Optimal storage rack design for a multi-deep compact AS/RS considering the acceleration/deceleration of the storage and retrieval machine. *International Journal of Production Research, 53*(3), 929–943.

Yu, Y., & De Koster, R. B. M. (2009a). Designing an optimal turnover-based storage rack for a 3D compact automated storage and retrieval system. *International Journal of Production Research, 47*(6), 1551–1571.

Yu, Y., & De Koster, R. B. M. (2009b). Optimal zone boundaries for two-class-based compact three-dimensional automated storage and retrieval systems. *IIE Transactions, 41*(3), 194–208.

Yu, Y., & De Koster, R. B. M. (2012). Sequencing heuristics for storing and retrieving unit loads in 3D compact automated warehousing systems. *IIE Transactions, 44*(2), 69–87.

Zaerpour, N., Yu, Y., & de Koster, R. B. M. (2015). Storing fresh produce for fast retrieval in an automated compact cross-dock system. *Production and Operations Management, 24*(8), 1266–1284.

Zhao, N., Luo, L., & Lodewijks, G. (2018). Scheduling two lifts on a common rail considering acceleration and deceleration in a shuttle based storage and retrieval system. *Computers & Industrial Engineering, 124*, 48–57.

Zou, B., Koster, R. D., & Xu, X. (2018). Operating policies in robotic compact storage and retrieval systems. *Transportation Science, 52*(4), 788–811.

Zou, B., Xu, X., De Koster, R. B. M., et al., (2016). Modeling parallel movement of lifts and vehicles in tier-captive vehicle-based warehousing systems. *European Journal of Operational Research, 254*(1), 51–67.