Yard Operations Optimisation at Maritime Terminals under Uncertain Truck Arrivals: Container Stacking, Retrieval, and Relocation

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy

by Yuanjun Feng

September 2021

Supervisors: Prof. Dongping Song; Dr. Dong Li

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Prof. Dongping Song and Dr. Dong Li, who guided and supported me during my study at the University of Liverpool. I was honoured to work with them and learn from them. I deeply appreciate Prof. Song's interest in my project when I applied for the PhD position four years ago, and his support and encouragement to my research throughout the four years. I am sincerely grateful for his great availability of guidance, inspiration in discussions, constructive comments and suggestions on the thesis, and contributions and guidance throughout publishing papers. I am sincerely grateful to Dr. Li for providing different views to my research, especially in challenging me to reflect on the theory, and for providing constructive comments and suggestions on the thesis.

I would like to thank Dr. Ying Xie and Prof. Qingcheng Zeng for their contributions to our collaborated papers. I would like to express my gratitude to Prof. Zeng, who was the supervisor of my master study, for taking me to the research field of container terminal operations and having provided much guidance.

I would also like to thank Dr. Cagatay Iris, Prof. Niraj Kumar, Dr. Roula Michaelides, Prof. Ian McHale, and Dr. Hossein Sharifi, who served as the panel members for my annual progress review, for their constructive suggestions on my thesis. I am also thankful to all colleagues in the OSCM group and staff at the ULMS for creating a supportive research environment and my friends in Liverpool for their help and accompany.

My greatest appreciation goes to the support and love of my family. I also appreciate the support and accompany of my boyfriend who has been pursuing his PhD degree during this period in another university.

I dedicate this thesis to my grandparents for bringing me up and supporting my education.

Last but not least, I am thankful to the China Scholarship Council for awarding me the scholarship for this PhD project.

Declaration

The thesis is presented by the style of **thesis structured as papers** following the University of Liverpool guidelines, and it includes **publications**. The contribution made by the student and the co-authors are described in three Authorship Declaration Forms which have been submitted as independent documents along with the thesis.

Contents

Yard Operations Optimisation at Maritime Terminals under Uncertain Truck Arrivals: Container Stacking, Retrieval, and Relocation

by
Yuanjun Feng

Submitted to the University of Liverpool on Sep 13, 2021,
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

"On April 26, 1956, a crane lifted fifty-eight aluminum truck bodies aboard an aging tanker ship moored in Newark, New Jersey. Five days later, the Ideal- X sailed into Houston, where fifty-eight trucks waited to take on the metal boxes and haul them to their destinations. Such was the beginning of a revolution." (Levinson, 2016)

The Box: how the shipping container made the world smaller and the world economy bigger.
Marc Levinson (2016). Princeton University Press, New Jersey.

Abstract

Ocean container shipping fulfils an essential role in today's global supply chains. Maritime container terminals, which are specialised seaports to handle containers for them to be transferred between various transport modes, are essential transport facilities in the container shipping network, and their operation efficiencies are crucial to enable efficient container shipping. The yard operation system of a container terminal serves as the central point that connects the seaside and landside operations, which plays an important role in a terminal's overall performance. High yard storage density and port congestion have been an industry-wide challenge. The management of yard operations has become increasingly important to the efficiencies and competitiveness of container terminals.

A major source of inefficiency in yard operations is container relocation (or reshuffling) – moving the blocking containers out of the way to access the target container. Being non-productive, container relocation is not only costly to terminal operators but also results in delays in container retrieval and low quality of customer service. Reducing the number of relocations has therefore attracted increasing attention from the operations research community. This thesis addresses two key types of container management problems at the yard blocks: the Storage Location Assignment Problem (SLAP) – concerning where to store an incoming container in the container stacking process and the Container Relocation Problem (CRP) – concerning where to relocate a blocking container in the container retrieval process, with a focus on import containers, aiming to improve container yard operation performance. One of the central elements in yard operations is the availability and accuracy of container retrieval times. The retrieval times of import containers are usually subject to great uncertainties due to the uncertain arrival times of external trucks, but studies in this respect remain understudied. To tackle the challenges of uncertainties in yard operations planning, this thesis proposes novel strategies for import container stacking and retrieving, by utilising truck arrival information and customer information and incorporating service flexibilities. Correspondingly, new mathematical models are formulated, and exact and heuristic algorithms are developed for solving the arising optimisation problems.

Specifically, three different but closely related optimisation problems are addressed and presented in the format of three papers, with the first two papers on the Stochastic CRP (SCRP) and the third paper on the SLAP. In the **first paper**, a flexible service policy is applied to the container retrieval process, where the trucks arriving at the same sub-time window are allowed to be served out-of-order. A new methodological framework is constructed that jointly optimises the container retrieval sequence and the relocation positions, where stochastic dynamic programming is used to formulate the SCRP, tree search-based exact algorithms and heuristic algorithms are designed as solution methods, and a discrete-event simulation model is developed to evaluate the solutions under uncertainties. This flexible policy is shown to be effective to reduce both the number of relocations and average truck waiting times in a wide range of scenarios. Note that flexible policies may raise the issue of service fairness to customers because of the out-of-order service phenomenon. The **second paper** generalises the problem in the first paper to the case with multiple sub-time windows and incorporates service fairness into the optimisation objective. The new problem is formulated by a stochastic dynamic programming model with two lexicographically ordered objectives and solved via a hierarchical iterative approach. The results show interesting trade-offs between relocation efficiency and service fairness. It is also found that the information of trucks' arrival preference over a time window can be valuable in reducing the number of relocations in some circumstances. Finally, the **third paper** proposes a smart stacking strategy for import container stacking by utilising the customer identity information (the consignees of containers), which intelligently classifies containers into smart containers

(free from relocation) and non-smart containers (may require relocation) and allocates them to smart stacks and non-smart stacks respectively. Two variants of SLAPs are investigated under two forms of smart stacking policies, the split policy and the non-split policy, depending on whether the containers from the same customer are allowed to be split between smart stacks and non-smart stacks. Mixed-integer programming is used to formulate the SLAPs, and a reformulation with enhanced computational performance is developed by leveraging the structural properties of the optimal solution. A divide-and-conquer heuristic algorithm based on the structure of the model is designed to seek higher solution efficiencies. The results show that the smart stacking strategy can significantly reduce the total container retrieval time compared with the traditional practice, and the split policy is superior to the non-split policy. Some beneficial scenarios regarding customer information quality and yard utilisation rate are also identified.

The findings of the thesis can help terminal operators to gain insights into the benefit of the proposed novel strategies and the value of customer information. Such an understanding could promote industry-wide collaboration between terminal operators, trucking companies and cargo owners, and revolutionise the current import container handling practice towards a more efficient and sustainable fashion. The developed modelling and solution approaches advance the knowledge on operations research methods for yard operations problems and can facilitate decision support for terminal operators to implement the proposed strategies.

Chapter 1

Introduction

Containerisation – the stowage of freight in sealed, reusable metal boxes of standardized dimensions – is one of the most important cargo-moving techniques developed in the 20th century. Since its introduction in the 1960s, containerisation has revolutionised not only the shipping industry and ports, it has also fundamentally changed the world's whole international trade and transport systems (Stahlbock and Voß, 2008a). Being highly efficient compared to other shipping sectors, this concept has been accepted worldwide with the development of supporting transport facilities and handling equipment (Jiang et al., 2015). Container terminals, which are specialised seaports to handle containers for them to be transferred between various transport modes, are essential transport facilities in the container shipping network, and their operation efficiencies are crucial to enable efficient container shipping. This thesis aims to enhance the operational efficiencies of container terminals by approaching a series of optimisation problems on import container handlings in container yards under uncertainties.

This Chapter provides an overview of the operations management problems in maritime container terminals and outlines the work executed in this thesis. Section 1.1 introduces the role of container terminals in global container shipping. Section 1.2 describes the logistics systems in container terminals and provides a general introduction of the operations management problems arising at container terminals. Section 1.3 specifically introduces the optimisation problems on the container handling operations in yard blocks and describes the uncertainties in container handling. Section 1.4 states the research scope and explains the research motivation. Section 1.5 and Section 1.6 respectively provide a compendious literature review for the two types of optimisation problems dealt with in this thesis, to prepare the background for the main research chapters and justify the research gap and methodology applied. Section 1.7 explains the impact of new storage systems to the thesis. Section 1.8 presents the thesis structure.

1.1 Container terminals in global container shipping

With the globalisation of the supply chain, ocean transport has become an essential component of international trade, and ocean container shipping fulfils an essential role in today's global supply chains. (Fransoo and Lee, 2013). According to the research conducted by Lloyd's Marine Intelligence Unit in 2009, 75% of the world trade by volume or 60% by value was carried by sea; and within the sea transport industry (including container, tanker, dry bulk, and general cargo), 52% of cargoes by value were containerised cargoes (Lee and Song, 2017). The importance of container shipping is also supported by the fact that container shipping is the fastest growing sector in shipping in the last two decades and it is regarded as the world's truly global industry due to its ability to achieve integrated end-to-end supply

chains (Song, 2021a). Since its inception in the 1960s, containerisation has experienced modest growth in the first three decades and then has expanded at a rapid rate in the last two decades (Lee and Song, 2017). Over the past two decades, the global containerised trade has experienced a 5.85% average annual growth (UNCTAD, 2019). Although the global financial crisis halted its growth in 2009, a recovery was witnessed in 2010 and onwards, bringing the total container trade to 152 million Twenty-Foot Equivalent Units (TEUs) in 2019 (UNCTAD, 2020).

Maritime container ports/terminals are essential facilities to enable container shipping. As an indispensable part of the global container shipping network, maritime container terminals act as an interface connecting seaborn transport and inland transport, where containers are transferred between different transport modes (container ships, barges, trucks, rails). The total throughput of container terminals has experienced significant growth during the last few decades. In 2019, 811 million TEUs of containers were handled in ports worldwide, with an average annual growth rate of 4.05% from 2015 to 2019 (UNCTADstat). There are more than 1000 ports in 200 countries that can handle container ships (Song, 2021a). Leading container ports such as Shanghai handle more than 40 million TEUs annually (see Table 1.1 in the appendix at the end of this chapter) and this volume is projected to continue to rise (THE MARITIME EXECUTIVE, 2021). Affected by the COVID-19 pandemic, although the world port container throughput is expected to contract by 7.3% in 2020, a jump up to more than 10% is projected in 2021, according to Drewry baseline forecast (UNCTAD, 2020). It has been reported that Los Angeles-Long Beach is preparing to handle a record 20 million TEUs this year (Mongelluzzo, 2021), an increase of about 17.9% over 2019. The increasing volume calls for more efficient container handling at container terminals. In addition, other factors and trends of container shipping, including mega-vessels, slow steaming, hub-and-spoke network, intense port competition, etc., bring challenges for container terminals to improve their competitiveness and necessitate more efficient operations planning (Jiang et al., 2015; Kim and Lee, 2015).

Port congestion has stretched around the globe nowadays, which puts pressure on terminal operators to ensure efficient global container shipping. It is reported that through to May this year, containerships spent more than double the time in waiting on anchors for berths since 2019. In later July this year, 116 ports around the world reported congestion challenges and 328 ships were idling in front of ports waiting for berth spaces (Chambers, 2021). Some container ports, for example, major UK container ports (e.g., Felixstowe and Southampton) (Todd and Waters, 2020) and US west coastal ports (e.g., Los Angeles and Long Beach) (Porter, 2020) have experienced severe congestion in the second half of 2020. Container terminals need to find solutions for reversing the situation of congestion.

Facing the challenges of increasing volumes and port congestion, it is of increasing importance for many ports to seek more efficient and sustainable operations. In this thesis, we propose novel operation strategies for container retrieving and stacking at container terminal yards, which can help to improve yard operation efficiency and relieve the situation of port congestion.

1.2 Container terminal operations and decision problems

Container terminals are complex logistics systems that involve many interdependent operations management problems across three sub-systems: seaside, yardside and landside operations. Section 1.1.2 presents an overview of container terminals. Section 1.2.2 introduces the operational management problems in container terminals from three planning horizon levels and three logistics sub-systems.

1.2.1 Overview of container terminals

The layout of a container terminal can be divided into different areas physically according to the functional purpose of each area. Though the division can be different in terms of the level of details by different researchers (e.g., Carlo et al., 2014 a, b; Meisel, 2009), usually, a container terminal could be divided into six specific functional areas: berth, quay, transport area near the quay, yard, transport area near the gate, and gate (see Fig. 1.1). According to the logistics process of handling a container in a terminal, a terminal may be categorised into three logistics sub-systems, namely the seaside, yardside and landside. The berth, quay and transport area near the quay are considered *seaside*, while the transport area near the gate and the gate are considered *landside*. A few researchers consider that the landside also includes the yard area (e.g., Carlo et al., 2014 a, b), while others discuss the yard area or *yardside* separately as the buffer area between seaside and landside (e.g., Song, 2021a,b; Caserta et al., 2020; Zhen et al., 2013).

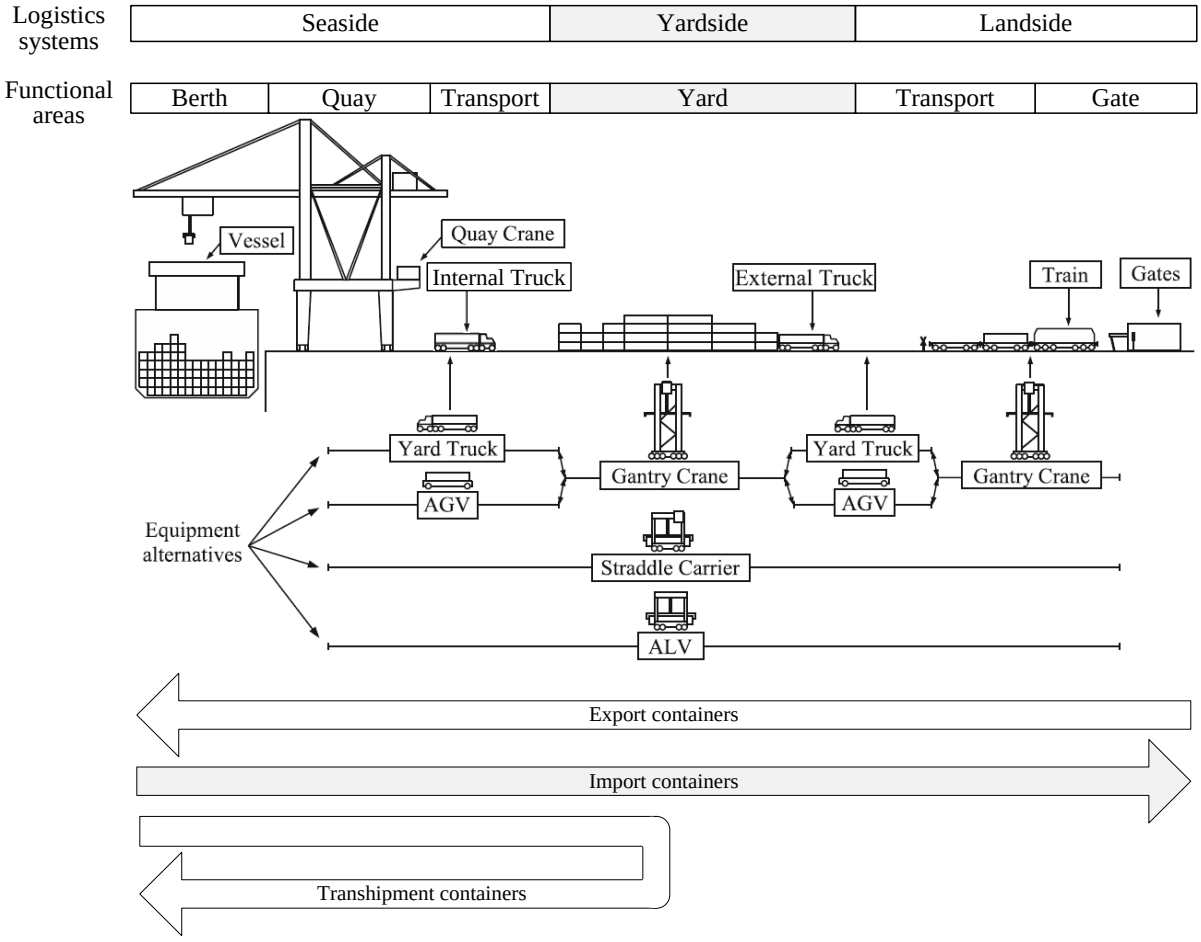


Fig. 1.1 Schematic representation of a typical container terminal (adapted from Meisel (2009) and inspired by Galle (2018))

The *berth* area is the water place near the shore that connects vessels and the terminal. Vessels moor at the berths when they arrive at the terminal from the sea and wait here while being served. The *quay* area is the place where containers are (un) loaded onto (from) vessels by using Quay Cranes (QCs). The *yard* area is the storage area where containers dwell after arriving at the terminal and wait for onwards transport. It serves as a decoupling point between seaside and landside transport. Due to the asynchronism between

containers' arrival times and departure times, all containers (except direct transhipment containers, see Zeng et al., 2017) need to dwell at the yard for a period of time, typically ranging from a day to a couple of weeks (Park et al., 2011), before being loaded to another transport mode. Various equipment can be used to handle containers in the yard, which mainly include the gantry Yard Crane (YC) and the Straddle Carrier (SC). YCs are used for stacking containers and moving containers to the transfer points where the containers are exchanged between the YC and the transfer vehicles. There are mainly two types of YCs, Rail-Mounted Gantry Cranes (RMGCs) and Rubber-Tyred Gantry Cranes (RTGCs). In a survey of 114 terminals all over the world, RTGCs were found to be the most popular yard equipment used in 63.2% of the terminals studied (Wiese et al., 2011). The second most used equipment is the SCs (20.2%), which are capable of self-lifting and stacking containers. The layout and the technicalities of the yard area will be further introduced in detail in Section 1.3. The *transport area near the quay* is where containers are transported from (to) vessels to (from) the storage yard. Common transfer vehicles used for performing this transport include trailers, SCs, Automated Lifting Vehicles (ALVs), Automated Guided Vehicles (AGVs), and reach stackers, all of which are categorised as internal trucks. Last, the *gate* is the interface between the terminal and hinterland where containers come into or go out of the terminal via External Trucks (ETs). If YCs are used for yard operations, ETs are sent directly to the dedicated yard blocks to deliver or pick up containers. If SCs are used, ETs enter the parking area near the yard waiting to be served. It is also possible that ETs are capable of self-service if containers are stored on chassis in the yard. Other inland transport vehicles that link the terminal with hinterland transport include trains and barges. Note that barge can be regarded as an inland transport vehicle but barge operations are performed at the quayside. For the terminal that can serve trains, railway tracks lead into the terminal. If the railway yard is operated by YCs, containers are loaded (unloaded) to (from) the trains by YCs, where horizontal transport of containers is required to transfer the containers between the terminal yard and the railway yard. Otherwise, SCs or ALVs are used to serve the trains. The *transport area near the gate* is the place where containers are transferred between the terminal yard and the ETs at the parking area or the trains at the railway yard.

There are three types of containers that are handled in a container terminal based on the directions of the container flow: import containers, export containers, and transhipment containers. *Import* containers arrive from the seaside by vessels and then are discharged to the yard for temporary storage, and finally, they leave the terminal via hinterland transport. This operation flow is vice versa for *export* containers. For *transhipment* containers, after being discharged from the vessels and stored in the yard, they will be loaded onto other vessels for their next journey.

1.2.2 Operations management problems in terminal operations

The complex container terminal operation systems give rise to many operations management problems to deal with. The planning of a problem is usually made under a specific length of the planning horizon that depends on the characteristics of the problem. According to the planning horizon levels, these planning problems can be classified into strategic, tactical, and operational planning problems. While the performance of a container terminal can be improved at the strategic level by adopting such as new technologies or terminal layout re-design, at the operational level, a proven means to enhance the efficiency of terminal operations is optimising how the operations are carried out (Caserta et al., 2020). Researchers have devoted a great deal of effort to the definition of the optimisation problems of terminal operations and the development of Operations Research (OR) methods to tackle these problems. The optimisation of these operations problems can be differentiated between tactical level and operational level depending on the planning horizon length. A number of general review papers on terminal operations have been published

(Vis and de Koster, 2003; Steenken et al., 2004; Stahlbock and Voß, 2008b; Angeloudis and Bell, 2011; Kim and Lee, 2015). Meanwhile, some concentrate on the review of seaside operations (Bierwirth and Meisel, 2010, 2015; Carlo et al., 2015), transport operations (Carlo et al., 2014b), yard operations (Zhen et al., 2013; Carlo et al., 2014a), and new technologies and OR models for terminal operations (Gharehgozliet al., 2016). In the following, we provide a brief introduction to the problems at each planning level. For a detailed description, we refer the readers to Iris (2016).

Strategic problems are often related to long-term investment decisions with years of lifespan that are very costly to change. Typical problems include port competition, terminal layout design, multi-modal interface design, selection of equipment types, and selection of terminal operating systems.

Tactical problems usually deal with plannings that are made several months or weeks in advance. These decisions are mostly related to the long-term deployment and assignment of the terminal resources that will not be changed easily once determined. Typical examples include berth template design (assigning berthing windows and positions for cyclic calling vessels), yard template design (assigning yard space for cyclic calling vessels), storage yard division (dedicating storage space for import containers, export containers, empty containers, and reefer containers), vehicle fleet size (determining the number of transport equipment) etc.

Operational problems correspond to the planning of terminal operations for the efficient utilisation of key resources during a short-term that varies between days to seconds, which are closely related to the key performance indicators of a container terminal. In most container terminals, key resources include berths, storage spaces, QCs, and YCs because of the high cost of increasing their capacity. As pointed out by many researchers (e.g., Zhang et al., 2003; Stahlbock and Voß, 2008b; Knatz, 2017; de Melo da Silva et al., 2018), key port performance indicators include: (1) the vessel berthing time, which is a measure of the service to shipping lines, (2) the throughput of the quay cranes, which is a measure of the productivity of a terminal, (3) the turnaround time of trains and external trucks, which is a measure of the customer service level to hinterland transport carriers, and the (4) container dwell time at the yard, which is a measure of the yard throughput capacity.

The thesis is concerned with operational level problems. In the following, we introduce the main operational level planning problems in relation to three sub-systems: quayside, yardside, and landside. Some problems cover multiple logistics sub-systems.

Quayside

Operational planning at quayside focuses on berth planning, stowage planning, QC work scheduling, container loading/unloading sequencing, and vehicle dispatching and routing.

Berth planning is addressed in the berth allocation problem (BAP) that determines the berthing times and positions of a set of vessels within a planning horizon. The goal is to maximise the service levels of vessels given spatial (discrete or continuous berths), temporal (static or dynamic), and handling time (dependent on berthing position and/or QC assignment and schedules or fixed) constraints. Because vessels' arrival times may deviate from their estimated arrival times, recent studies focus on designing robust and reliable berth planning by considering such uncertainty (e.g., Xiang et al., 2017; Iris and Lam, 2019).

The Stowage Planning Problem (SPP) is concerned with determining the attributes of containers (such as weight, size, port of destination, and type of containers) to be loaded into slots in a vessel or determining the exact slot of each container in the vessel, by considering various vessel stability and strength measures and physical properties of the vessel. The goal is to minimise the stay times of vessels at the terminal, maximise QC utilisation, or minimise the costs related to the yard and transport operations (see e.g., Monaco et al., 2014).

The QC work scheduling involves the QC Assignment Problem (QCAP) and the QC Scheduling Problem (QCSP). The QCAP determines the number of QCs and/or the specific QCs to be assigned to each vessel, and/or the movements of QCs between vessels, aiming at maximising the QC productivity. This problem is often inter-related and jointly addressed with the BAP because the QC deployment affects the berthing during of a vessel (see e.g., Giallombardo et al., 2010; Iris et al., 2015, 2017). Given a QC assignment, the QCSP determines the schedules of QCs to perform a set of tasks, including the start and end times of a QC in each position of the vessel and the order of performing these tasks. The popular goal is to minimise the makespan of the operation by satisfying constraints such as the precedence relations among tasks and safety distance between cranes. Integrating the BAP, the QCAP and the QCSP is prevailing in studies such as Chen et al. (2012) and Agra and Oliveira (2018). After constructing the QC schedule, the container loading/unloading problem determines the sequence of (un)loading containers.

The Vehicle Dispatching and Routing Problem (VDRP) is tackled at the interface of transferring containers between the quay and the storage yard, aiming at minimising the container (un)loading time, the idle times of QCs and vehicles, etc. The vehicle dispatching problem determines the assignment of vehicles to specific (groups of) containers. The vehicle routing problem is mostly considered jointly with the vehicle dispatching problem to determine the travel paths and the schedule of each vehicle to pick up and drop a container.

Landside

Operational planning at landside addresses the issues related to receiving and delivering containers at the interface of the yardside operation and the hinterland operations. In the case where trucks are used as hinterland vehicles, gate operations planning problems are addressed to reduce gate congestions, such as the truck appointment system design (e.g., Torkjazi et al., 2018) and the truck arrival management (e.g., Chen et al., 2013a, b). When there is a rail system at the interface, problems related to transferring containers between the yard and the train are addressed, including train loading and unloading, wagon shunting between trains, internal transportation problems between the yard and the rail terminal, and yard crane operations at the rail terminal (see the references in Song (2021a)).

Yardside

Operational planning at yardside can be divided into yard crane operations planning and container handling operations planning.

Yard crane operations planning involves the Yard Crane Deployment Problem (YCDP) and the Yard Crane Schedule Problem (YCSP) (Zhen et al., 2013). The YCDP deals with the determination of the number of YCs to be assigned in each yard block and how to move the YCs among the blocks (for RTGCs). The YCSP is concerned with sequencing storage and/or retrieval operations for each YC considering constraints such as the safety distance between cranes, the job precedence, and the job due time, etc, to minimise the makespan, the total crane travel time or distance, or the total job waiting time. Depending on the number and types of YCs deployed in the block, the YCSPs can be classified into the scheduling of single crane (Ng and Mak, 2005), twin cranes (Hu et al., 2016), triple cranes (Dorndorf and Schneider, 2010) and crossover cranes (Vis and Carlo, 2010). In a recent work by Speer and Fischer (2017), the performances of these four types of yard crane systems are examined and compared.

Container handling operations in the yard involves container stacking during the storage, marshalling and retrieval processes. Corresponding optimisation problems will be discussed specifically in Section 1.3.

1.3 Container handling operations in yard blocks

The yard operation system of a container terminal plays an important role in a terminal's overall performance because it serves as the central point of synchronising the asynchronous container flows between the transport by deep-sea vessels and the hinterland (Covic, 2018). Yard operation is an integral part of the whole process of terminal logistics and needs highly coordinated planning and operations with the seaside and the landside. A yard is composed of multiple blocks, and container handling operations are conducted within yard blocks. The past two decades have seen increasing interests in the field of container handling operations in yard blocks. One of the recent research streams focuses on coping with the uncertainties in the operations. In the following, we first introduce the technicalities of yard blocks in Section 1.3.1, then describe the optimisation problems related to container handling in yard blocks in Section 1.3.2, followed by the presentation of uncertainties in Section 1.3.3.

1.3.1 Technicalities of yard blocks

A typical yard consists of multiple rectangular blocks. Due to the limited land space, containers are usually stacked in multi-tiers in a block. As presented in Fig. 1.2, a typical block can be described in three dimensions: bay, row, and tier. A row and a bay jointly characterise a stack, which is a vertical column for storing containers. Note that in the literature, a stack is also referred to as a column, and both blocks and bays are sometimes referred to as stacks. Each stack consists of several tiers. A row, bay, and tier jointly characterise a slot, which is the basic unit of the block for storing one container (TEU). The number of rows and tiers in a block is limited by the type of yard cranes used, and the optimal design of the block size depends on specific performance measures in the terminal (see e.g., Lee and Kim, 2010). Typically, these values range from 40 to 60 for bays, 6 to 13 for rows, and 3 to 6 for tiers (Galle et al., 2018b).

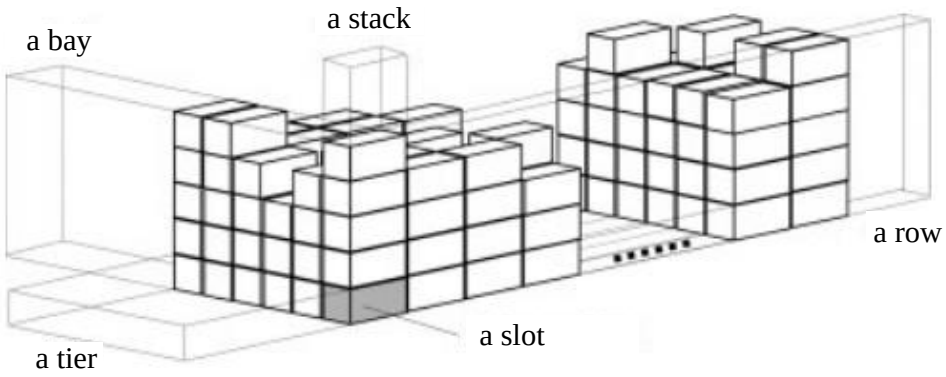


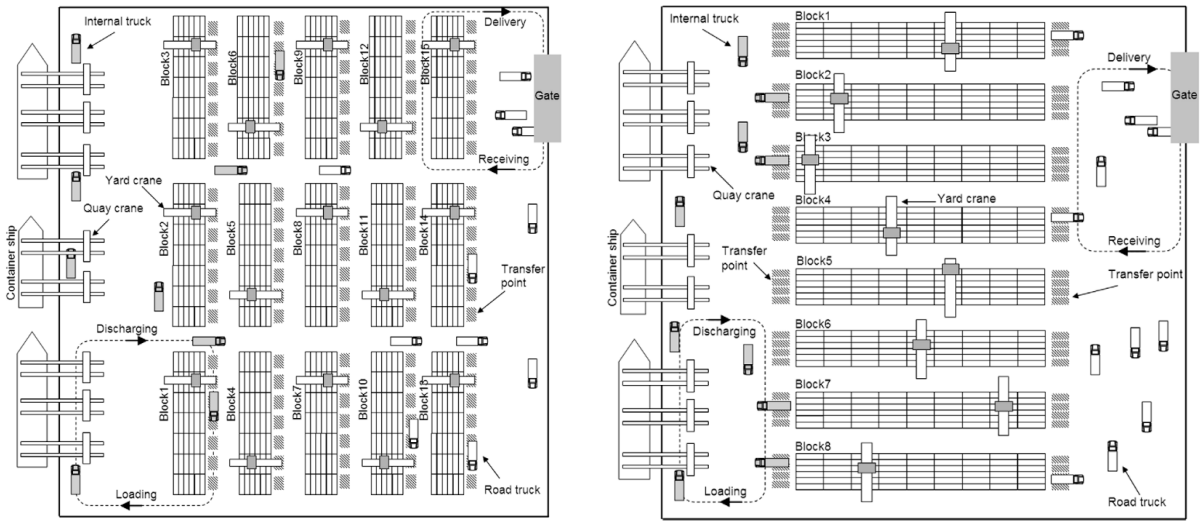
Fig. 1.2 Schematic representation of a typical yard block at a container terminal (adapted from Wan et al., 2009)

There are two main configurations of yard layouts: a parallel layout and a perpendicular layout. They differ in the orientation of the blocks with respect to the quay and the location of the transfer points (TPs) where containers enter and leave the yard.

The parallel layout has blocks positioned parallel to the quay (see Fig. 1.3a). In such a layout, the TPs are located at either both sides or one side of the block, which form truck lanes. When trucks (internal and external) arrive at the block, they drive to the bay associated with its storage or retrieval request and wait for their turns to be served. In this configuration, the YC travels to the bay where the truck parks to pick up (drop) containers from (to) the trucks. This type of yard layout is more common in transhipment terminals

(Gharehgozli et al., 2020) and is the most popular in large Asian terminals. It is referred to as Asian layout by some researchers (Carlo et al., 2014a).

For the perpendicular layout, the yard block is positioned perpendicular to the quay and the TPs are located at both ends of the block (see Fig. 1.3b). The TPs on the seaside are for internal vehicles and those on the landside are for external trucks. For example, at London Gateway, there are six TPs in front of the yard block at the landside. In this configuration, the traffic of external trucks is separated from that of internal vehicles, which is more popular in Automated Container Terminals (ACTs). Automated stacking cranes (ASCs), which are automated RMGCs are normally used to stack containers in this type of layout. When a truck with a storage request arrives at a TP, a YC travels to that TP, picks up the container and then travels back with the container to the storage location that is assigned to the container. When serving a truck with a retrieval request, the YC travels from the location where the requested container is stored to the TP that the truck parks, and then it drops the container at the truck. Compared to the parallel layout, the perpendicular layout reduces the distances travelled by internal and external vehicles at the expense of longer travel distances for YCs. This type of yard layout is more common in export and import terminals (Gharehgozli et al., 2020). As it was first implemented in some large European container terminals, it is also referred to as European layout by some researchers (Carlo et al., 2014a).



(a) Parallel (b) Perpendicular

Fig. 1.3 Two typical types of block layouts in container terminals (Lee and Kim, 2010)

1.3.2 Optimisation problems in container handling in yard blocks

The structure of the multi-tier stacking of containers in yard blocks stipulates that only the topmost container can be accessed directly by a YC. If the target container (i.e., the container with the highest retrieval priority) to be retrieved is not on the topmost tier, those above it – that is, the blocking containers - need to be moved out of the way in order to access the target one. The moves of blocking containers are called relocation, reshuffling, or rehandling (see Fig. 1.4). For consistency, we use the term “relocation” to represent such moves in the rest of the thesis. Container relocation is an unproductive operation that is costly to terminals and results in delays to container deliveries, which has been perceived as the major source of inefficiency in yard operations (Ku and Arthanari, 2016a). The discussion on container relocation is initiated by Kim (1997) who propose a method for estimating the expected number of relocations. Since this starting point, researchers have aimed to reduce relocations through better planning the container

stacking positions in yard blocks. Such motivation has shaped the framework of container handling in yard blocks, leading to a set of problem definitions and solution methods. In a recent systematic literature review on this theme conducted by Covic (2018), 61 studies between 1997 to 2018 are identified and five major problems are defined. All these problem types in some way focus on minimising the number of relocations at retrieval, but the means of how this is achieved are different for each problem type. These problem types are described below.

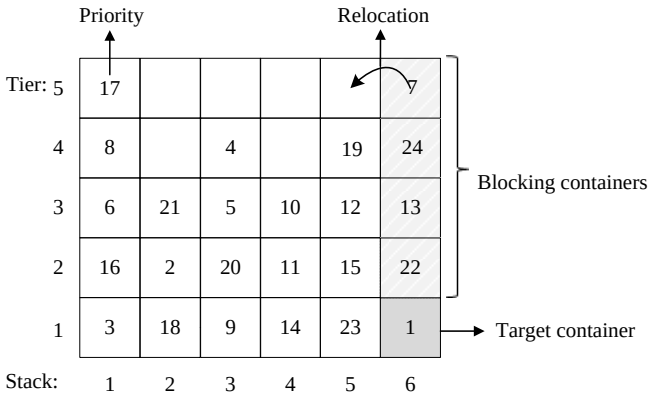


Fig. 1.4 Illustration of the terms regarding relocation

The stacking positions of containers directly affect the number of relocations. The determination of the initial storage of containers into the yard is usually addressed hierarchically in the following two problems: the Storage Space Allocation Problem (SSAP) and the Storage Location Assignment Problem (SLAP).

Storage Space Allocation Problem (SSAP)

The SSAP deals with the container allocation to the yard area at a macro-level, which is the primary problem that needs to be solved before a container reaches the yard. It determines the amount of yard storage space allocated to each vessel for their containers, which can be addressed at various levels according to the storage space unit considered: yard section, yard block, yard sub-block, and yard bay. The main goal of the SSAP is to improve the efficiency and productivity at the seaside interface (e.g., Zhang et al., 2003; Jiang et al., 2012, 2013; Jin et al., 2016), while a few others focus on the performance at both the seaside interface and the landside interface (e.g., Yu et al., 2021).

Storage Location Assignment Problem (SLAP)

The SLAP deals with the container assignment to the yard at a micro-level. It determines the exact storage locations (i.e., slots) in a yard for individual containers. Some of the SLAPs aim at improving the efficiency of the future retrieval process (e.g., Kim et al., 2000; Saur íand Mart ń, 2011; Zhu et al., 2020), while some others focus on the efficiency of the stacking process (e.g., Luo et al., 2016) or the efficiency of both the stacking process and the future retrieval process (e.g., Razouk et al., 2016; Rekik and Elkosantini, 2019; Wang et al., 2020).

The following three problem types arise after the stacking area has already been filled with containers, which are referred to as post-stacking problems by Caserta et al. (2011a), including the Container Relocation Problem (CRP), the Container Remarshalling Problem (CRMP), and the Container Premarshalling Problem (CPMP).

Container Relocation Problem (CRP)

The CRP is addressed in the container retrieval stage. Each container has a *retrieval priority* determining when the container will depart from the yard block (see Fig. 1.4). *Blocking* occurs when the *target container* to be retrieved is stored beneath containers of lower retrieval priorities. The containers on top of the target container are *blocking containers*. When blocking occurs, the blocking containers must be repositioned to other stacks. Each such reposition results in one *relocation*. If the stacking positions of these *relocated containers* are not well planned, they may require further relocation in the future. The standard CRP is, given a pre-specified retrieval sequence for containers stacked in a given bay layout, to retrieve all containers from the bay with the minimum number of relocations (Ku and Arthanari, 2016a). The CRP seems to have received the prime focus within the container handling problems (Covic, 2018). There are several variants of the CRP, which will be introduced in Section 1.5.

Container relocations also take place in *marshalling* operations, which are usually performed during the idle time of YCs to rearrange the containers stacking configuration such that the future retrieval operations will be carried out with maximum efficiency. In marshalling operations, the number of containers remains constant as the operations only involve relocation activities but no retrieval operations. Typically, two types of marshalling operations are distinguished, i.e., *premarshalling* and *remarshalling*.

Container Premarshalling Problem (CPMP)

The CPMP is concerned with finding a minimum length sequence of relocation moves that reorganizes the containers within a bay in such a way that, for a known or estimated retrieval sequence, no further relocations are required during the subsequent retrieval operations. Since containers are relocated only within the same bay, this type of operation is also called intra-bay re-marshalling.

Container Remarshalling Problem (CRMP)

The CRMP arises when marshalling operations are allowed to be performed across bays. It is concerned with finding the shortest length sequence of relocation moves that rearrange a set of containers stored in source bays into target bays within the same block in such a way that no further relocations will be required during the future retrieval process. Since containers are relocated within the same block, this type of operation is also called intra-block re-marshalling. In the CRMP, the crane moving distance for retrieving containers during the future retrieval process should be taken into account as it may substantially contribute to the waiting times of vehicles at the transfer points (Covic, 2017). When more than one YCs are used to handle containers in a yard block, the problem also considers avoiding interference among YCs (Choe et al., 2011).

In this thesis, we address the Storage Location Assignment Problem (SLAP) and the Container Relocation Problem (CRP). They will be discussed in further detail in Sections 1.5 and 1.6.

1.3.3 Uncertainties in container retrieval time

Uncertainties widely exist in terminal operations, which can be sourced from many factors, including the fluctuation of demand, vessels' arrival time and operation time, container dwell times, interfaces between different equipment units, disruptive events like natural disasters, etc (see e.g., Zhen, 2014, 2015; Lu and Xi, 2010; Ku et al., 2012; Iris and Lam, 2019; Tong et al., 2015; Angeloudis and Bell, 2010; Pant et al., 2014). Uncertainty affects every container terminal, which, unless appropriately handled, may degrade the operational gains that stem from optimisation (Angeloudis and Bell, 2010).

For the yard operation system, one of the central elements in container handling in yard blocks is the

availability of container retrieval times (Covic, 2018). This is particularly important in order to avoid relocation activities as relocations are required if containers with higher retrieval priorities are stacked below containers with lower priorities. Usually, the information on container retrieval priorities is not directly given, but can only be predicted from the other characteristics of containers. The availability and accuracy of such data greatly depend on the flow direction of the containers (Kemme, 2020).

Export containers arrive at terminals individually through hinterland transport and departure in large volumes through vessels. The arrival times of export containers are uncertain, but there is usually a cut-off time by which an export container must be checked in at the terminal for loading to a scheduled vessel or train. The cut-off time is usually 24 - 48 hours before the estimated time of departure of the vessel but will vary from the carrier and the terminal. The departure of export containers is usually more predictable as it is connected with a scheduled vessel. To some degree, the loading sequence of export containers is determined by the stowage plan of the vessel (Forster and Bortfeldt, 2012). A stowage plan stipulates the attributes of the containers to be loaded into each slot of the vessel. The attributes include the size (20-foot, 40-foot, 45-foot, high-cube, oversized), the type (reefer, open-top), the weight class (light, medium, heavy), the cargo (dangerous, perishable), and the destination port (Monaco et al., 2014). Containers are usually categorised into groups by these attributes. A vessel slot can be loaded with any container of the group that is specified by the slot. As such, the departure priority of export containers from the storage yard can be derived from the stowage plan. For example, to maintain vessel stability, heavier containers are stowed in lower slots in a vessel and thus should be loaded earlier than lighter containers. Also, to avoid the relocation costs during the vessel unloading process, containers with closer destination ports are placed above those for more distant ports on a vessel. With the stowage plan at hand, terminal operators can plan efficient storage positions for containers in the yard such that containers to be loaded later are not placed above those to be loaded earlier to avoid relocation. However, the information for attributing a group is often not available or can be inaccurate (Forster and Bortfeldt, 2012), or the data (e.g., the port of destination) is subsequently changed due to changing plans of shipping lines (Caserta et al., 2011a). For example, when dedicated weighing procedures are not in place, the weight information of a container is oftentimes not accurate (Heilig et al., 2020). The International Maritime Organization (IMO) Safety of Life at Sea (SOLAS) has amended new regulations that require a mandatory Verified Gross Mass (VGW) of laden containers in order to be loaded onto a vessel from the 1st July 2016. In the meantime, many ports have started to provide weighing services, and some ports charge for the container (e.g., £20.00 in the Port of Felixstowe) if the VGW provided by the shipper has a large error. These regulations and practices may help to improve the quality of the weight data. In addition, the arrival time of vessels also affects the loading sequence of export containers. The actual arrival time of a vessel may vary from the planned time due to the uncertain sailing time and turnaround time in terminals (long sailing distance, unexpected weather conditions, port congestions, etc) (Luo et al., 2011). Most often, the planning of the stacking positions of export containers has to rely on the estimated time of arrival, the historical call pattern or the arrival time windows of vessels.

For import containers, the situation is reversed. Import containers arrive at terminals through vessels in large batches and depart through hinterland transport individually. The arrival of import containers can be predicted by the vessel arrival time, but the departure time is random. When the container is to be stored in the yard, the arrival time of external trucks (or trains) is generally not known to terminal operators. It is thus hardly possible to arrange the import containers in the yard without causing future relocations. Hence, when a truck arrives at the yard, it is quite often that the requested container needs to be dug out from a pile of blocking containers. According to empirical studies, in busy ports such as Los Angeles-Long Beach, it

takes on average two to three relocations to deliver one container to an external truck (Mongelluzzo, 2015a).

With the implementation of the Truck Appointment System (TAS) at container ports, truck arrival information can be obtained in advance to some extent. The TAS, also known as Vehicle Booking System (VBS), was introduced in the mid-1990s for controlling truck arrivals at the terminal to avoid congestion during peak periods (Davies, 2009). It has now been widely applied at container ports worldwide, such as Vancouver, Long Beach, Los Angeles, Felixstowe, and Southampton. The TAS is implemented in a quota system in which a certain number of appointments (e.g. 100) are allocated to each time window (also known as time slot) that is open to be booked by external trucks. When a TAS is in place at a terminal, truck drivers must book their arrival time windows before they arrive at the terminal. Hence, each import container can be matched with a truck and a corresponding retrieval time window (see Fig. 1.5). Such information can then be used for deciding efficient relocation positions for import containers. However, the exact arrival time of a truck is still uncertain, which usually can only become known when the truck has arrived at the terminal. In addition, although with a mandatory TAS, unpredictable events like road traffic congestions, driver shortage or truck breakdowns can lead to changes in the scheduled arrival time windows ("no show", "later show"). Therefore, the retrieval priorities of import containers are much less predictable than export containers.

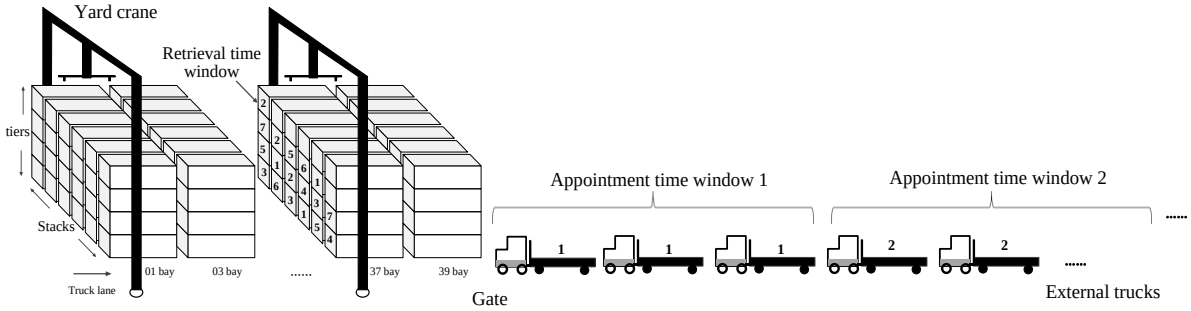


Fig. 1.5 Retrieval time windows of import container obtained by truck appointment information

1.4 Research scope and motivation

In the Section, we define the research scope of the thesis and explain the research motivation.

Among the three logistics sub-systems in a container terminal, the quayside sub-system was typically recognised as the bottlenecks that limit port performance (Forster and Bortfeldt, 2012; Hu et al., 2014). Terminal operators usually give priority to quayside operations as the efficiency of quayside operations determines the vessel turnaround time at the terminal which directly relates to the revenue generation for the terminal (Covic, 2018). As a result, quayside operations have received probably the most extensive studies in the logistics activities within container ports (see review paper by Bierwirth and Meisel, 2015).

However, the management of yard operations is of paramount importance in determining the efficiency of a terminal (Caserta et al., 2020) and has become increasingly important to the competitiveness of a container terminal (Hakan Akyüz and Lee, 2014). As new technologies and advanced equipment come into use at the seaside (e.g., new generations of QCs and indented berths) (see Gharehgozli et al., 2016), the storage yard has now been arguably perceived as the new bottleneck of terminal operations (Zhen et al., 2013; Jiang et al., 2015; Song, 2021a). With the deployment of mega container ships, the requirements for yard management have changed. Nowadays, the biggest container ship can carry up to nearly 24,000 TEUs

of containers, implying a dramatic increase in the container moves per hour at the terminal. This trend will not only demand a higher QC productivity but also require more storage capacity and productivity for the yard and better coordination of quayside and yardside operations. Without effective storage and retrieval of containers in the yard, the overall terminal performance would not benefit much from faster quayside operations (Jiang et al., 2015). In addition, with the growing container traffic, more and more containers will be stored in container yards, which can bring the risk of over-using the yard space and existing facilities. Simple capacity expansions by enlarging the terminal space is often impractical because of the constraints of the scarcity of land, availability of investment, and environmental concerns (Jiang et al., 2015). In fact, port congestion has been an industry-wide problem (Song, 2021b). Nowadays, due to the disruptive events, including the outbreak of the COVID-19 pandemic, the Suez Canal blockage and the Yantian terminal shutdown in 2021, port congestion has stretched out the globe and exacerbated the global supply chain delays. This has led to further overloading of the already insufficient yard and gate handling capacity for many ports and result in delays in releasing cargo for hinterland transportation. The congestion at yards and the incapability of landside productivity can be recognised as a leading factor for port congestion. For example, the concentration of volumes at ports caused by the COVID-19 pandemic (e.g., 10,000 TEUs per port call) has created peaks in yard activity and gate congestion in North America and Europe in the first half of 2020 (Notteboom et al., 2021). In another example, the surging ship calls at Southern Californian ports in December 2020 brought near-record levels of congestion to the ports with many ships forced to wait at anchor for several days to berth because there was nowhere to store the containers that need to be unloaded. Due to the storage space shortage, containers had to be piled five high, exacerbating the rate of relocation and adding to the truck turnaround time (Porter, 2020). Therefore, facing the challenge of the ever-growing demand and the pressure of port congestion, it is important for many ports to adjust their yard operation strategies, such as looking for new container stacking and retrieval solutions, in order to accelerate the container flow between the seaside and the landside. This is especially imperative in the current era.

Given the above challenges, this thesis focuses on yard operations management in order to improve the container handling efficiencies at the yard and at the interface between the yard and the landside. Because the major source of inefficiency in yard operations is container relocation, a significant amount of research in the literature has been dedicated to minimising the number of relocations in the yard, and different problem types have been developed and addressed. Although these operations are facing uncertainties in nature, the literature shows a dominant focus on deterministic problem context, as pointed out in Covic (2019). In this thesis, we propose novel container stacking and retrieving strategies to cope with the challenge of uncertainties. The proposed strategies utilise customer information (truck arrival information and customer identity information) and incorporate flexibility in the decision-making process. Correspondingly, two problem types are addressed, the Container Relocation Problem (CRP) for the container retrieving process (Chapters 2 and 3) and the Storage Location Assignment Problem (SLAP) (Chapters 4) for the container stacking process. Specifically, we focus on import containers, whose retrieval priority is subject to greater uncertainties and is less predictable than export containers but the majority of studies assume deterministic information.

By addressing the two types of problems, we aim to improve both the import container retrieval efficiency and the turnaround time of external trucks during retrieval. The turnaround time of external trucks is a key performance measure of terminal efficiencies and concerns the evaluation of customer satisfaction and port competitiveness (de Melo da Silva, 2018). Longer truck turnaround times can cause a serious environmental concern because of the emissions generated by truck congestions (Phan and Kim,

2016). Terminal operators have therefore been under enormous pressure from different entities who require terminals to reduce the truck turnaround time, including governments (Giuliano and O'Brien, 2007), port authorities (e.g., Port Metro Vancouver), and the stakeholders of the hinterland transport (Bonney, 2015). Therefore, the turnaround time of external trucks needs to be considered when addressing the yard operations problem.

The main research questions (RQ) of the thesis are summarised as follows:

RQ1. How to improve the import container retrieval performance during the retrieving process?

RQ2. How to improve the import container retrieval performance during the stacking process?

RQ3. What are the values of customer information to terminal yard operations?

In the following two sections, we provide a compendious literature review on the CRP and the SLAP, respectively, with the purpose to justify the research gaps and the methodologies used to solve the arising problems. A more detailed literature review on the two problems is presented in the corresponding research Chapters (Chapters 2, 3, and 4). Overviews of storage yard operations can be found in Carlo et al. (2014a), Zhen et al. (2013), Caserta et al. (2011a, 2020), Covic (2018), Luo et al. (2011), and Rekik et al. (2015). While the first two papers encompass entire aspects of yard management including the material handling equipment (yard cranes, yard vehicles, etc), the rest concentrate on container handling operations in yard blocks. Besides, a general survey of loading, unloading and premarshalling arising in practical storage areas including container terminals, container ship, warehouse and tram depots is proposed in Lehnfeld and Knust (2014).

1.5 Container relocation problem

Generally, the Container Relocation Problem (CRP) is also called the Block Relocation Problem (BRP) wherein "blocks" refers to a set of identically-sized items. The BRP is encountered in many logistics facilities such as maritime container terminals and other industries where inventory (such as boxes, pallets, containers, and steel plates) is stored in stacks and is to be retrieved on a last-in-first-out (LIFO) basis (Petering and Hussein, 2013). As the thesis is addressing the maritime industry context, hereafter, we mainly use the term CRP.

The CRP is initiated by Kim and Hong (2006) who formulate the problem with a dynamic programming model to minimise the total number of relocations to retrieve the containers with given priorities in a bay. Since this work, several variants and extensions to the CRP have appeared in the literature. Overviews of container relocation at maritime container terminals can be found in Caserta et al. (2011a, 2020). In the following, we first present a set of generic properties that basically hold for all CRP variants, followed by the assumptions in the standard CRP. Then, we introduce the variants and extensions of the CRP when some of these assumptions are relaxed or modified. Next, we classify the solution approaches for these CRPs. Finally, we justify the research gap and the corresponding methodology adopted in addressing the arising CRP in the thesis.

1.5.1 Properties and assumptions

1.5.1.1 Terminologies and generic properties of the CRP

P1. The initial configuration of the bay is given in advance. A bay consists of S stacks, T tiers, and C containers. To avoid infeasible relocations, the storage capacity of the bay is restricted to be $(S-1)T+1$ containers, that is, $C \le (S-1)T+1$.

P2. Containers to be retrieved next are called *target containers*.

P3. Relocations are performed only within the same bay.

P4. Each container is associated with a retrieval priority number, and containers with lower priorities can only be retrieved after all containers with higher priorities have already been retrieved.

P5. The situation where a container with a lower priority is stacked above a container with a higher priority is called *mis-overlay*.

1.5.1.2 Assumptions in the standard CRP

The following set of assumptions exist in the standard CRPs.

A1. A container is relocatable only when it is blocking the target container.

A2. Each container has a unique retrieval priority.

A3. The deterministic retrieval priority is given in advance.

A4. No new containers enter the bay during the container retrieval process.

A5. The relocation cost is measured only by the total number of relocations.

When some of these assumptions are relaxed or modified, variants and extensions of the CRP arise.

1.5.2 Variants and extensions

1.5.2.1 Variants

- **Restricted or unrestricted**

Under A1, only the containers above the current target container are allowed to be relocated. The CRP with this assumption is called the *restricted* CRP, which is considered in the majority of studies on the CRP. By relaxing this assumption, the *unrestricted* CRP is studied (e.g., Expósito-Izquierdo et al., 2014; Zhu et al., 2012; Jin et al., 2015; Tanaka and Mizuno, 2018).

- **Distinct or duplicate priorities**

A2 and A3 regard the container retrieval priorities, which are important aspects of the CRP. Under A2, each container has a unique retrieval priority and is regarded as a single group by itself. This type of CRP is referred to as the CRP with *distinct* priorities. On the other hand, the CRP with *duplicate* priorities assumes that more than one container can have the same retrieval priorities, which belong to the same group (Kim and Hong, 2006; Tanaka and Takii, 2016; de Melo da Silva et al., 2018). In this case, the retrieval sequence of the containers in the same group is to be determined.

- **Deterministic or uncertain priorities**

One of the central aspects of the CRP is the availability of container retrieval priorities. The accuracy of the retrieval priorities is important for avoiding mis-overlay. The retrieval priorities of containers depend on various factors, such as departure time, weight, destination port, etc. A3 regards whether the retrieval priorities of containers are deterministic or uncertain. Most of the studies on the CRP assume given *deterministic* retrieval priorities. However, since the precise departure time of a container may not be available before the container leaves the yard, the decisions made under the assumed deterministic priorities could be impractical. Only a few works have dealt with the CRP with *uncertain* retrieval priorities.

The uncertain CRPs can be categorised into two sub-categories: the online setting and the probabilistic setting. In the online setting, the container retrieval sequence is revealed over time, and the knowledge of future retrievals is limited to a look-ahead horizon. The goal is to design efficient online heuristics to determine the relocation positions for the blocking containers using information updated real-time (e.g.,

Zehendner et al., 2017; Zhao and Goodchild, 2010). In the probabilistic setting, the containers' retrieval priorities are modelled by a probability distribution and the research aim is to minimise the expectation of associated performance measures, such as the expected total number of relocations (Tong et al., 2015; Ku and Arthanari, 2016a; Galle et al., 2018a) and the weighted sum of the expected number of relocations and total retrieval delays (Borjian et al., 2013).

A more detailed literature review for the uncertain CRPs is provided in Chapter two. Here, we would like to note that the CRP that considers randomness for container retrieval order among containers in the same retrieval time window is termed as the *CRP with Time Windows* (*CRPTW*) by Ku and Arthanari (2016a) for the first time in the literature. This problem class is then referred to as the *Stochastic CRP* (*SCRP*) by Galle et al. (2018a). The CRP considered in this thesis belongs to the SCRP. In such a problem class, import containers are ordered by the arrival time windows of external trucks which are obtained through the TAS of a terminal. In this scenario, the retrieval priorities among containers in different time windows are pre-defined, but the actual arrival time or sequence of the trucks within a time window is stochastic.

1.5.2.2 Extensions

- **Static or dynamic**

The CRP under A4 is called the *static* CRP, which only involves retrieval and relocation activities. On the other hand, the *dynamic* CRP considers a dynamic environment with containers continually being stored and retrieved (Wan et al., 2009; Borjian et al., 2013; Hakan Akyüz and Lee, 2014; Tang et al., 2015). As the dynamic CRP joints retrieval and relocation of existing containers and storage of incoming containers, it can be regarded as an extension of the CRP that integrates the CRP and the SLAP. In most of the studies on the static CRP, since only the container retrieval order is given, there is no need to consider specific time points when retrieving containers. In contrast, in the dynamic CRP, the storage activity and retrieval activity may be prioritised differently by the terminal operators, therefore, time points such as discrete time steps (Borjian et al., 2013) are considered in the problem formulation.

- **Alternative objective functions**

The objective function of the standard CRP is to minimise the total number of relocations. Alternatively, objectives can be designed considering time consumption, which might be of interest for a more realistic model. A few studies take the YC working time into account, including spreader moving time for picking-up/putting-down containers, trolley moving time across stacks and gantry travel time across bays. For instance, Lee and Lee (2010) study the CRP in a three-dimensional stacking area and aim to minimise the weighted sum of the number of movements and the total working time of the YC. For a two-dimensional bay, Ünlüyurt and Aydin (2012) aim to minimise the total time spent for retrieving the containers. Voß and Schwarze (2019) investigates the CRP under different objectives by analysing to which extent the function value of the initial objective is changed when alternative objective functions are used. In addition, López-Plata et al. (2017) address the Blocks Relocation Problem with Waiting Times (BRP-WT) that focuses on minimising the total waiting times of customers during retrieval.

- **Alternative service policies**

A service policy for the CRP deals with the service sequence of transfer vehicles arriving at the yard. The service sequence determines the container retrieval sequence, which affects the number of relocations and the waiting time of transfer vehicles. However, as described above, most of the studies assume that the container retrieval sequence is pre-defined exogenously. Only a few studies have discussed service policies in the CRP.

For import containers, the most commonly used service policy for external trucks is the first-come-first-serve (FCFS) policy, that is, containers are retrieved in the arrival order of external trucks. However, the FCFS policy does not lead to the most efficient container retrievals. A few studies have proposed flexible service policies or out-of-order service, in which the container retrieval sequence (i.e., truck service sequence) is determined by the terminal operators to some extent according to the containers and the trucks to be processed (Zhao and Goodchild, 2010; Zeng et al., 2019; Borjian et al., 2013; Borjian et al., 2015). One major concern of flexible retrievals is the possibility of causing extra delays for some trucks and thus may raise the issue of service unfairness. Existing studies in this regard attempt to address this issue by restricting out-of-order service within a group of trucks in the same arrival time window (Zhao and Goodchild, 2010; Zeng et al., 2019), by setting a maximum service delay for each container (Borjian et al., 2013), or by limiting the number of out-of-order retrievals before each truck (Borjian et al., 2015). Note that there is no service fairness issue if the out-of-order retrieval is implemented to a group of containers with the same retrieval priority, for example, when they will be retrieved by the same customer (de Melo da Silva et al., 2018).

In contrast, for export containers, when the retrieval sequence is treated as a decision, it is usually the case that a group of containers have the same retrieval priority, i.e. they are exchangeable in terms of loading sequence (Kim and Hong, 2006). In this case, there is no service fairness concern. A ship stowage plan only specifies the container class for each slot in the vessel. This provides flexibility for terminal operators to determine the exact container to be loaded in each slot while optimising the operative costs that always involve the cost of relocations occurring in the yard. For example, in Monaco et al. (2014), the retrieval sequence of export containers is determined in the ship loading problem that takes into account possible relocations in the yard, to minimise the costs of yard relocations and yard-to-quay transport operations.

- **Other extensions**

Moreover, there are several other interesting extensions though seldom studied. By adding a new parameter - the weight of containers, Hussein and Petering (2012) propose the BRP with Weights (BRP-W). In this setting, the weight of containers is considered in calculating the fuel consumption for moving containers. The objective is to minimise the total fuel consumption for retrieving containers. A recent extension is investigating the situation where the crane has multi-lift capacity. For example, Lin et al. (2015) investigate the situation in which the YC has the multi-lift capability, that is, the YC is equipped with multiple spreaders and thus can carry more than one container at a time. With multiple spreaders, the crane working time can be reduced, however, this introduction brings fundamental changes into the problem as the pick-up order can be different from the put-down order. In addition, Zhang et al. (2016) investigate the BRP with batch moves (BRP-BM) that mainly arises from the operations at slab yards in steel plants. This problem addresses new bridge crane technology that enables lifting several slabs, i.e., batch moves, at a time. Finally, some authors study the integrated problem. For instance, recently, Zweers et al. (2020) present a new optimisation problem related to the SCRP, called the Stochastic Container Relocation Problem with Pre-Processing (SCRPPP), which aims to minimise a weighted average of the pre-processing moves in the pre-processing phase (when the crane is idle) and the relocation moves in the relocation phase.

1.5.3 Solution approaches

In the last two decades, operations research methods have been extensively applied to address various

operations optimisation problems at container terminals, which can be mainly categorised into exact approaches, heuristics approaches and metaheuristics approaches. In the following, the solution approaches for the CRP are introduced by these categories. A subset of studies focuses only on either exact approaches or heuristics, whereas some others introduce exact methods but also add heuristics for addressing larger problem sizes.

- **Exact approaches**

Studies in this field aim at finding an optimal solution by exploiting (i) Mathematical programming techniques, e.g., (mixed) integer programming (MIP) (Wan et al., 2009; Caserta et al. 2012; Petering and Hussein, 2013; Tang et al., 2015; Zehendner et al., 2015), dynamic programming (e.g., Kim and Hong, 2006), and stochastic dynamic programming (e.g., Ku and Arthanari, 2016a; Galle et al., 2018a); and (ii) search-based algorithms, e.g., branch and bound (Kim and Hong, 2006; Expósito-Izquierdo et al., 2015; Tanaka and Takii, 2016; Tanaka and Mizuno, 2018), branch and price (Zehendner and Feillet, 2014), branch and cut (Bacci et al., 2020), A* algorithm (Expósito-Izquierdo et al., 2014), iterative deepening A* (IDA*) algorithm (Zhu et al., 2012; Quispe et al., 2018), and tree search (Ku and Arthanari, 2016a, b; Galle et al., 2018a).

- **Heuristic approaches**

The CRP has been proven to be NP-hard by Caserta et al. (2012), therefore, only small-scale instances can be solved exactly within reasonable times. For this reason, heuristic approaches are the major solution approaches for addressing the CRP with realistic sizes. These approaches are typically greedy by exploiting (i) a set of heuristic rules and greedy indexes for choosing the containers to move and their target slots (Kim and Hong, 2006; Caserta et al., 2012; Ünlüyurt and Aydin, 2012; Hakan Akyüz and Lee, 2014); (ii) look-ahead procedure that takes into account the next moves (Jovanovic and Voß, 2014; Petering and Hussein, 2013); and (iii) heuristic tree search procedure that explores only a subset of the search tree (Wu and Ting, 2010; Forster and Bortfeldt, 2012; Zhu et al., 2012; Expósito-Izquierdo et al., 2014; Jin et al., 2015; Bacci et al., 2019).

- **Metaheuristic approaches**

Metaheuristic approaches for the CRP apply intelligent strategies to generate candidate bay configurations, including (i) the corridor method by Caserta et al. (2011b) in which a dynamic programming algorithm is used in a metaheuristic fashion; (ii) the genetic algorithm by Maglić et al. (2020); (iii) the tabu search algorithm by Wu et al. (2010); (iv) the ant colony algorithm by Jovanovic et al. (2019); and (v) the rake search by Tricoire et al. (2018).

Recently, machine learning (Zhang et al., 2020; Jiang et al., 2021) has also been used to solve CRP.

1.5.4 Research gap and methodology justification

1.5.4.1 Research gap

When dealing with the CRP, solution approaches for the deterministic version have received the most attention over the last decade. However, the assumptions on deterministic information may jeopardise the practicality of the solutions. Approaches for the uncertain version of the CRP have started to receive attentions in recent years, though they are still understudied. The Stochastic Container Relocation Problem (SCRP) that considers the randomness of the arrival order of trucks within a time window can be identified as a promising research trend, as demonstrated by Ku and Arthanari (2016a) and Galle et al. (2018a). In the SCRP, traditionally, it is assumed that all scenarios of the arrival order of the trucks within a time window

have an equal probability. This assumption overlooks the customer arrival preference. Customers (i.e., trucks) may have different arrival probabilities over different segments of a time window, which has not been considered in the previous SCRP studies. In addition, applying flexible service policies can bring visible benefits to a terminal, which is worth receiving more attention from both academia and industry practitioners. However, as demonstrated in the literature, only a few studies have considered service flexibility in the CRP (Zhao and Goodchild, 2010; Zeng et al., 2019; Borjian et al., 2013; Borjian et al., 2015; de Melo da Silva et al., 2018). No studies have explicitly applied flexible service policies in the SCRP.

A major concern of applying flexible service policies is that it might raise the issue of service fairness, because out-of-order service can cause some early arriving trucks to be served later than some later arriving trucks. The issue of service fairness to customers may become a barrier to the implementation of the flexible service policy. Considering fairness issues in the container retrieval operation can bring tangible benefits to terminal operators, such as improving service reliability and customer satisfaction. However, to the best of our knowledge, only Borjian et al. (2015) have evaluated the impact of the relocation-minimisation-oriented solutions on the service equity (fairness) in the deterministic version of CRP. No studies have explicitly optimised service fairness in the CRP.

The two papers in Chapters 2 and 3 address the SCRP by applying flexible service policies. The research gaps to fill in each paper can be stated as the research objectives (RO) below:

RO1. Optimise the import container retrieval performance during the retrieving process under uncertain truck arrivals by applying flexible service policies and utilising customer preference information. It is mainly from the terminal operator's perspective (Paper 1 in Chapter 2).

RO2. Optimise operation efficiency and service fairness during the import container retrieving process under uncertain truck arrivals. It takes from both the terminal operator and customers' perspectives (Paper 2 in Chapter 3).

RO3. Evaluate the value of customer arrival preference to operation efficiency during the retrieving process (Paper 2 in Chapter 3).

1.5.4.2 Methodology justification

The SCRPs studied in Chapter 2 and Chapter 3 are both formulated by stochastic dynamic programming. The rationale for choosing this methodology is justified below.

Optimisation under uncertainty has long been a focus of the mathematical programming community. Different approaches have been proposed, which, most prominently, include stochastic optimisation and robust optimisation. As a direct extension of deterministic optimisation, stochastic optimisation adds uncertainty to the problem by modelling the uncertain parameters as a random variable whose probability distribution is known to the decision-maker. The objective function of traditional stochastic optimisation is usually minimising the expected value of the objective function of the deterministic version of a problem (Bakker et al., 2020). In contrast to stochastic optimisation, robust optimisation presents uncertainty by an uncertainty set that contains all possible scenarios of the uncertain parameters but does not require knowledge of the likelihood of each scenario. The decision-maker wants to construct a solution that is feasible for any realisation of the outcomes in the given uncertainty set. The main paradigm of robust optimisation aims to optimise the objective function of the worst-case (Gabrel et al., 2014). As such the solution can be very conservative depending on the specific application context. Last, another prevalent concept is algorithm-based online optimisation that evolves from the field of computer science (Bakker et al., 2020). It does not take into account any information on the probability of future outcomes and focuses on devising online algorithms that can prove quality guarantees when competing with an optimal offline

algorithm that knows the whole input parameters in advance.

Different concepts have been taken in tackling the container handling problems in yard blocks under uncertainties and various approaches have been used. Stochastic Dynamic Programming (SDP) is a combination of dynamic programming and stochastic programming, which belongs to the concept of stochastic optimisation. Kim et al. (2000) firstly use the SDP to address the export container stacking problem. Later, their model is corrected by Zhang et al. (2010). More recently, the SDP method has been used to tackle the SCRP by Ku and Arthanari (2016a). Later, Galle et al. (2018a) extends the problem in Ku and Arthanari (2016a) by incorporating more information on the truck arrival order, which is also formalised by the SDP concept, albeit the authors do not explicitly name their model as SDP. Besides, dynamic programming has also been used to solve the deterministic CRP by Kim and Hong (2006). Alternatively, a few studies use robust optimisation for tackling the Container Premarshalling Problem (CPMP), which is closely related to the CRP as described in Section 1.3. This line of research paradigm is currently represented by three papers, i.e., Rendl and Prandtstetter (2013), Tierney and Voß (2016), and Boge et al. (2020). In these papers, the retrieval priorities of export containers are uncertain due to the uncertainty in vessel arrivals. Such uncertainty is represented in the three papers, respectively, by intervals of possible priority values, time intervals within which containers will be retrieved, and scenarios of finite permutation of priority classes. In addition, some studies apply the recoverable robust optimisation approach for solving the operations optimisation problems at containers terminals, such as Iris and Lam (2019) for the berth and quay crane planning problem. Recoverable robustness combines the flexibility of stochastic programming with the tractability and performances guarantee of the classical robust approach. This approach aims to generate a recovery robust solution that can be recovered by limited costs in all likely scenarios (Liebchen et al., 2009). Moreover, in order to have reactive decisions, some studies use fuzzy logic-based expert systems for providing real-time decision support to deal with the uncertainty in the arrival and departure of containers at yards. In this approach, each input variable (e.g., weight, distance, dwell time) is mapped with a membership function (e.g., small, medium, large), and a set of rules are defined to determine the solution. Representative studies using this approach include Covic (2017) for the container re-marshalling problem and Ries et al. (2014) for the container stacking problem. In addition, multi-agent decision support systems have been developed for the reactive and decentralized control of container stacking in an uncertain and disturbed environment (e.g., Rekik et al., 2016; Rekik and Elkosantini, 2019). Such systems are integrated with a set of knowledge models and learning mechanisms for disturbance and reactive decision-making management. The goal of the system is to respond to disturbances (e.g., yard breakdown, fault in container placing, container's date out changes) in a way that minimises their impact, thus to avoid the need for re-planning.

This thesis addresses the SCRP by taking the concept of stochastic optimisation. The selection of this concept is driven by the characteristics of the uncertainty in the problem and the goal of the decision-making. The source of uncertainty in the SCRP stems from the randomness of truck arrivals within the same time window. Such uncertainty presents only in groups of containers/trucks while not across groups, as we assume that the truck arrival intervals (i.e., time windows) are already known through the TAS and a truck will arrive within its booked time window for sure without deviation to another time window. The range of this uncertainty tends to be small since a truck will arrive within a certain time window with a shorter length (e.g., 30 minutes or 60 minutes). In this context, the terminal operator may want to seek a risk-neutral solution instead of a conservative solution. Therefore, the goal of the SCRP is to find an optimal solution, i.e, a sequence of container retrieval and relocation moves, for each realisation of scenario, in order to minimise the expected total number of relocations. In this sense, it is different from the

goal of the robust approach that aims to find a feasible solution that can overcome the uncertainty in the worst-case, as applied in the CPMP. In the problem setting of the robust CPMP, the deviation of the actual arrival time of a vessel from its expected arrival intervals can have a large impact on the feasibility and optimality of a solution since many containers will be loaded onto the same vessel, therefore, robust optimisation is proper to seek a conservative solution. Also, the goal of the SCRP is different from the recoverable robust optimisation that has a focus on the recovery costs from the baseline schedule, and is different from the reactive decision support system that is designed for reacting after the fact of unexpected events.

The SCRP considered in this thesis falls into the category of multi-stage sequential decision-making problems. In this kind of problem, information on the stochastic outcomes is revealed gradually and a sequence of decisions in reaction to the realisations of the outcomes is made over time (Birge and Louveaux, 2011). For a structuring review on multi-stage optimisation under uncertainty, we refer to Bakker et al. (2020). As pointed by Bakker et al. (2020), SDP is an appropriate method to deal with the multi-stage sequential decision-making problem in the case the problem has a Markovian process structure. The SCRP studied in the thesis possesses the Markov property, that is, the future bay configurations during the retrieval process depends only upon the present state of the bay and action and not on the past state, which justifies the usage of SDP to formulate the problem. Alternative formulations could be using multi-stage stochastic programming, but the practical solvability of multi-stage stochastic programming might be strongly restrained due to large dimensions. While the SDP has the advantage of dealing with a problem with multi-stages due to the optimal substructure of the problem at each stage.

In terms of the solution approach, by applying the recursive equation of the SDP model, the optimal solutions can be obtained backwards from the final stage to the initial stage. This procedure is usually executed by a tree search-based algorithm in a state-space constructed by a decision tree. In this thesis, we use a tree search-based algorithm to find the optimal solution. When the problem gets larger, the algorithm is not able to find the optimal solution within a limited time. We thus develop two rule-based heuristic algorithms to overcome the computational limits of the exact algorithm. The NP-hardness of the CRP justifies the usage of heuristic algorithms to address realistic problems of larger sizes.

1.6 Storage Location Assignment Problem

The determination of the storage locations for incoming containers into the yard is usually addressed hierarchically in two decision problems: the Storage Space Allocation Problem (SSAP) and the Storage Location Assignment Problem (SLAP). The SSAP determines the amount of yard storage space allocated to each vessel for their containers, which can be addressed at various levels according to the storage space unit considered: yard section, yard block, yard sub-block, and yard bay (Jin et al., 2016). The SLAP deals with the assignment of individual containers to specific slots – which is specified by a bay number, a row number, and a tier - in a block. For a recent comprehensive review on yard space and storage location planning, we refer the readers to Kizilay and Eliiyi (2020).

In this thesis, we address the SLAP in Chapter 4, which focuses on the short-term operational decision to assign import containers to exact slots in a given storage area of a yard block. In the following, we first introduce the common container stacking strategies. Then, we classify the SLAP into two types based on the planning approaches, followed by a summary of different solution approaches. Finally, we justify the research gap and the corresponding methodology adopted in addressing the arising SLAP in the thesis.

1.6.1 Stacking strategies

Container stacking strategies are a set of rules or criteria that should be adhered to when determining the storage position of a container or the storage space of a group of containers. They are somewhat tactical level decisions of container terminals (Maldonado et al., 2019), which influence the allocation of stacking positions at the operational level. By avoiding stacking containers on top of those that will be retrieved earlier, an appropriate stacking strategy can reduce the number of relocations. Several types of stacking strategies have been applied in practice and studied in the literature, which usually differ between export containers and import containers due to their different arrival and departure characteristics.

Export containers usually arrive at terminals individually and are loaded onto vessels in batches. The arrival times of export containers are uncertain but their departure times are relatively fixed by the destination vessels. There are several types of stacking strategies for export containers. Depending on the type of information utilised to categorise containers, these stacking strategies include (i) the residence time stacking strategy that utilises the containers' residence/departure times (e.g., Borgman et al., 2010); and (ii) the category stacking strategy where containers are categorised according to the containers' attributes, such as the weight class, the port of destination, and the type of container (e.g., Dekker et al., 2006; Kim et al., 2000; Kang et al., 2006; Guerra-Olivares et al., 2018). On the other hand, depending on whether containers to different ships are allowed to share a stack, dedicated stacking strategies and shared stacking strategies are differentiated. Under the dedicated stacking strategy, containers to different ships cannot stack on top of each other. While the shared stacking strategy allows containers to different ships to be stacked on top of each other by considering the departure time of containers, which can better utilise the yard space (e.g., Gharehgozli et al., 2014; Gharehgozli and Zaerpour, 2018).

Import containers are unloaded from vessels in large volumes and then picked up by customers individually and randomly. Due to the high uncertainty in the retrieval sequences of import containers, it is not possible to stack import containers according to their individual departure times. The stacking strategies for import containers are usually based on the arrival ships of the containers, and two types of strategies are differentiated: segregation strategy and non-segregation strategy (De Castilho and Daganzo, 1993; Saur í and Mart ń, 2011; Yu and Qi, 2013). Under the segregation strategy, containers from different ships are stacked separately in the container yard. Under the non-segregation strategy, containers from different ships are mixed in the storage area such that newly discharged containers are stacked on top of old ones. The segregation strategy may have the advantage of reducing the number of relocations during the container retrieval process because earlier-arrived containers are likely to be retrieved earlier; but it requires additional clearing moves before each ship's arrival to create enough space for the new containers. On the other hand, the non-segregation strategy would increase relocation moves because the containers that have stayed for a longer time and thus tend to be picked up soon will be buried under recently arrived ones. A few studies develop more detailed stacking strategies based on the container departure dates (e.g., Guldogan, 2011) and the estimated dwell times of import containers (e.g., Lee et al., 2008; Gaete et al., 2017; Maldonado et al., 2019), where containers with longer dwell times are stored under those with smaller values to reduce the number of future relocations.

1.6.2 Online SLAP and offline SLAP

The SLAP may be classified into two broad categories according to the planning approaches: online planning and offline planning.

The online planning approach uses online heuristic stacking rules to allocate containers to slots in real-time by considering the dynamic characteristics of the problem and the uncertain information on

containers (e.g, Park et al., 2011; Lin et al., 2017; Petering et al., 2017; He et al., 2019). Simulation is a prevalent approach to evaluate a set of heuristic stacking rules (e.g., Dekker et al., 2006; Borgman et al., 2010; Guldogan, 2011). In addition, decentralized approaches such as case-based reasoning (Rekik et al., 2018) and multi-agent approach (Rekik and Elkosantini, 2019) are developed for the reactive container stacking systems that face uncertainties and disturbances during the container stacking process.

The offline planning approach focuses on finding an optimal plan at the beginning of the planning period for an offline environment where the input data of the defined problems are known. The SLAP in Chapter 4 follows this research stream. In the offline SLAPs, minimising the number of relocations during the future retrieval process is an important objective function when the container retrieval efficiency is the focus. In order to avoid future relocations, accurate data on container retrieval sequence is the most crucial information that is needed (Kemme, 2020). As such, the optimisation models for the SLAPs of export containers have been relatively well defined because when export containers are to be stacked, their retrieval sequence can often be pre-determined to some extent by certain criteria. Such criteria include vessel loading schedules (Preston and Kozan, 2001), containers' weight class (Kim et al., 2000; Zhang et al., 2010; Kang et al., 2006; Zhang et al., 2014), the vessel departure time windows (Gharehgozli and Zaerpour, 2018), etc. In contrast, when import containers are to be stacked, it is often hard to know when they will leave the yard, therefore, it is not possible to optimally arrange their stacking slots according to their retrieval time. A few studies model the SLAP by assuming that the exact retrieval time/sequence is known so that the number of relocations can be measured exactly (Chang and Zhu, 2019; Wang et al., 2020) or relocations can be completely avoided (Razouk et al., 2016). A recent study assumes that the group retrieval priorities are given by the truck arrival time windows (Zhu et al., 2020).

1.6.3 Solution approaches

As described above, online heuristic stacking rules and simulation are prevalent approaches to cope with the uncertainties and computational complexity of the container stacking problem. The majority of the studies on the offline SLAP assumes deterministic information and apply deterministic optimisation approaches. From the mathematical modelling perspective, these approaches include integer programming (IP) (e.g., Chang and Zhu, 2019; Zhu et al., 2020) and mixed-integer programming (MIP) (e.g., Preston and Kozan, 2001; Razouk et al., 2016; Gharehgozli and Zaerpour, 2018; Wang et al., 2020). From the solution method perspective, metaheuristics (e.g., Preston and Kozan, 2011; Razouk et al., 2016; Gharehgozli and Zaerpour, 2018; Wang et al., 2020) and rule-based heuristics (e.g., Chen and Lu, 2012; Li et al., 2017; Chang and Zhu, 2019) are mostly applied. A few authors use commercial solvers such as CPLEX to solve their MIP or IP models only for small-scale problem instances (e.g., Razouk et al., 2016; Gharehgozli and Zaerpour, 2018).

Only a few studies address the offline SLAP under uncertainties. In this research stream, stochastic dynamic programming models (Kim et al., 2000; Zhang et al., 2010; Zhang et al., 2014) and simulated annealing algorithm (Kang et al., 2006) are proposed to determine the stacking positions of export containers based on the uncertain weight class information of containers.

1.6.4 Research gap and methodology justification

1.6.4.1 Research gap

When dealing with the offline SLAP of import containers, most of the studies assume that container retrieval priorities are given. However, in reality, when import containers arrive at the yard to be stacked, it is often not clear exactly when they will be retrieved. For terminals that are equipped with a TAS, although

the retrieval priorities among groups of containers can be obtained from the truck appointment information, such information is only available after the containers have been stacked in the yard. This is because the fact that, in most cases, appointments are not bookable until the container has already been customs cleared for pick-up, which can be days after the container have been stacked in the terminal (e.g., DP World London Gateway). Therefore, it is not possible to optimally stack import containers according to their retrieval time.

To improve container pickup efficiency and reduce truck waiting times, an innovative container delivery and staging program — Import Free Flow (IFF) — has been initiated in practice at Port of Los Angeles in 2015 (Mongelluzzo, 2015b). The idea behind IFF is to eliminate the need for relocation through pre-staging large groups of containers to be picked up by the same customer (i.e., free-flow containers) in order to realise rapid retrieval flow. Container stacking is an important problem involved in running the IFF program. Currently, there is no work that investigates how to plan container stacking under the IFF program. The current IFF practice works with predefined rules and is inadequate for its mass application. For example, the free-flow service is only available to high-volume customers who own at least 50 containers, and the free-flow containers are simply pre-staged in specific stacks that are separated from the traditional containers (i.e., non-free-flow containers) (Parker, 2015). There is a need for an optimal method to determine which customers or containers should be free-flowed, how many free-flow stacks should be selected, and where these stacks should be located in the yard. A key decision is to group containers. There are only two most relevant studies that have investigated grouped-based stacking strategies for import containers, which involve the yard slot decision (Jang et al. 2013) and the yard bay decision (Yu and Qi, 2013). However, neither of them incorporates the concept of free-flow.

Motivated by the gap between academic research and the IFF practice, we conceptualise a new import container stacking strategy - smart stacking strategy - where import containers are classified into either smart (free-flow) or non-smart (non-free-flow) containers based on customer identity information and are allocated to smart stacks and non-smart stacks respectively in a yard block in an optimal way.

Chapters 4 addresses the SLAP by applying the smart stacking strategy. The research gaps to fill can be stated as the research objectives (RO) below:

RO1. Optimise the import container retrieval performance during the stacking process by utilising customer identity information.

RO2. Evaluate the value of the customer identity information to terminal operators.

1.6.4.2 Methodology justification

The SALP studied in this thesis is formulated as MIP models, which are solved by commercial software CPLEX for small-scale problem instances. A divide-and-conquer heuristic algorithm is developed to obtain near-optimal solutions for large-scale problem instances. The rationale is justified below.

The uncertainty in our SLAP is concerned with the expected number of future relocations, which is dealt with by estimation. Consequently, the problem can be regarded as a deterministic optimisation problem, therefore, it can be formulated by a MIP as commonly did in the literature on offline SLAPs (e.g., Preston and Kozan, 2001; Razouk et al., 2016; Gharehgozli and Zaerpour, 2018; Wang et al., 2020). How the uncertainty is dealt with is briefly explained here. Under the proposed smart stacking strategy, the containers in a smart stack are retrieved from the top to the bottom of the stack, meaning that there is no uncertainty regarding the retrieval priorities of the containers in a smart stack; while the containers in a non-smart stack will be retrieved in random order. For the containers in a non-smart stack, due to the uncertainty in the trucks arrival order, the number of relocations during the retrieval process cannot be easily determined in advance (Bruns et al., 2016). In order to overcome the unavailability of the retrieval

priority, we estimate the expected number of relocations of a non-smart stack using the lower bound of the expected number of blocking containers proven by Galle et al. (2016). This lower bound has also been used in a recent study on yard crane scheduling by Galle et al. (2018b) to approximate the number of relocations to retrieval all the containers in a single stack when no information is assumed on the retrieval requests.

We prove that our SLAP is NP-hard in general. Due to the NP-hardness, the proposed MIPs are computationally expensive to solve for large-scale problems, which drives us to seek heuristic algorithms. The developed heuristic algorithm is based on the principle of divide and conquer. The divide-and-conquer paradigm is a commonly used strategy to design efficient algorithms for complicated combinatorial optimisation problems (e.g., Reimann et al., 2004; Jin et al., 2016; Wei et al., 2019). Its principle is to decompose the original problem into two or more subproblems until they become sufficiently simple to be solved directly; the subproblems are solved independently and their solutions are composed to give a solution to the original problem (Smith et al., 1985). The structure of the developed MIP motivates us to employ the divide-and-conquer strategy to decompose the original problem into several subproblems and solve them sequentially and iteratively. It is observed that in the MIP, the decision vector about the smart stacks and the decision vector about the non-smart stacks are coupled only by a few constraints. By relaxing these constraints, the original problem can be decomposed into two smaller subproblems: one involving only the smart stack decisions and the other only concerning the non-smart decisions, which are much easier to solve. The solution to the original problem can be obtained by solving a third subproblem in which the solutions to the first two subproblems are combined.

1.7 Impact of new storage systems

Recently, there are some innovations on new storage systems at container terminals with a particular focus on vertical expansion, such as the rack storage systems and container tower (see Gharehgozli et al., 2020). The new storage systems are higher and more compact by storing containers in tall structures. For example, the prototype of an ultra-high container warehouse system of Ez-Indus in South Korea can stack containers up to 50 tiers high (Gharehgozli et al., 2016). The trial "Boxbay" vertical storage system installed by DP World can stack containers 11 high in racks (DP World). The rack-based storage system is designed such that each container is placed on an individual chassis/slot in the rack and can be accessed directly without having to move other containers, which is similar to the warehouse environments. Therefore, no relocations are needed in such systems.

The new storage systems would revolutionise the way containers are handled in terminals and may result in a higher terminal efficiency. However, many are still at the concept and experimental stage (Gharehgozli et al., 2020). For example, the “BoxBay” of DP World has completed a six-month successful trial in 2021, but the parties have not yet detailed plans on next steps (THE MARITIME EXECUTIVE, 2021). In addition, implementing such systems can be costly, not only due to the high investment cost, but also the potentially high energy consumption as large and heavy containers are lifted to quite high racks. Although power regeneration and solar panels on the roof can help improve energy efficiency (e.g., the Boxbay of DP World), such technologies may not be applicable to each system and could be expensive. The potential of the new storage systems still needs to be evaluated.

Researchers have started to investigate the design and performance of the new vertical systems (e.g., Zaerpour et al., 2019), but it is just beginning. As pointed out by Gharehgozli et al. (2020), in order to realise and implement new layouts and systems at container terminals, challenges at strategic and tactical levels (such as the optimal configuration and financial feasibility of a layout) and important operational

problems (including the impact of design variables on the performance of a layout) need to be addressed. In addition, safety and security may be the top concerns. Container terminals are vulnerable to weather disruptions such as hurricanes and typhoons. How the new high storage systems can withstand such disruptive events need to be ensured.

The advent of the new storage systems has the potential to radically transform the nature of the problem studied in the thesis since relocations are eliminated. However, the profitability and sustainability of such new systems in the long term are still unknown, therefore the work in the thesis is still significant and valuable. Under the context of the new systems, for the container stacking and retrieval problems, the energy consumption could be very relevant since the containers are stored at quite high positions. One of the important questions therefore could be where to put the containers in the rack to minimise the energy consumption and emissions.

1.8 Thesis structure

The thesis is composed of five chapters. Chapter 1 introduces the research context, identifies the research gaps, and justifies the research methods. The middle three chapters correspond to three research papers (two published and one under review), in which Chapters 2 and 3 address the Container Relocation Problem (CRP) and Chapter 4 addresses the Storage Location Assignment Problem (SLAP). Chapter 5 concludes the thesis and provides future research directions.

All three papers focus on improving the import container retrieval performance, but the ways how this is achieved are different. On one hand, Paper 1 and Paper 2 aim to achieve this by optimising the container relocation and retrieval operations through addressing the Container Relocation Problem (CRP). These two papers are developed progressively. Paper 1 focuses on the overall operational efficiency of container retrieval, including the total number of relocations and the average truck waiting times, which is mainly from the terminal operator's perspective. Paper 2 is built upon Paper 1 by adding the performance measures of service fairness from the customer service perspective. On the other hand, Paper 3 aims to improve the import container retrieval efficiency by optimising the container stacking process through addressing the Storage Location Assignment Problem (SLAP). Besides the relocation time, the travel time of the yard crane for moving containers from their storage slots to the transfer points is also incorporated into the objective function.

The problems in the three papers are solved by different OR methods. Some of the methods are exact solution methods that can find optimal solutions, which aims at increasing the methodological knowledge about the studied problems and providing benchmarks. Motivated by the characteristics of the studied problem, stochastic dynamic programming, tree search-based algorithm, and mixed-integer programming are used. Others are heuristic methods that aim to obtain efficient solutions rapidly, which include rule-based heuristic algorithms and a divide-and-conquer heuristic algorithm. Although the developed heuristic methods may not be readily applicable in practice as the real context can involve more complexities, the ideas in the methods can provide terminal operators with insights for designing decision support tools. Besides, from a practical perspective, the fast speed of the heuristic algorithms can meet the need for frequent or even real-time decision makings, as the studied problems are all at the operational planning level that has a short planning horizon. According to industry experts, a solution for real-time decisions at yards should take no more than a minute (Song, 2021b).

In the following, the outline of the three research chapters/papers is presented, along with the introduction of the dissemination of each paper.

Chapter 2: The stochastic container relocation problem with flexible service policies. (i) We apply a flexible service policy to the Stochastic Container Relocation Problem (SCRP) and extend the SCRP to the *SCRP with Flexible Sevice policies* (*SCRP-FS*), which provide more opportunities for improving the import container retrieval performance. We generalise the probabilistic model of truck arrivals so that the customers' preference-based arrival behaviour can be captured more accurately. (ii) We introduce a new model that jointly optimises the expected number of relocations and the truck waiting times. (iii) We develop exact algorithms by extending an existing algorithm with major adaptions. The incremental contributions of our exact algorithms include a decision tree with a new structure, a new lower bound for the expected number of relocations in the SCRP-FS, and a procedure for minimising the truck waiting times. We also develop two heuristic algorithms that can obtain efficient solutions in a matter of milliseconds. (iv) We construct a discrete-event simulation model for evaluating the two optimised performance measures by using the optimal solutions derived from the decision tree. The framework enables terminal operators to quantify the benefits of the flexible service policy to both terminals and truckers. (v) We conduct extensive experiments to demonstrate the effectiveness of the flexible service policy and the influence of customer arrival preference on the results. The findings can provide managerial insights for terminal operators to manage import container relocation and retrieval operations more efficiently. This work has been disseminated as follows:

Feng, Y., Song, D. P., Li, D., & Zeng, Q. (2020). The stochastic container relocation problem with flexible service policies. *Transportation Research Part B: Methodological*, 141, 116-163, published.

Feng, Y., Song, D., Li, D., & Zeng, Q. (2019). The stochastic container relocation problem under a flexible service policy considering truck waiting times. *The 9th International Conference on Logistics and Maritime Systems (LOGMS)*, Singapore, August 2019, presented, First Place Winner of Best Paper Competition (Student) Award.

Feng, Y., Song, D., Li, D., & Zeng, Q. (2019). The impact of flexible retrievals on import container relocation at container terminals. *The 10th International Conference on Systematic Innovation*, Liverpool, UK, July 2019, presented, Excellent Presentation Award.

Feng, Y. Managing import container relocations at container terminals under uncertainty. *The 24th European Logistics Association (ELA) Doctoral Workshop*, Edinburgh, UK, June 2019, presented.

Feng, Y., Song, D., Li, D., & Zeng, Q. The stochastic container relocation problem with flexible service policy. *Postgraduate Research Showcase across the Faculty*, University of Liverpool, June 2019, poster presented.

Feng, Y., Song, D., Li, D., & Zeng, Q. (2019). The stochastic container relocation problem considering customer preference and flexible service. *The Transportation Research Board (TRB) 98th Annual Meeting*, Washington, D.C, January 2019, presented.

Chapter 3: Service fairness and value of customer information for the stochastic container relocation problem under flexible service policy. (i) This paper extends the SCRP-FS to the case with multiple sub-time windows, which is termed as the *SCRP with Multiple sub-time windows-based Flexible Service policy* (*SCRP-MFS*). The SCRP-MFS optimises both the relocation efficiency and service fairness, which is the first in the CRP studies. (ii) This paper investigates the impact of the customer preference information (i.e., truck arrival probabilities for different sub-time windows) on relocation efficiency. (iii) The results demonstrate how the operation efficiency of container retrieval and the quality of service to individual trucks are traded off under different levels of service flexibilities, which can help the terminal

operators to determine the appropriate number of sub-time windows when applying the flexible service policy. (iv) The value of customer preference information demonstrated in different scenarios can provide terminal operators with insights into whether it is worth committing effort to capture such information. This work has been disseminated as follows:

Feng, Y., Song, D.P., Li, D., & Xie, Y. (2021). Service fairness and value of customer information for the stochastic container relocation problem under flexible service policy, *Transportation Research Part E: Logistics and Transportation Review*, under the second-round review.

Chapter 4: Smart stacking for import containers using customer information at automated container terminals. (i) We propose a new stacking strategy – Smart Stacking (SS) strategy - to improve the import container retrieval efficiency. (ii) We introduce two forms of stacking policies under the SS strategy, depending on whether the containers from the same customer are allowed to be split between smart stacks and non-smart stacks or not. Correspondingly, we formulate two variants of the Storage Location Assignment Problem (SLAP), that is, the non-split model and the split model. The proposed models enable terminal operators to determine which customers or containers should be dedicated to smart stacks and to quantify the advantage of the splitting policy over the non-splitting policy. (iii) We derive structural properties of the optimal solution to the non-split model and then make use of these properties to improve the computational efficiency of the model. (iv) To overcome computational complexity, we develop a heuristic algorithm to solve the non-split model (which is the focus of this paper) based on the structure of the model. The heuristic algorithm can obtain near-optimal solutions in a few seconds. (v) We conduct extensive experiments to demonstrate the effectiveness of the SS strategy, and the impact of the customer information and the yard utilisation rate on the results. The findings can help terminal operators to understand the effectiveness of the SS strategy under a variety of scenarios and assess the value of customer information to container retrieval efficiency, which could promote the vertical collaboration between terminal operators, trucking companies, and cargo owners to improve supply chain performance. This work has been disseminated as follows:

Feng, Y., Song, D. P., & Li, D. (2021). Smart stacking for import containers using customer information at automated container terminals, *European Journal of Operational Research*. In Press. https://doi.org/10.1016/j.ejor.2021.10.044.

Feng, Y. Improving Import Yard Operations at Maritime Container Terminals under Uncertainty. *The 25th European Logistics Association (ELA) Doctoral Workshop*, Helsinki, Finland, October 2021, accepted.

1.9 Appendix

Table 1.1. Top 10 global container ports, 2018 and 2019-rank, throughput in million TEUs, and percentage change (Source: Lloyd's List One Hundred Ports, 2020)

Rank (2019)	Port	Country	2019 (Throughput)	2018 (Throughput)	2018-2019 (Percentage change)
1	Shanghai	China	43.303	42.010	3.1
2	Singapore	Singapore	37.196	36.599	1.6
3	Ningbo-Zhoushan	China	27.530	26.351	4.5
4	Shenzhen	China	25.770	25.740	0.1

5	Guangzhou	China	23.236	21.922	6.0
6	Busan	South Korea	21.992	21.663	1.5
7	Qingdao	China	21.010	19.315	8.8
8	Hong Kong	China	18.361	19.596	-6.3
9	Tianjin	China	17.264	15.972	8.1
10	Rotterdam	The Netherlands	14.811	14.513	2.1

References

Agra, A., & Oliveira, M. (2018). MIP approaches for the integrated berth allocation and quay crane assignment and scheduling problem. European Journal of Operational Research, 264(1), 138-148.

Angeloudis, P., & Bell, M. G. (2010). An uncertainty-aware AGV assignment algorithm for automated container terminals. Transportation Research Part E: Logistics and Transportation Review, 46(3), 354-366.

Angeloudis, P., & Bell, M. G. (2011). A review of container terminal simulation models. Maritime Policy & Management, 38(5), 523-540.

Bacci, T., Mattia, S., & Ventura, P. (2019). The bounded beam search algorithm for the block relocation problem. Computers & Operations Research, 103, 252-264.

Bacci, T., Mattia, S., & Ventura, P. (2020). A branch-and-cut algorithm for the restricted Block Relocation Problem. European Journal of Operational Research, 287(2), 452-459.

Bakker, H., Dunke, F., & Nickel, S. (2020). A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. Omega, 96, 102080.

Birge, J.R., Louveaux, F. (2011). Introduction to stochastic programming (Second Ed). Springer, New York.

Bierwirth, C., & Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research, 202(3), 615-627.

Bierwirth, C., & Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research, 244(3), 675-689.

Boge, S., Goerigk, M., & Knust, S. (2020). Robust optimization for premarshalling with uncertain priority classes. European Journal of Operational Research, 287(1), 191-210.

Bonney, J. (2015). US ports move toward truck appointment model. JOC.com. Apr 27, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-new-york-and-new-jersey/us-ports-move-toward-truck-appointment-model_20150427.html. Accessed February 23, 2020.

Borgman, B., van Asperen, E., & Dekker, R. (2010). Online rules for container stacking. OR Spectrum, 32(3), 687-716.

Borjian, S., Manshadi, V., Barnhart, C., & Jaillet, P. (2013). Dynamic Stochastic Optimization of Reshuffles in Container Terminals. In Manufacturing and Service Operations Management (MSOM) conference.

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). Managing relocation and delay in container terminals with flexible service policies. arXiv preprint 1503.01535.

Bruns, F., Knust, S., & Shakhlevich, N. V. (2016). Complexity results for storage loading problems with stacking constraints. European Journal of Operational Research, 249(3), 1074-1081.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. European Journal of Operational Research, 219(1), 96-104.

Caserta, M., Schwarze, S., & Voß, S. (2011a). Container Rehandling at Maritime Container Terminals. In J.

W. Böse (Ed.), *Handbook of Terminal Planning*, Operations research/computer science interfaces series 49 (pp. 247-269). Springer, New York.

Caserta, M., Voß, S., & Sniedovich, M. (2011b). Applying the corridor method to a blocks relocation problem. OR Spectrum, 33(4), 915-929.

Caserta, M., Schwarze, S., & Voß, S. (2020). Container rehandling at maritime container terminals: A literature update. In J. W. Böse (Ed.), *Handbook of Terminal Planning (2nd ed.)*, Operations Research/Computer Science Interfaces Series 64 (pp. 343-382). Springer, Cham.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014a). Storage yard operations in container terminals: Literature overview, trends, and research directions. European journal of operational research, 235(2), 412-430.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014b). Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. European journal of operational research, 236(1), 1-13.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2015). Seaside operations in container terminals: literature overview, trends, and research directions. Flexible Services and Manufacturing Journal, 27(2), 224-262.

Chambers, S. (2021). Global liner congestion worsens, 116 ports report disruption. Splash247.com. July 21, 2021. Retrieved from https://splash247.com/global-port-congestion-worsens-116-ports-report-disruption/. Accessed September 2, 2021.

Chang, Y., & Zhu, X. (2019). A Novel Two-Stage Heuristic for Solving Storage Space Allocation Problems in Rail–Water Intermodal Container Terminals. Symmetry, 11(10), 1229.

Chen, L., & Lu, Z. (2012). The storage location assignment problem for outbound containers in a maritime terminal. International Journal of Production Economics, 135(1), 73-80.

Chen, J. H., Lee, D. H., & Cao, J. X. (2012). A combinatorial benders' cuts algorithm for the quayside operation problem at container terminals. Transportation Research Part E: Logistics and Transportation Review, 48(1), 266-275.

Chen, G., Govindan, K., & Golias, M. M. (2013a). Reducing truck emissions at container terminals in a low carbon economy: Proposal of a queueing-based bi-objective model for optimizing truck arrival pattern. Transportation Research Part E: Logistics and Transportation Review, 55, 3-22.

Chen, G., Govindan, K., Yang, Z. Z., Choi, T. M., & Jiang, L. (2013b). Terminal appointment system design by non-stationary M(t)/Ek/c(t) queueing model and genetic algorithm. International Journal of Production Economics, 146(2), 694-703.

Choe, R., Park, T., Oh, M. S., Kang, J., & Ryu, K. R. (2011). Generating a rehandling-free intra-block remarshaling plan for an automated container yard. Journal of Intelligent Manufacturing, 22(2), 201-217.

Covic, F. (2017). Re-marshalling in automated container yards with terminal appointment systems. Flexible Services and Manufacturing Journal, 29(3), 433-503.

Covic, F. (2018). A literature review on container handling in yard blocks. In International Conference on Computational Logistics (pp. 139-167). Springer, Cham.

Covic, F. (2019). Container Handling in Automated Yard Blocks: An Integrative Approach Based on Time Information. Springer. https://doi.org/10.1007/978-3-030-05291-1

Davies, P. (2009). Container Terminal Reservation Systems: Paper presented at the 3rd Annual METRANS National Urban Freight Conference. Long Beach, USA.

de Melo da Silva, M., Erdoğan, G., Battarra, M., & Strusevich, V. (2018). The block retrieval problem. European Journal of Operational Research, 265(3), 931-950.

Dekker, R., Voogd, P., & van Asperen, E. (2006). Advanced methods for container stacking. OR Spectrum,

28(4), 563-586.

De Castillo, B., & Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals. Transportation Research Part B: Methodological, 27(2), 151-166.

Dorndorf, U., & Schneider, F. (2010). Scheduling automated triple cross-over stacking cranes in a container yard. OR Spectrum, 32(3), 617-632.

DP World. INTRODUCING BOXBAY. https://www.dpworld.com/smart-trade/boxbay. Accessed January 12, 2022.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2014). A domain-specific knowledge-based heuristic for the blocks relocation problem. Advanced Engineering Informatics, 28(4), 327-343.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2015). An exact approach for the blocks relocation problem. Expert Systems with Applications, 42(17-18), 6408-6422.

Forster, F., & Bortfeldt, A. (2012). A tree search procedure for the container relocation problem. Computers & Operations Research, 39(2), 299-309.

Fransoo, J. C., & Lee, C. Y. (2013). The critical role of ocean container transport in global supply chain performance. Production and Operations Management, 22(2), 253-268.

Gabrel, V., Murat, C., & Thiele, A. (2014). Recent advances in robust optimization: An overview. European journal of operational research, 235(3), 471-483.

Gaete, M., González-Araya, M. C., González-Ramírez, R. G., & Astudillo, C. (2017). A Novel Storage Space Allocation Policy for Import Containers. International Conference on Operations Research and Enterprise Systems, vol. 884, Springer, 2017, pp. 293–316.

Galle, V. (2018). Optimization models and methods for storage yard operations in maritime container terminals (Doctoral thesis, Massachusetts Institute of Technology, USA). MIT Libraries. Retrieved from https://dspace.mit.edu/handle/1721.1/115592

Galle, V., Boroujeni, S. B., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2016). An average-case asymptotic analysis of the Container Relocation Problem. Operations Research Letters, 44(6), 723-728.

Galle, V., Manshadi, V. H., Boroujeni, S. B., Barnhart, C., & Jaillet, P. (2018a). The stochastic container relocation problem. Transportation Science, 52(5), 1035-1058.

Galle, V., Barnhart, C., & Jaillet, P. (2018b). Yard Crane Scheduling for container storage, retrieval, and relocation. European Journal of Operational Research, 271(1), 288-316.

Gharehgozli, A. H., Roy, D., & De Koster, R. (2016). Sea container terminals: New technologies and OR models. Maritime Economics & Logistics, 18(2), 103-140.

Gharehgozli, A. H., Yu, Y., de Koster, R., & Udding, J. T. (2014). A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal. International Journal of Production Research, 52(9), 2592-2611.

Gharehgozli, A., & Zaerpour, N. (2018). Stacking outbound barge containers in an automated deep-sea terminal. European Journal of Operational Research, 267(3), 977-995.

Gharehgozli, A., Zaerpour, N., & de Koster, R. (2020). Container terminal layout design: transition and future. Maritime Economics & Logistics, 22(4), 610-639.

Giallombardo, G., Moccia, L., Salani, M., & Vacca, I. (2010). Modeling and solving the tactical berth allocation problem. Transportation Research Part B: Methodological, 44(2), 232-245.

Giuliano, G., & O'Brien, T. (2007). Reducing port-related truck emissions: The terminal gate appointment system at the Ports of Los Angeles and Long Beach. Transportation Research Part D: Transport and Environment, 12(7), 460-473.

Guerra-Olivares, R., Smith, N. R., González-Ramírez, R. G., García-Mendoza, E., & Cárdenas-Barrón, L. E. (2018). A heuristic procedure for the outbound container space assignment problem for small and midsize maritime terminals. International Journal of Machine Learning and Cybernetics, 9(10), 1719-1732.

Hakan Akyüz, M., & Lee, C. Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Research Logistics (NRL), 61(2), 101-118.

He, Y., Wang, A., & Su, H. (2019). The impact of incomplete vessel arrival information on container stacking. International Journal of Production Research, 1-15.

Heilig, L., Stahlbock, R., & Voß, S. (2020). From digitalization to data-driven decision making in container terminals. In J. W. Böse (Ed.), *Handbook of Terminal Planning (2nd ed.)*, Operations Research/Computer Science Interfaces Series 64 (pp. 125-154). Springer, Cham.

Hussein, M., & Petering, M. E. (2012). Genetic algorithm-based simulation optimization of stacking algorithms for yard cranes to reduce fuel consumption at seaport container transshipment terminals. In 2012 IEEE Congress on Evolutionary Computation (pp. 1-8). IEEE.

Hu, Q. M., Hu, Z. H., & Du, Y. (2014). Berth and quay-crane allocation problem considering fuel consumption and emissions from vessels. Computers & Industrial Engineering, 70, 1-10.

Hu, Z. H., Sheu, J. B., & Luo, J. X. (2016). Sequencing twin automated stacking cranes in a block at automated container terminal. Transportation Research Part C: Emerging Technologies, 69, 208-227.

Iris, Ç., Pacino, D., Ropke, S., & Larsen, A. (2015). Integrated berth allocation and quay crane assignment problem: Set partitioning models and computational results. Transportation Research Part E: Logistics and Transportation Review, 81, 75-97.

Iris, C. (2016). Exact and Heuristic Methods for Integrated Container Terminal Problems (Doctoral thesis, Technical University of Denmark, Denmark). DTU orbit. Retrieved from https://orbit.dtu.dk/en/publications/exact-and-heuristic-methods-for-integrated-container-terminal-pro

Iris, Ç., Pacino, D., & Ropke, S. (2017). Improved formulations and an adaptive large neighborhood search heuristic for the integrated berth allocation and quay crane assignment problem. Transportation Research Part E: Logistics and Transportation Review, 105, 123-147.

Iris, Ç., & Lam, J. S. L. (2019). Recoverable robustness in weekly berth and quay crane planning. Transportation Research Part B: Methodological, 122, 365-389.

Jang, D. W., Kim, S. W., & Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations. International Journal of Production Economics, 143(2), 256-262.

Jiang, X., Chew, E. P., & Lee, L. H. (2015). Innovative container terminals to improve global container transport chains. In C.-Y. Lee & Q. Meng (Eds.), *Handbook of Ocean Container Transport Logistics*, International Series in Operations Research & Management Science 220 (pp. 3-41). Springer, Cham.

Jiang, X., Lee, L. H., Chew, E. P., Han, Y., & Tan, K. C. (2012). A container yard storage strategy for improving land utilization and operation efficiency in a transshipment hub port. European journal of operational research, 221(1), 64-73.

Jiang, X., Chew, E. P., Lee, L. H., & Tan, K. C. (2013). Flexible space-sharing strategy for storage yard management in a transshipment hub port. OR Spectrum, 35(2), 417-439.

Jiang, T., Zeng, B., Wang, Y., & Yan, W. (2021). A New Heuristic Reinforcement Learning for Container Relocation Problem. In Journal of Physics: Conference Series (Vol. 1873, No. 1, p. 012050). IOP Publishing.

Jin, J. G., Lee, D. H., & Cao, J. X. (2016). Storage yard management in maritime container terminals. Transportation Science, 50(4), 1300-1313.

Jin, B., Zhu, W., & Lim, A. (2015). Solving the container relocation problem by an improved greedy look-ahead heuristic. European Journal of Operational Research, 240(3), 837-847.

Jovanovic, R., Tuba, M., & Voß, S. (2019). An efficient ant colony optimization algorithm for the blocks relocation problem. European Journal of Operational Research, 274(1), 78-90.

Jovanovic, R., & Voß, S. (2014). A chain heuristic for the blocks relocation problem. Computers & Industrial Engineering, 75, 79-86.

Kemme, N. (2020). State-of-the-Art Yard Crane Scheduling and Stacking. In J. W. Böse (Ed.), *Handbook of Terminal Planning (2nd ed.)*, Operations Research/Computer Science Interfaces Series 64 (pp. 383-413). Springer, Cham.

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. Computers & Industrial Engineering, 32(4), 701-711.

Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 124(1), 89-101.

Kim, K. H., & Hong, G. P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research, 33(4), 940-954.

Kim, K. H., & Lee, H. (2015). Container terminal operation: current trends and future challenges. In C.-Y. Lee & Q. Meng (Eds.), *Handbook of Ocean Container Transport Logistics – Making Global Supply Chain Effective* (pp. 43-73). Springer, Cham.

Kizilay, D., & Eliiyi, D. T. (2020). A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals. Flexible Services and Manufacturing Journal, 1-42.

Knatz, G. (2017). How competition is driving change in port governance, strategic decision-making and government policy in the United States. Research in Transportation Business & Management, 22, 67-77.

Ku, L. P., Chew, E. P., Lee, L. H., & Tan, K. C. (2012). A novel approach to yard planning under vessel arrival uncertainty. Flexible Services and Manufacturing Journal, 24(3), 274-293.

Ku, D., & Arthanari, T. S. (2016a). Container relocation problem with time windows for container departure. European Journal of Operational Research, 252(3), 1031-1039.

Ku, D., & Arthanari, T. S. (2016b). On the abstraction method for the container relocation problem. Computers & Operations Research, 68, 110-122.

Lee, B. K., & Kim, K. H. (2010). Optimizing the block size in container yards. Transportation Research Part E: Logistics and Transportation Review, 46(1), 120-135.

Lee, Y., & Lee, Y. J. (2010). A heuristic for retrieving containers from a yard. Computers & Operations Research, 37(6), 1139-1147.

Lee, C. Y., & Song, D. P. (2017). Ocean container transport in global supply chains: Overview and research opportunities. Transportation Research Part B: Methodological, 95, 442-474.

Lee, W. S., Ottjes, J. A., Veeke, H. P., & Rijsenbrij, J. C. (2008). Using container call time information for restacking reduction. In Proc. 6th International Industrial Simulation Conference, pp. 293- 298.

Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. European Journal of Operational Research, 239(2), 297-312.

Li, Y., Zhu, X., Wang, L., & Chen, X. (2017). Stowage plan based slot optimal allocation in rail-water container terminal. Journal of Control Science and Engineering, 2017.

Liebchen, C., Lübbecke, M., Möhring, R., & Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja et al. (Eds.), *Robust and online large-scale optimization* (pp. 1-27). Springer, Berlin, Heidelberg.

Lin, D. Y., & Chiang, C. W. (2017). The storage space allocation problem at a container terminal. Maritime Policy & Management, 44(6), 685-704.

Lin, D. Y., Lee, Y. J., & Lee, Y. (2015). The container retrieval problem with respect to relocation. Transportation Research Part C: Emerging Technologies, 52, 132-143.

Lloyd's List (2020). One Hundred Container Ports 2020. Retrieved from https://lloydslist.maritimeintelligence.informa.com/one-hundred-container-ports-2020. Accessed July 28, 2021.

López-Plata, I., Expósito-Izquierdo, C., Lalla-Ruiz, E., Melián-Batista, B., & Moreno-Vega, J. M. (2017). Minimizing the Waiting Times of block retrieval operations in stacking facilities. Computers & Industrial Engineering, 103, 70-84.

Lu, Z. Q., & Xi, L. F. (2010). A proactive approach for simultaneous berth and quay crane scheduling problem with stochastic arrival and handling time. European Journal of Operational Research, 207(3), 1327-1340.

Luo, J., Wu, Y., & Mendes, A. B. (2016). Modelling of integrated vehicle scheduling and container storage problems in unloading process at an automated container terminal. Computers & Industrial Engineering, 94, 32-44.

Luo, J., Wu, Y., Halldorsson, A., & Song, X. (2011). Storage and stacking logistics problems in container terminals. OR insight, 24(4), 256-275.

Maglić, L., Gulić, M., & Maglić, L. (2020). Optimization of container relocation operations in port container terminals. Transport, 35(1), 37-47.

Maldonado, S., González-Ramírez, R. G., Quijada, F., & Ramírez-Nafarrate, A. (2019). Analytics meets port logistics: A decision support system for container stacking operations. Decision Support Systems, 121, 84-93.

Meisel, F. (2009). Seaside operations planning in container terminals. Physica-Verlag. https://doi-org.liverpool.idm.oclc.org/10.1007/978-3-7908-2191-8

Monaco, M. F., Sammarra, M., & Sorrentino, G. (2014). The terminal-oriented ship stowage planning problem. European Journal of Operational Research, 239(1), 256-265.

Mongelluzzo, B. (2015a). Free-flow program finds success in Los Angeles port. JOC.com. May 22, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-los-angeles/free-flow-program-finds-success-los-angeles-port_20150522.html. Accessed January 13, 2021.

Mongelluzzo, B. (2015b). Free-flow container operation launched in Port of Los Angeles. JOC.com. Feb 27, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-los-angeles/initiative-aimed-reducing-los-angeles-port-congestion-launched_20150227.html. Accessed January 13, 2021.

Mongelluzzo, B. (2021). LA-LB preparing for record 20 million TEU this year. JOC. com. Sep 03, 2021. Retrieved from https://www.joc.com/port-news/us-ports/la-lb-preparing-record-20-million-teu-year_20210903.html?utm_source=Eloqua&utm_medium=email&utm_campaign=CL_JOC%20Daily%209%2F7%2F21%20_PC00000_e-production_E-111179_TF_0907_0617. Accessed September 7, 2021.

Ng, W. C., & Mak, K. L. (2005). Yard crane scheduling in port container terminals. Applied Mathematical Modelling, 29(3), 263-276.

Notteboom, T., Pallis, T., & Rodrigue, J. P. (2021). Disruptions and resilience in global container shipping and ports: the COVID-19 pandemic versus the 2008–2009 financial crisis. Maritime Economics & Logistics, 23(2), 179-210.

Pant, R., Barker, K., Ramirez-Marquez, J. E., & Rocco, C. M. (2014). Stochastic measures of resilience and their application to container terminals. Computers & Industrial Engineering, 70, 183-194.

Park, T., Choe, R., Kim, Y. H., & Ryu, K. R. (2011). Dynamic adjustment of container stacking policy in an automated container terminal. International Journal of Production Economics, 133(1), 385-392.

Parker, B. (2015). Reduce Port Congestion By Expanding Container 'Free-Flow' Systems. Retrieved from https://www.supplychainbrain.com/articles/22934-reduce-port-congestion-by-expanding-container-free-flow-systems. Accessed January 14, 2021.

Petering, M. E., & Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. European Journal of Operational Research, 231(1), 120-130.

Petering, M. E., Wu, Y., Li, W., Goh, M., de Souza, R., & Murty, K. G. (2017). Real-time container storage location assignment at a seaport container transshipment terminal: dispersion levels, yard templates, and sensitivity analyzes. Flexible Services and Manufacturing Journal, 29(3-4), 369-402.

Phan, M. H., & Kim, K. H. (2016). Collaborative truck scheduling and appointments for trucking companies and container terminals. Transportation Research Part B: Methodological, 86, 37-50.

Porter J. (2020). LA/Long Beach anchorages full as cargo surge clogs Californian supply chains. lloydsloadinglist.com. December 30, 2020. Retrieved from https://www.lloydsloadinglist.com/freight-directory/news/LALong-Beach-anchorages-full-as-cargo-surge-clogs-Californian-supply-chains/78163.htm#.YTEFFo70mUl. Accessed September 2, 2021.

Preston, P., & Kozan, E. (2001). An approach to determine storage locations of containers at seaport terminals. Computers & Operations Research, 28(10), 983-995.

Quispe, K. E. Y., Lintzmayer, C. N., & Xavier, E. C. (2018). An exact algorithm for the blocks relocation problem with new lower bounds. Computers & Operations Research, 99, 206-217.

Razouk, C., Benadada, Y., & Boukachour, J. (2016). New approaches for solving the container stacking problem. In 2016 3rd International Conference on Logistics Operations Management (GOL) (pp. 1-7). IEEE.

Rekik, I., Elkosantini, S., & Chabchoub, H. (2015). Container stacking problem: a literature review. In: International Conference on Computers and Industrial Engineering (CIE45, 2015) Proceedings (pp, 1-10), Metz/France.

Rekik, I., Elkosantini, S., & Chabchoub, H. (2016). Toward a Knowledge Based Multi-agent Architecture for the Reactive Container Stacking in Seaport Terminals. In International Conference on Artificial Intelligence and Soft Computing (pp. 718-728). Springer, Cham.

Rekik, I., Elkosantini, S., & Chabchoub, H. (2018). A case based heuristic for container stacking in seaport terminals. Advanced Engineering Informatics, 38, 658-669.

Rekik, I., & Elkosantini, S. (2019). A multi agent system for the online container stacking in seaport terminals. Journal of Computational Science, 35, 12-24.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research, 31(4), 563-591.

Rendl, A., & Prandtstetter, M. (2013). Constraint models for the container pre-marshaling problem. In: G. Katsirelos & C. Quimper (Eds.), Modref 2013: 12th International Workshop on Constraint Modelling and Reformulation, pp 44–56.

Ries, J., Gonz ález-Ram írez, R. G., & Miranda, P. (2014). A fuzzy logic model for the container stacking

problem at container terminals. In R.G. Gonz âlez-Ram írez et al. (Eds.), *International Conference on Computational Logistics* (ICCL) 2014 (pp. 93-111), Springer.

Sauri, S., & Martin, E. (2011). Space allocating strategies for improving import yard performance at marine terminals. Transportation Research Part E: Logistics and Transportation Review, 47(6), 1038-1057.

Smith, D. R. (1985). The design of divide and conquer algorithms. Science of Computer Programming, 5, 37-58.

Song, D. P. (2021a). Container Logistics and Maritime Transport. Routledge. https://doi-org.liverpool.idm.oclc.org/10.4324/9780429320996

Song, D. (2021b). A Literature Review, Container Shipping Supply Chain: Planning Problems and Research Opportunities. Logistics, 5(2), 41.

Speer, U., & Fischer, K. (2017). Scheduling of different automated yard crane systems at container terminals. Transportation Science, 51(1), 305-324.

Stahlbock, R., & Voß, S. (2008a). Vehicle routing problems and container terminal operations - un update of research, in B. Golden et al. (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (pp. 551–589), Springer, Boston, MA.

Stahlbock, R., & Voß, S. (2008b). Operations research at container terminals: a literature update. OR Spectrum, 30(1), 1-52.

Steenken, D., Voß, S., & Stahlbock, R. (2004). Container terminal operation and operations research-a classification and literature review. OR Spectrum, 26(1), 3-49.

Tanaka, S., & Mizuno, F. (2018). An exact algorithm for the unrestricted block relocation problem. Computers & Operations Research, 95, 12-31.

Tanaka, S., & Takii, K. (2016). A faster branch-and-bound algorithm for the block relocation problem. IEEE Transactions on Automation Science and Engineering, 13(1), 181-190.

Tang, L., Jiang, W., Liu, J., & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions, 47(7), 751-766.

THE MARITIME EXECUTIVE (2021). Shanghai Targets 47 Million TEU Annually Through Growth and Automation. maritime-executive.com. JUL 9, 2021. Retrieved from https://www.maritime-executive.com/article/shanghai-targets-47-million-teu-annually-through-growth-and-automation. Accessed September 2, 2021.

THE MARITIME EXECUTIVE (2021). Giant Automated Container Rack Proves Successful in DP World Trial. maritime-executive.com. AUG 18, 2021. Retrieved from https://www.maritime-executive.com/article/giant-automated-container-rack-proves-successful-in-dp-world-trial. Accessed January 12, 2022.

Tierney, K., & Voß, S. (2016). Solving the robust container pre-marshalling problem. In A. Paias et al. (Eds.), *International Conference on Computational Logistics* (ICCL) 2016 (pp. 131-145), Springer.

Tricoire, F., Scagnetti, J., & Beham, A. (2018). New insights on the block relocation problem. Computers & Operations Research, 89, 127-139.

Todd, S., & Waters, W. (2020). UK port congestion could continue for months. lloydsloadinglist.com. December 10, 2020. Retrieved from https://www.lloydsloadinglist.com/freight-directory/news/UK-port-congestion-could-continue-for-months/78050.htm#.YTDz_o70mUk. Accessed September 2, 2021.

Tong, X., Woo, Y. J., Jang, D. W., & Kim, K. H. (2015). Heuristic Rules Based on a Probabilistic Model and a Genetic Algorithm for Relocating Inbound Containers with Uncertain Pickup Times. International Journal of Industrial Engineering, 22, 93–101.

Torkjazi, M., Huynh, N., & Shiri, S. (2018). Truck appointment systems considering impact to drayage truck tours. Transportation Research Part E: Logistics and Transportation Review, 116, 208-228.

UNCTAD (2019). Review of Maritime Transport 2019. United Nations Conference on Trade and Development, United Nations publication. Sales No. E.19.II.D.20.

UNCTAD (2020). Review of Maritime Transport 2020. United Nations Conference on Trade and Development, United Nations publication. Sales No. E.20.II.D.31.

UNCTADstat, Container port throughput, annual. Retrieved from https://unctadstat.unctad.org/wds/TableViewer/tableView.aspx?ReportId=13321. Accessed August 4, 2021.

Ünlüyurt, T., & Aydın, C. (2012). Improved rehandling strategies for the container retrieval process. Journal of Advanced Transportation, 46(4), 378-393.

Vis, I. F., & Carlo, H. J. (2010). Sequencing two cooperating automated stacking cranes in a container terminal. Transportation Science, 44(2), 169-182.

Vis, I. F., & de Koster, R. (2003). Transshipment of containers at a container terminal: An overview. European journal of operational research, 147(1), 1-16.

Voß, S., & Schwarze, S. (2019). A note on alternative objectives for the blocks relocation problem. In C. Paternina-Arboleda and S. Voß (Eds.), *Computational Logistics*, ICCL 2019, LNCS 11756 (pp. 101–121). Springer, Cham.

Wan, Y. W., Liu, J., & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack. Naval Research Logistics (NRL), 56(8), 699-713.

Wang, L., Zhu, X., & Xie, Z. (2020). Efficient container stacking approach to improve handling: efficiency in Chinese rail–truck transshipment terminals. Simulation, 96(1), 3-15.

Wei, C., Gao, W. W., Hu, Z. H., Yin, Y. Q., & Pan, S. D. (2019). Assigning customer-dependent travel time limits to routes in a cold-chain inventory routing problem. Computers & Industrial Engineering, 133, 275-291.

Wiese, J., Suhl, L., & Kliewer, N. (2011). Planning container terminal layouts considering equipment types and storage block design. In J. W. Böse (Ed.), *Handbook of Terminal Planning*, Operations research/computer science interfaces series 49 (pp. 219-245). Springer, New York.

Wu, K.C., & Ting, C.J. (2010). A beam search algorithm for minimizing reshuffle operations at container yards. In: Proceedings of the 2010 International Conference on Logistics and Maritime Systems. Busan, Korea.

Wu, K. C., Ting, C. J., & HERNÁNDEZ, R. (2010). Appling tabu search for minimizing reshuffle operations at container yards. Journal of the Eastern Asia Society for Transportation Studies, 8, 2379-2393.

Xiang, X., Liu, C., & Miao, L. (2017). A bi-objective robust model for berth allocation scheduling under uncertainty. Transportation Research Part E: Logistics and Transportation Review, 106, 294-319.

Yu, M., & Qi, X. (2013). Storage space allocation models for inbound containers in an automatic container terminal. European Journal of Operational Research, 226(1), 32-45.

Yu, M., Liang, Z., Teng, Y., Zhang, Z., & Cong, X. (2021). The inbound container space allocation in the automated container terminals. Expert Systems with Applications, 179, 115014.

Zaerpour, N., Gharehgozli, A., & De Koster, R. (2019). Vertical expansion: A solution for future container terminals. Transportation Science, 53(5), 1235-1251.

Zeng, Q., Feng, Y., & Chen, Z. (2017). Optimizing berth allocation and storage space in direct transshipment operations at container terminals. Maritime Economics & Logistics, 19(3), 474-503.

Zeng, Q., Feng, Y., & Yang, Z. (2019). Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering, 127, 366-382.

Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. European Journal of Operational Research, 245(2), 415-422.

Zehendner, E., & Feillet, D. (2014). A branch and price approach for the container relocation problem. International Journal of Production Research, 52(24), 7159-7176.

Zehendner, E., Feillet, D., & Jaillet, P. (2017). An algorithm with performance guarantee for the online container relocation problem. European Journal of Operational Research, 259(1), 48-62.

Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. Transportation Research Part B: Methodological, 37(10), 883-903.

Zhang, C., Chen, W., Shi, L., & Zheng, L. (2010). A note on deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 205(2), 483-485.

Zhang, C., Guan, H., Yuan, Y., Chen, W., & Wu, T. (2020). Machine learning-driven algorithms for the container relocation problem. Transportation Research Part B: Methodological, 139, 102-131.

Zhang, C., Wu, T., Kim, K. H., & Miao, L. (2014). Conservative allocation models for outbound containers in container terminals. European Journal of Operational Research, 238(1), 155-165.

Zhang, R., Liu, S., & Kopfer, H. (2016). Tree search procedures for the blocks relocation problem with batch moves. Flexible Services and Manufacturing Journal, 28(3), 397-424.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. Transportation Research Part E: Logistics and Transportation Review, 46(3), 327-343.

Zhen, L., Jiang, X., Lee, L. H., & Chew, E. P. (2013). A review on yard management in container terminals. Industrial Engineering and Management Systems, 12(4), 289-304.

Zhen, L. (2014). Container yard template planning under uncertain maritime market. Transportation Research Part E: Logistics and Transportation Review, 69, 199-217.

Zhen, L. (2015). Tactical berth allocation under uncertainty. European Journal of Operational Research, 247(3), 928-944.

Zhu, H., Ji, M., & Guo, W. (2020). Two-stage search algorithm for the inbound container unloading and stacking problem. Applied Mathematical Modelling, 77, 1000-1024.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. IEEE Transactions on Automation Science and Engineering, 9(4), 710-722.

Zweers, B. G., Bhulai, S., & van der Mei, R. D. (2020). Optimizing pre-processing and relocation moves in the stochastic container relocation problem. European Journal of Operational Research, 283(3), 954-971.

Chapter 2

The stochastic container relocation problem with flexible service policies

Abstract: This paper investigates the Stochastic Container Relocation Problem in which a flexible service policy is adopted in the import container retrieval process. The flexible policy allows the terminal operators to determine the container retrieval sequence to some extent, which provides more opportunity for reducing the number of relocations and the truck waiting times. A more general probabilistic model that captures customers' arrival preference is presented to describe the randomness for external truck arrivals within their appointed time windows. Being a multi-stage stochastic sequential decision-making problem, it is first formulated into a stochastic dynamic programming (SDP) model to minimise the expected number of relocations. Then, the SDP model is extended considering a secondary objective representing the truck waiting times. Tree search-based algorithms are adapted to solve the two models to their optimality. Heuristic algorithms are designed to seek high-quality solutions efficiently for larger problems. A discrete-event simulation model is developed to evaluate the optimal solutions and the heuristic solutions respectively on two performance metrics. Extensive computational experiments are performed based on instances from literature to verify the effectiveness of the proposed models and algorithms. [1]
Keywords: stochastic container relocation problem, appointment time window, flexible service, stochastic dynamic programming, tree search-based algorithm

2.1 Introduction

As critical nodes in the global container transport networks, container terminals play an important role in transhipping containerized cargoes between different transport modes. At container terminals, containers are handled through a series of operations, which can be generally divided into seaside operations (unloading/loading operations) and landside operations (stacking/retrieval operations). Methods for improving the operational efficiencies at container terminals have been studied for years and many models and algorithms have been developed (see review papers: Stahlbock and Voß, 2008; Zhen et al., 2013; Lehnfeld and Knust, 2014; Carlo et al., 2014; Lee and Song, 2017; Dragović et al., 2017).

[1] Feng, Y., Song, D. P., Li, D., & Zeng, Q. (2020). The stochastic container relocation problem with flexible service policies. Transportation Research Part B: Methodological, 141, 116-163, published.

One major source of inefficiency in most container terminals is the relocation move (Caserta et al., 2011a; Ku and Arthanari, 2016a). In a typical container terminal, containers are stored in the terminal yard after their arrivals, waiting for onward transport. The storage area of a yard is divided into blocks, each including 20-40 bays with each bay consisting of several stacks. Containers are often piled up vertically in stacks. During the container retrieval process, if the target container to be retrieved is not on the topmost tier, those above it – that is, the *blocking* containers - need to be moved to other stacks in the same bay. The move of blocking containers is called *relocation*, *reshuffling*, or *rehandling*, which is an unproductive operation. The Container Relocation Problem (CRP) aims at seeking a sequence of moves to retrieve all containers from a given bay with the minimum number of relocations, which is a combinatorial optimisation problem. Most existing studies on the CRP assume a priori given container retrieval sequence. The CRP for import containers whose retrieval times are subject to uncertainty has been less investigated.

For import containers, the stochastic arrival times of external trucks complicates the CRP as it will easily result in re-relocations in the future. The Truck Appointment System (TAS), also known as vehicle booking system (VBS), can increase the predictability for truck arrival times, which has been implemented in many container ports to control the truck arrivals at the terminal (Davies, 2009). Under TAS, a truck must make an appointment with the terminal in advance to indicate a time window in which the truck will arrive at the terminal. Therefore, each arrival truck will have an appointed arrival time window. As a result, the arrival precedence of trucks with different appointed time windows becomes pre-specified. However, the arrival sequence of the trucks within the same appointed time window remains uncertain, which is typically revealed during the retrieval process. The CRP that considers the randomness truck arrivals in the same time window is termed as the Stochastic Container Relocation Problem (SCRP) or the CRP with Time Windows (CRPTW) in the literature.

The SCRP is more realistic to model the import container retrieval process. Among the very few studies on the SCRP (e.g. Ku and Arthanari, 2016a; Galle et al., 2018b), a common assumption is that retrieval requests are fulfilled on a first-come-first-served (FCFS) basis. The FCFS rule appears to be reasonable in practice to ensure service equity but may lead to a sub-optimal solution from the overall system perspective. Besides, the service equity of the FCFS rule is debatable, because the trucks may experience different waiting times and the required number of relocations may be affected by previous trucks. Truck waiting time is part of the truck turn time, which is a key performance metric to measure the efficiency of a container terminal and also contributes to the evaluation of customer service levels and port competitiveness (de Melo da Silva et al., 2018). Some ports (e.g., Port Botany; Port Metro Vancouver) are even charged for a penalty if they exceed a stipulated turn time. As an alternative to the FCFS service, a flexible service may yield more opportunities for optimisation on the number of relocations, as well as the truck waiting time. In this paper, we extend the SCRP to a general setting that allows some flexibility in the container retrieval sequences. We term this type of problem as the *SCRP with flexible service policies* or *SCRP-FS*.

With the SCRP-FS, we also generalize the probabilistic model of truck arrivals. In the existing studies, the arrival order of trucks booked in the same time window is assumed to be uniformly distributed, which is not necessarily realistic. Customers (truckers) may have different preferences for different segments of their appointed time windows. Firstly, in the TAS, the trucks may not always get their desired time windows because slots are often oversubscribed (Mongelluzzo, 2019). In order to narrow the deviation from its desired time window, the truck will have preference for either the earlier segment or the latter segment of the shifted appointed time window. Secondly, some terminal operators (e.g., DP World, Patrick) impose financial penalties on no-show (or late) trucks to ensure truckers compliance with their appointed time

windows. To avoid such penalty, those trucks that are subject to high uncertainties on the road tend to target the earlier segment of the appointed time window. Such customer preference information may come from the TAS or from historical data that record the trucks' arrival behaviour, which can be utilised to make better decisions.

This paper aims to investigate how to use flexible service policies to improve import container relocation and retrieval in the presence of truck arrival uncertainties characterised by customers' preference. The objective of this paper is: (i) to seek the optimal solution (including retrieval sequence and relocation positions) that retrieves all containers from a given bay considering both terminal relocation and external truck waiting; (ii) to quantify the reduction on the number of relocations and the truck waiting times by adopting the flexible service policy; (iii) to evaluate the impact of different bay layouts (i.e. size and fill rate) and truck arrival patterns (i.e., the number of trucks booked in a time window and the customer preference) on the effectiveness of the flexible policy compared with the FCFS rule; (iv) and to analyse the influence of customer preference on the results.

Our contributions to the existing literature and practice can be summarized as follows: (i) We propose a new service policy to improve import container retrieval performance in the context of stochastic container relocation problem (SCRP) and generalize the SCRP to be SCRP-FS, which can provide more opportunities for optimising the current retrieval practice. We also generalize the probabilistic model of truck arrivals so that the customers' preference-based arrival behaviour can be captured more accurately. (ii) We introduce a new optimisation framework that jointly optimises the expected number of relocations and the truck waiting times. The proposed model enables port operators to quantify the benefits of controlling the truck service sequence to both terminals and truckers, which is the first in the SCRP studies. (iii) We develop exact algorithms by extending an existing algorithm with major adaptions. The incremental contributions of our exact algorithms include a decision tree with a new structure, a new lower bound for the expected number of relocations for the SCRP-FS, and a procedure for minimising the truck waiting times batch by batch. We develop two efficient heuristic algorithms for solving the SCRP-FS, which can serve as decision support tools for terminal operators in real applications. (iv) We construct a discrete-event simulation model for evaluating the exact solutions in terms of two performance measures: the expected number of relocations and the truck waiting time. To the best of our knowledge, this study is among the first to evaluate the truck waiting time of the exact solutions with a tree structure in the context of uncertain CRP. (v) We conduct extensive experiments to demonstrate the effectiveness of the flexible service policy and the influence of the customer preference on the results. The results can provide managerial insights for terminal operators to manage import container operations more efficiently.

The remainder of the paper is organized as follows. Section 2.2 reviews the previous work related to the CRP and SCRP and discusses the service policies applied in the (S)CRP. Section 2.3 describes the problem in detail and formulate it using stochastic dynamic programming. Section 2.4 and section 2.5 respectively propose exact solution algorithms and heuristic solution methods. A simulation model is developed in Section 2.6 to evaluate the solutions. The results of computational experiments are reported in Section 2.7. Section 2.8 summarizes the findings and provides managerial insights.

2.2 Literature review

Container relocation is related to several container handling processes at container terminals. Four types of relevant problems have been identified by Carlo et al. (2014): storage location assignment problem, joint retrieval sequencing and relocation problem, pre-marshalling problem and re-marshalling problem. An

overview of these problems from a mathematical perspective is given in Lehnfeld and Knust (2014). The joint retrieval sequencing and relocation problem is the focus of this paper.

2.2.1 Deterministic CRP and uncertain CRP

Most studies on the joint retrieval sequencing and relocation problem assume a priori given retrieval sequence and only focus on optimising relocation positions, which leads to the standard CRP. The basic objective of the standard CRP is to retrieve all containers in a given bay in a pre-defined order with the minimum number of relocations. There are also several variants of the CRP, such as the dynamic CRP, the unrestricted CRP, etc. We classify the relevant literature into two research streams: *deterministic CRP* and *uncertain* CRP, which are differentiated by whether the information on the containers' retrieval times/sequences is deterministic or uncertain.

Previous researches have largely concentrated on the first research stream, i.e. the certain version of the CRP. A number of (mixed) integer programming have been proposed to solve the problem (e.g. Wan et al., 2009; Caserta et al., 2012; Petering and Hussein, 2013; Tang et al., 2015; Zehendner et al., 2015; Expósito-Izquierdo et al., 2015; Galle et al., 2018a). Other studies focus on developing effective solution algorithms. Exact solution algorithms are mainly by search-based algorithms, e.g., (iterative deepening) A* algorithms (Zhu et al., 2012; Borjian et al., 2015a; Quispe et al., 2018), Branch and Bound (B&B) (Kim and Hong, 2006; Ünlüyurt and Aydın, 2012; Expósito-Izquierdo et al., 2015; Tanaka and Takii, 2016), Branch and Price (Zehendner and Feillet, 2014), and the abstraction method (Ku and Arthanari, 2016b), Besides, heuristics algorithms are presented to overcome computational complexities of the CRP, e.g., beam search algorithms (Bacci et al., 2019; Ting and Wu, 2017) and greedy heuristics (Jin et al., 2015; Jovanovic and Voß, 2014) (c.f.Ku and Arthanari, 2016b and the references therein).

In the second research stream, the uncertain CRP may be further categorised into two sub-groups: the online setting and the probabilistic setting, according to whether the uncertainties of the containers' retrieval times/sequences are represented by probabilities or not.

In the online setting, the knowledge of the exact container retrieval sequence is limited to a given look-ahead horizon and is revealed over time gradually, and the research focus is to design efficient online heuristics to relocate containers in real-time. Zehendner et al. (2017) investigate a case of the online CRP where the look-ahead horizon is zero and one container is revealed at a time. They analyse the theoretical performance of an online relocation rule called heuristic *L* (leveling) that relocates containers to the lowest tier. Zhao and Goodchild (2010) make use of truck appointment information to deal with the online CRP using a simulation method. At the beginning of the retrieval process, the retrieval containers booked in different time windows are known, but the exact retrieval sequence of the containers booked in the same time windows is unknown or partially known, which is revealed as the retrieval proceeds periodically. Two heuristics are designed to reduce the number of relocations utilising the truck arrival information.

In the probabilistic setting, the uncertainties on the containers' retrieval times/sequences are modelled by a probability distribution and the research purpose is to minimise the expectation of the performance measures. Given the probabilistic distribution of containers dwell times, Tong et al. (2015) propose two heuristic rules to determine the positions of relocated containers with the objective of minimising the total expected number of relocations for retrieving all the containers from a bay. Considering groups of containers with uncertain group retrieval orders, de Melo da Silva et al. (2018) introduce the Block Retrieval Problem (BRP) and the Bi-objective Block Retrieval Problem (2BRTP). The BRP aims to minimise the number of relocations for the initial target group by optimising the retrieval sequence and the relocation positions, which is solved by a B&B algorithm and a linear time algorithm. Then, assuming that

the probability of any remaining group being the forthcoming one is known, the 2BRTP is introduced with the primary objective of minimising the number of relocations for the initial target group and the secondary objective of minimising the expected number of relocations for the forthcoming group. A B&B algorithm and a beam search algorithm are proposed for solving the 2BRTP.

The above studies of the uncertain CRP focus on the solution algorithms without providing the details of problem formulation. Mathematical optimisation models for the uncertain CRP in the probabilistic setting are presented in a few studies. Borjian et al. (2013) introduce a two-stage stochastic optimisation framework for the CRP with partial information, in which the departure times of some containers are known, while for the remaining containers only a probability distribution of the retrieval order is given. The model is to minimise the weighted sum of the expected number of relocations and total retrieval delays, which yields the optimal sequence of moves for each possible scenario. A heuristic based on the stochastic optimisation model is designed to obtain sub-optimal solutions. Ku and Arthanari (2016a) propose the CRP with Time Windows (CRPTW), in which the retrieval sequences of containers with different departure time windows are in ascending order by their departure time windows. It is assumed that containers with the same departure time windows are retrieved in the uniformly distributed order which is revealed one container at a time. The problem is formulated into a stochastic dynamic programming (SDP) model to minimise the expected number of relocations. A search-based algorithm (depth-first search) in a tree space is proposed to solve the model optimally. More recently, Galle et al. (2018b) study a similar CRPTW using the term SCRP. Different from Ku and Arthanari (2016a), the full exact retrieval order of containers booked in the same time window is revealed at once after all containers booked in the previous time window have been retrieved. The SCRP is formulated as a multi-stage stochastic model, called the batch model. An optimal search-based algorithm called Pruning-Best-First-Search (PBFS), a randomized approximate algorithm called PBFSA, and two new heuristics are proposed to solve the batch model. The batch model is compared with the SDP model in Ku and Arthanari (2016a) both theoretically and computationally to prove that it is beneficial to use the batch model in terms of the expected number of relocations. Note that because of the use of the same information revealing mechanism and the similar modelling techniques to seek global optimal solutions, the current paper positions itself to the SCRP proposed by Galle et al. (2018b). However, our study differs from Galle et al. (2018b) in many aspects that will be elaborated at the end of this section.

Finally, it is worth mentioning that several interesting variants of the (S)CRP have also been investigated. For instance, a few studies consider the CRP in a three-dimensional storage area and take into account both the number of container movements and the working time of the yard crane (e.g., Lee and Lee, 2010; Lin et al., 2015). From the service-oriented standpoint, López-Plata et al. (2017) address the Blocks Relocation Problem with Waiting Times (BRP-WT) focusing on minimising the sum of waiting times of a set of blocks during retrieval. Recently, Zweers et al. (2020) present a new optimisation problem related to the SCRP, called the Stochastic Container Relocation Problem with Pre-Processing (SCRPPP), which aims to minimise a weighted average of the pre-processing moves in the pre-processing phase (when the crane is idle) and the relocation moves in the relocation phase. A B&B algorithm and a local search heuristic are proposed to solve the problem.

2.2.2 Service policies

Among the literature reviewed above, most of them assume that each container has a distinct retrieval sequence that is exogenously determined. For import containers, this assumption corresponds to the FCFS service policy, under which the containers are retrieved in the order of external truck arrivals. Using the

FCFS service policy may make the problem more tractable but meanwhile loses some opportunities for optimisation. In the uncertain CRP context, three studies have considered flexible service policies or service out-of-order (SOOO). Zhao and Goodchild (2010) propose a pickup sequence dictation heuristic that dictates the retrieval sequence of containers within the first arrival group to reduce relocations. In the study of de Melo da Silva et al. (2018), the containers in the same group are assigned the same retrieval priority and their retrieval order is to be optimised. Borjian et al. (2013) assume a service time window for each container instead of imposing strict service order on containers. In the deterministic CRP context, a few researchers have also introduced the flexibility of container retrieval sequence, e.g., Kim and Hong (2006), Borjian et al. (2015b) and Zeng et al. (2019).

One major concern of flexible retrievals is the possibility of causing extra delay and service inequity to some trucks. Zhao and Goodchild (2010) and Zeng et al. (2019) attempt to avoid this by restricting out-of-order retrievals within a group of containers booked in the same time window. However, Zhao and Goodchild (2010) do not analyse the impact of dictating the pickup sequence on external trucks. Zeng et al. (2019) find that when the number of containers booked in each time window is over a certain number, adjusting the pickup sequence may increase the average waiting time of external trucks. One possible reason for this is that they do not consider the common phenomenon of trucks queuing before getting the retrieval service at congested container terminals, where out-of-order retrievals may not create more waiting times. Borjian et al. (2015b) control the level of flexibility by limiting the number of out-of-order retrievals before each truck. They conclude that the average retrieval delay is decreased as a result of out-of-order retrievals, and in the long term, the service equity that each truck receives is not adversely affected. In the study of Kim and Hong (2006), the containers in the same group are assumed to be loaded into a cluster of slots of a vessel in any order, which means those containers share the same retrieval priority, and thus there is no issue of delay and service inequity. Similarly, de Melo da Silva et al. (2018) do not consider this issue either, as they assume that the containers in the same group are to be retrieved by the same customer. Borjian et al. (2013) set a maximum service delay for each container and consider a weighted objective function that jointly minimises the expected number of relocations and total delays. It can be seen that Borjian et al (2013) is the only paper that applies flexible retrievals in an uncertain CRP using a mathematical optimisation model. However, in their model, all uncertain information is revealed at once, which is not close to reality.

In this paper, out-of-order retrievals are limited to the trucks arriving in the same sub-time window to maintain the service equity among subsystems (Yang et al., 2013) and to avoid excessive delay to any trucks. Meanwhile, this ensures that trucks arriving in the first sub time window are served before those arriving in the second sub-window, which is consistent with the customers' preferences.

Table 2. 1 compares this paper with the closely related studies from four key aspects. This paper is differentiated from previous works in several ways. From the problem perspective, the SCRP-FS we propose generalizes the SCRP in the sense that first, a flexible service policy (SOOO), as opposed to the FCFS policy, is integrated into the multi-stage stochastic optimisation framework. The SOOO policy allows some flexibility in the retrieval sequences of containers in the same group and thus provides more opportunities to reduce the number of relocations and the truck waiting times. Second, instead of assuming uniformly distributed truck arrival order, we propose a more general probabilistic model to describe the randomness of the truck arrivals within the same group. Specifically, our probabilistic model has the capability of capturing the customer preference-based arrival behaviour, which has more practical relevance. From the methodology perspective, we propose a new optimisation framework that not only optimises the expected number of relocations (primary objective) but also optimises the truck waiting times (secondary

objective). Although our exact solution algorithm is built upon the PBFS algorithm in Galle et al. (2018b), the existing PBFS algorithm does not allow for a straightforward adaption to our problem due to the substantial differences between the SCRP and the SCRP-FS. A great deal of effort has been made to adapt the PBFS algorithm to solve our problem. The major adaptions are: first, we construct a more general decision tree with a new structure, which adds a new layer of decision node for sequencing trucks and expresses nodes with a dual-matrix configuration that represents both truck appointment information and customer preference; second, we propose a new lower bound for the expected number of relocations for the SCRP-FS by including the characteristics of flexible retrieval orders and customer preference-based arrivals, which is necessary to prune unpromising nodes; third, we add a procedure for minimising the truck waiting times batch by batch by using the derivation of a waiting time indicator. In addition, we design two fast and efficient heuristic algorithms for the SCRP-FS. Last, we construct a discrete event simulation model to evaluate the exact solutions with a tree structure, which is the pioneer in the relevant literature. The simulation model is especially needed to evaluate the truck waiting times for the exact solutions because the exact algorithms do not record time-related performance.

Table 2. 1 The comparison with the most relevant studies

Characteristics	**The probabilistic model of truck arrival**	**Information updating**	**Service policy**	**Objectives**
Borjian et al. (2015b)	Deterministic	-	A limited number of out-of-order retrievals before each truck	The weighted number of relocations and retrieval delays
Zeng et al. (2019)	Deterministic	-	Out-of-order retrieval within each group	The number of relocations
Borjian et al. (2013)	Scenario-based uncertainty	Two-stage	Out-of-order retrievals s.t. a maximum delay	The weighted expected number of relocations and total delays
Ku and Arthanari (2016a)	Uniform distribution	Multi-stage over individual trucks	FCFS	The expected number of relocations
Galle et al. (2018b)	Uniform distribution	Multi-stage over groups	FCFS	The expected number of relocations
This paper	Customer preference-based uncertainty (incl. uniform distribution)	Multi-stage over groups	Out-of-order retrievals within each sub-group	Primary objective: the expected number of relocations; Secondary objective: the total truck waiting times of each group

2.3 Problem description and formulation

In this section, we first describe the SCRP-FS in detail and then formulate the problem by stochastic dynamic programming.

2.3.1 Problem description

The studied problem is a multi-stage stochastic optimisation problem. The problem is described along with the introduction of the basic assumptions of the SCRP, the probabilistic model of truck arrivals, the containers' attributes, and the service priority.

2.3.1.1 Basic assumptions

The following assumptions are generic to the (S)CRP (e.g. Kim and Hong, 2006; Caserta et al., 2011b; Ku and Arthanari 2016a; Galle et al., 2018b).

A1: Relocations are performed only within the bay being considered. The initial bay layout consisting of S stacks, T tiers, and C containers. In order to avoid infeasible relocations, the storage capacity of the bay is

restricted to be $(S-1)T+1$ containers.

A2: A container is relocatable only when it is blocking the target container.

A3: No new containers arrive at the bay during the container retrieval process.

A4: The travel distance of the trolley and spreader of the yard crane does not have an impact on relocation costs, which means that the relocation effort is measured only by the number of relocations.

A5: Each container is booked to a time window and the corresponding truck will arrive at the terminal within the appointed time window. A ***batch*** of containers (trucks) (i.e., one container corresponds to one truck) is defined as the set of containers (trucks) booked to the same time window. The arrival precedence relationship among batches of trucks is known, but the exact arrival order of trucks within each batch is uncertain, which is revealed as the retrieval proceeds.

A6: For each batch, the full arrival order of trucks from this batch is revealed at once after all containers in its prior batch have been retrieved.

It is worth mentioning that A6 follows the assumption of Galle et al. (2018b), which is based on the phenomenon of truck congestion at gates and yards in busy terminals. The considerable number of trucks in the queue enables the terminal operator to have information about the full arrival order of trucks in a batch before the retrieval of this batch begins.

2.3.1.2 Probabilistic model of truck arrivals

The specific probability distribution of the arrival order of trucks within each batch is hard to predict. The existing studies assume a uniform distribution. In the practical situation, trucks (customers) have their preferred arrival times and may have preferences for either the earlier segment or the latter segment of the booked time windows, which leads to unequal probabilities of arriving in each segment.

We propose a more general probabilistic model of truck arrivals, which can capture customer preference-based arrivals. We divide each appointment time window into two sub-time windows with identical time length. More generally, our proposed approach can be applied to the case where each time window is divided into multiple (more than two) sub-time windows. For the sake of brevity and noticing that the current TAS usually sets 30 min or 60 min for each appointment time window, we only focus on the case of two sub-time windows in this paper. The following assumption is made in the probabilistic model.

A7: 1) Within each batch, the probability of a truck arriving at which sub time window is dependent on customer preference, and 2) within each sub time window, the truck arrival order is drawn from a uniform random permutation.

This enables us to list all potential scenarios of the assignments of a batch of trucks to two sub-time windows with associated probabilities calculated by the customer preference. The calculation is presented in the next sub-section along with the introduction of containers' attributes.

2.3.1.3 Containers' attributes

We introduce containers' attributes to help describe the problem. The following notations are adopted throughout the paper. Let B_k denote the set of containers in batch k and C_k denote the number of containers in batch k, $k \in \{1,...,K\}$. By definition $\sum_{k=1}^{K} C_k = C$. Each container has three attributes:

(1) The first attribute, denoted by l_i, $i \in \{1,...,C\}$, is the ***priority label*** that represents i) the arrival precedence relationship among the trucks and ii) the container retrieval sequence. This label changes during the container retrieval process. Initially, containers in batch k are labelled by L_k that represents the arrival precedence among batches of trucks, which we call ***batch priority*** (see Fig. 2. 1(a)). Let $L_k = 1 + \sum_{j=1}^{k-1} C_j$,

such that given L_k, there is a unique $k \in \{1,...,K\}$. Then, once the full arrival order of trucks in batch *k* is revealed, we get the ***sub-batch priority*** for batch *k*, which represents the arrival precedence among sub-batches of trucks. A ***sub-batch*** of trucks is the set of trucks that have arrived in the same sub-time window. For a container in batch *k*, $k \in \{1,...,K\}$, if its corresponding truck is revealed to arrive in the second sub-time window, its label changes to $L_k + 1$; otherwise, its label does not change. Once the retrieval sequence of a container in batch *k* is determined, its label changes to the exact retrieval sequence that is within $[L_k, L_k + C_k - 1]$.

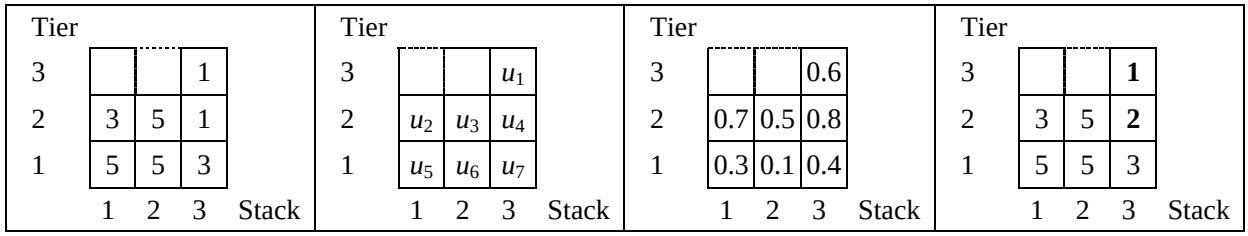
(2) The second attribute, denoted by u_i, $i \in \{1,...,C\}$, is a ***unique ID,*** which is used for identifying individual containers (trucks) (see Fig. 2. 1(b)).

(3) The third attribute, denoted by $p_i \in [0,1]$, $i \in \{1,...,C\}$, represents the ***customer preference*** of container u_i (see Fig. 2. 1(c)). We define that truck u_i arrives at the first sub-time window with p_i and at the second sub-time window with $1 - p_i$. For $k \in \{1,...,K\}$, let ζ_k refer to the possible scenario of sub-batches of trucks in batch *k* and $p(\zeta_k)$ refer to its probability. ζ_k^1 and ζ_k^2 represent two mutually exclusive and collectively exhaustive random sets, the random variables in which take values in $u_i \in B_k$, such that $\{u_i \in \zeta_k^1\}$ is the event that truck u_i arrives in the first sub-batch and $\{u_i \in \zeta_k^2\}$ is the event that it arrives at the second sub-batch. Then, by definition, we have $\mathbb{P}[u_i \in \zeta_k^1] = p_i$ and $\mathbb{P}[u_i \in \zeta_k^2] = 1 - p_i$, $u_i \in B_k$. There are a total of 2^{C_k} possible scenarios of ζ_k for batch *k*, and a total of $\prod_{k=1}^{K} 2^{C_k}$ scenarios for all the batches.

A simple example is given in Fig. 2. 1 to illustrate the containers' attributes and the calculation of the probability of ζ_k. There are seven containers in the initial bay that consists of three stacks and three tiers. Fig. 2. 1(a) displays the priority labels represented by batch priority (1, 3, 5). Fig. 2. 1(b) gives the container/truck ID (u_1 ~ u_7). Fig. 2. 1(c) presents the customer preference (0.0~1.0). Fig. 2. 1(d) displays the revealed sub-batch priority of the first batch in bold. Given the information in Fig. 2. 1(a)-(c), we have the initial bay configuration. For example, container u_1 is located in the third tier of stack three; truck u_1 is in the first batch, which will arrive in the first sub-time window with a probability of 0.6 and the second sub-time window with a probability 0.4. Let us consider ζ_1. There are totally four scenarios of ζ_1, which are respectively $\{\zeta_1^1 = \{u_1\}, \zeta_1^2 = \{u_4\}\}$, $\{\zeta_1^1 = \{u_1, u_4\}, \zeta_1^2 = \varnothing\}$, $\{\zeta_1^1 = \{u_4\}, \zeta_1^2 = \{u_1\}\}$, and $\{\zeta_1^1 = \varnothing, \zeta_1^2 = \{u_1, u_4\}\}$. Their probabilities are computed as: $p(\{\zeta_1^1 = \{u_1\}, \zeta_1^2 = \{u_4\}\}) = 0.6 \times (1 - 0.8) = 0.12$;

$p(\{\zeta_1^1 = \{u_1, u_4\}, \zeta_1^2 = \varnothing\}) = 0.6 \times 0.8 = 0.48$; $p(\{\zeta_1^1 = \{u_4\}, \zeta_1^2 = \{u_1\}\}) = (1 - 0.6) \times 0.8 = 0.32$;

$p(\{\zeta_1^1 = \varnothing, \zeta_1^2 = \{u_1, u_4\}\}) = (1 - 0.6) \times (1 - 0.8) = 0.08$. If truck u_1 has arrived at the terminal in the first sub-time window and truck u_4 has arrived in the second sub-time window as shown in Fig. 2. 1(d), ζ_1 is revealed to be $\{\zeta_1^1 = \{u_1\}, \zeta_1^2 = \{u_4\}\}$.



(a) Batch priority (b) Container (truck) ID (c) Customer preference (d) Sub-batch priority

Fig. 2. 1 An illustration of containers' attributes

2.3.1.4 Service policy

As an alternative to the FCFS policy, we propose a flexible service policy that allows Out-Of-Order retrievals for containers in the same Sub-batch, which is called the SOOO policy. Under this policy, a former sub-batch of trucks is surely served before a latter sub-batch of trucks, and the service sequence for trucks in the same sub-batch is to be determined by the terminal operators. As the root cause of relocation is the mismatch between containers' stacking positions and their retrieval sequences, relocations can be reduced by optimising the retrieval sequence. Besides, as relocation operations increase retrieval times, out-of-order retrievals can also create opportunities for reducing the truck waiting time in the retrieval process. Similar to Galle et al. (2018b), we make the following assumption on the retrieval service begin time of a batch.

A8: The retrieval service of a batch begins at the end of the appointed time window associated with the batch.

A8 can be justified from the following two aspects. Firstly, A8 represents the practical situation of crowded terminals in which trucks often queue up at gates and yards after their arrivals and wait to be served (see Pham et al., 2011; Chen et al., 2013a,b). On one hand, several activities, e.g., security check, permission check, etc. (see Huynh and Zumerchik, 2010) need to be performed at the entry gates before the truck can proceed to the yard. On the other hand, for container terminals having a high level of congestion in the yard, internal waiting queues are also formed and trucks have to wait to be served (Talley and Ng, 2016; Li et al., 2019). Secondly, it is observed that the average truck turnaround time could be much longer than the length of the appointment time window. For example, in Los Angeles-Long Beach, the average truck turn time at the 12 container terminals for the last two years was above 67 minutes and the maximum value was nearly 100 minutes (Mongelluzzo, 2020). For some terminals, a truck appointment system with 30-minutes time windows has been implemented (e.g., Fenix Marine Services container (fenixmarineservices.com), Middle Harbor (Mongelluzzo, 2016)). Given above, it is reasonable to assume that the service of a batch begins at the end of the appointed time window associated with the batch.

2.3.2 Problem formulation

We propose two mathematical models for the SCRP-FS. First, a Sooo model is developed with the objective of minimising the expected total number of relocations to retrieval all containers from a given bay. The Sooo model is important to the terminal operators in reducing relocations; however, it does not consider the truck waiting times. Second, we develop a Sooo extension model to fully take advantage of the flexible service policy. The Sooo extension model has two lexicographically ordered objectives: the primary objective is to minimise the expected total number of relocations, and the second objective is to minimise the total truck waiting times in each batch. To a large extend, our study on the Sooo model is a starting point to develop the Sooo extension model, which is one of the main contributions of this paper. Still, the results of the Sooo are of a certain interest in their own right, and the developed algorithm serves

as building blocks for the exact algorithm for the Sooo extension model.

2.3.2.1 Sooo model

The SCRP is a multi-stage sequential decision-making problem with dynamic information revealing. The stochastic dynamic programming (SDP) method is appropriate to deal with such problems (Bakker et al., 2020). The CRP related problems have been tackled using (stochastic) dynamic programming method, e.g. the deterministic CRP (Kim and Hong, 2006), the SCRP (Ku and Arthanari, 2016a), and the export container stacking problem (Kim et al., 2000; Zhang et al., 2010). In this paper, we formulate the SCRP-FS into an SDP model. The emphasis in SDP is typically in identifying the system states and the actions (variables) at each state (Birge and Louveaux, 2011). In the following, we first define the stage, state, and action for the Sooo model.

Stage: the sequence number of the batch to be retrieved. The example in Fig. 2. 1 is considered as stage 1 since the *1*st batch of containers is to be retrieved.

State: the state of each stage is the state of the bay that consists of the stacking positions of the remaining containers and their attributes. The input state of the *k*th stage is the state of the bay after the (*k*-1)th batch has been retrieved and before the scenario of the sub-batches of the *k*th batch is revealed. For example, Fig. 2. 1(a)-(c) constitute the input state of stage 1.

Action: a feasible action is defined as a sequence of moves to retrieve a batch of containers. Different from the conventional SCRP, the actions in the SCRP-FS consists of two types of actions: (i) the retrieval sequences of the containers (i.e., the service sequence of trucks) in each of the two sub-batches, called *sequencing*, and (ii) the storage positions of the relocated containers, called *relocating*.

With these definitions, optimal actions are taken under uncertainty stage by stage. The uncertainty in the model refers to the scenarios of the sub-batches of each batch (at each stage). At the beginning of a stage, firstly, the scenario of this stage is revealed, and then the optimal actions to retrieve the batch of containers at this stage are sought accordingly considering all the potential scenarios of future stages. The objective is to minimise the expected total number of relocations to retrieve all the containers. Mathematically, the Sooo model can be formulated in a similar way in which it is done in Ku and Arthanari (2016a). The notations used in the model are defined as follows.

K: the total number of batches in the initial bay, which is also the total number of stages.

k: the stage number (the *k*th batch of containers to be retrieved), $k \in \{1,...,K\}$.

ζ_k : The scenario of the sub-batches of stage *k*, $k \in \{1,...,K\}$ (a random variable).

S_k : the input state of stage *k*, that is, the state of the bay after the (*k*-1)th batch has been retrieved and before ζ_k is revealed, $k \in \{1,...,K\}$.

$p(\zeta_k)$: The probability of ζ_k. This is calculated by the probabilistic model of truck arrivals introduced in section 2.3.1.2.

$\mathbf{a}_k(S_k,\zeta_k)$: The actions (a decision variable) taken for retrieving the *k*th batches of containers given S_k and ζ_k. $\mathbf{a}_k(S_k,\zeta_k)=\{\mathbf{a}_k^S(S_k,\zeta_k),\mathbf{a}_k^R(S_k,\zeta_k)\}$, wherein $\mathbf{a}_k^S(S_k,\zeta_k)$ is the retrieval sequence decision of the containers in each of the two sub-batches at stage *k* given S_k and ζ_k, and $\mathbf{a}_k^R(S_k,\zeta_k)$ is the relocation position decision that respects $\mathbf{a}_k^S(S_k,\zeta_k)$. For notational convenience, the dependence on (S_k,ζ_k) is suppressed from $\mathbf{a}_k(S_k,\zeta_k)$, and we use $\mathbf{a}_k$ instead.

$r_k(\mathbf{a}_k|S_k,\zeta_k)$: The number of relocations that are required during action $\mathbf{a}_k$ on the bay of state S_k given ζ_k.

$t_k(S_k,\zeta_k,\mathbf{a}_k)$: The state transition function that maps S_k, ζ_k, and $\mathbf{a}_k$ into the next state S_{k+1}. By

$t_k(S_k, \zeta_k, \mathbf{a}_k)$, the *k*th batch of containers revealed by ζ_k are retrieved according to $\mathbf{a}_k$ from state S_k, after which S_{k+1} is obtained.

$f_k(S_k)$: The expected minimum total number of relocations to retrieve the remaining *K*-*k*+1 batches of containers from the state S_k.

The Sooo model is formulated as a recursive equation as follows:

$$f_1(S_1) = E\left[\min_{\mathbf{a}_1}\{r_1(\mathbf{a}_1 \mid S_1, \zeta_1) + f_2(S_2)\}\right] = \sum_{\zeta_1} p(\zeta_1) \min_{\mathbf{a}_1}\left[r_1(\mathbf{a}_1 \mid S_1, \zeta_1) + f_2(S_2)\right],$$

where $S_2 = t_1(S_1, \zeta_1, \mathbf{a}_1)$ (2.1)

Generally, $f_k(S_k) = \sum_{\zeta_k} p(\zeta_k) \min_{\mathbf{a}_k}\left[r_k(\mathbf{a}_k \mid S_k, \zeta_k) + f_{k+1}(S_{k+1})\right],\ k \in \{1, \ldots, K\}$,

where $S_{k+1} = t_k(S_k, \zeta_k, \mathbf{a}_k)$, for $k \in \{1, \ldots, K\}$, and $f_{K+1}(S_{K+1}) = 0$ (2.2)

The recursive function of equation (2.2) indicates that optimal decisions can be obtained by optimising the recursive function in a backward manner stage by stage.

2.3.2.2 Sooo extension model

The Sooo extension model considers two lexicographically ordered objectives. The primary objective is to minimise the expected total number of relocations and the secondary objective is to minimise the total truck waiting times of each batch sequentially. The use of the secondary objective is justified from the following three perspectives.

Firstly, our motivation for considering the metric of truck waiting times stems from its importance not only to the container terminals but also to the container transport supply chain. The truck waiting time is a key performance indicator that measures the efficiency of storage area at a container terminal (Stahlbock and Voß, 2008; Carlo et al., 2014; Gharehgozli et al., 2016) and is one of the main reasons causing delays in handling external trucks and leading to low quality of customer service (Borjian et al. 2013). A reduction in the truck waiting time would improve the terminals' competitiveness and act as an incentive to encourage external truckers' cooperation, which is essential to achieve a smooth implementation of the flexible service policy.

Secondly, longer truck waiting time in the yard for service leads to higher truck turn time and more emissions (Huynh et al., 2004). Terminal operators have been under enormous pressure from different parties requiring to reduce the truck turn time. For example, from the legislative perspective, the California Assembly Bill AB 2650 became active in 2003 in the US requiring port terminals to lower port-related truck congestion and vehicle emissions. Under this law, external trucks were a major target of regulatory efforts (Giuliano and O'Brien, 2007). Besides, from the economic perspective, some port authorities (e.g., Port Botany; Port Metro Vancouver) have implemented a penalty system that imposes fees on terminals that exceed a specified threshold of truck turn time. In addition, from the perspective of vertical cooperation in the container transport chain, truckers, as an import stakeholder of the hinterland transport, have stated that they won't accept truck appointments until terminals can shorten turn times and end long queues **(**Bonney, 2015). Reducing the truck waiting time in the container retrieval process helps to alleviate yard congestion and thus reduce the truck turn time.

Thirdly, the importance of the truck waiting time metric in the CRP has also been confirmed by the increasing attention it has received in the literature. For instance, López-Plata et al. (2017) minimise the total waiting times of the containers that have expected retrieval times in the deterministic CRP. Borjian et al. (2015b) and Borjian et al. (2013) minimise the weighted sum of total relocations and delays in the deterministic CRP and uncertain CRP respectively. Our study is the first that considers two lexicographically ordered objectives when taking both the number of relocations and the trucks waiting

time into consideration in the uncertain CRP. The reasons to sequence these two objectives (i.e. the number of relocations as the primary objective and the truck waiting time as the secondary objective) are: (i) the number of relocations has a more direct impact on the terminal; (ii) this treatment will appropriately evaluate the effect of the flexible policy on reducing the expected number of relocations compared to the conventional SCRP; (iii) because there may exist multiple optimal solutions to minimising the expected number of relocations in the SCRP-FS, introducing the secondary objective can further optimise the second objective without sacrificing the primary objective.

Fig. 2.2 illustrates the idea behind the Sooo extension model. Let us consider the solutions for the first batch, in which truck u_1 and u_5 have been revealed to be in the first sub-batch and u_7 in the second sub-batch. With regard to the primary objective, there are two optimal solutions to the retrieval sequence for u_1, u_5, and u_7. Solution one is $u_5 \rightarrow u_1 \rightarrow u_7$, and solution two is $u_1 \rightarrow u_5 \rightarrow u_7$. The two solutions contribute the same number of relocations to the expected total number of relocations since no matter which retrieval sequence is used the blocking container u_2 is relocated to stack two. The Sooo extension model wants to choose one that is optimal with respect to the secondary objective, i.e., minimising the total waiting times of the trucks in the first batch/stage. By A8, both truck u_1 and u_5 are ready to be served when the service of this batch begins. If u_5 is retrieved before u_1, truck u_1 suffers waiting due to the relocation of u_2, which could have been avoided if using the alternative solution. Therefore, the optimal solution of the Sooo extension model is $u_1 \rightarrow u_5 \rightarrow u_7$.

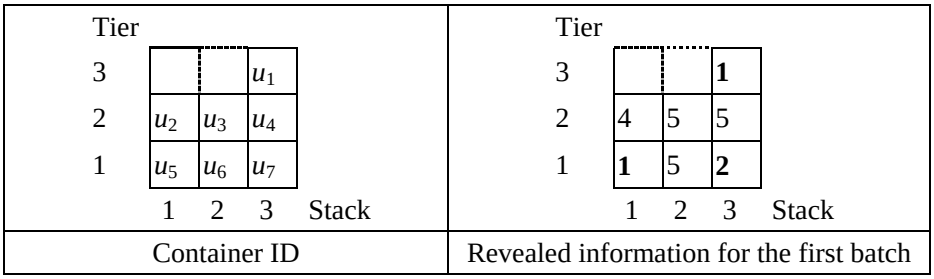


Fig. 2. 2 An example of illustrating the primary objective and the secondary objective

Objective function

Here, we develop the secondary objective function. Before getting retrieval service, truck waiting can occur at any point from arriving outside the in-gate to arriving at the designated yard stack until exiting the out-gate. As our main focus is on the container retrieval process, only the yard-to-retrieval waiting time is of our interest. In particular, we are more concerned with how much waiting times in the container retrieval service process can be saved as a result of the flexible service policy as opposed to the FCFS policy. By the assumption A8, the retrieval service for a truck cannot commence before the end of its appointed time window. Therefore, the waiting time of a truck before the end of its appointed time window is independent of our decision variables. We hence define the truck waiting time as the elapsed time between the end of its appointed time window and its actual retrieval time. To avoid confusion, we use the term "relevant truck waiting time" to represent the truck waiting time considered in this paper. It is worthwhile to note that the relevant truck waiting time under A8 has its practical interpretation. In practice, a truck would have an expected time to retrieve its container, and the difference between that and the actual retrieval time is a common measure of service quality in general terms (López-Plata et al., 2017). For a truck that has booked an arrival time window through the TAS, the end of its appointed time window can be regarded as its expected retrieval time, and the relevant truck waiting time can be used as a measure of service quality for the container retrieval process.

To measure the relevant truck waiting times, we need some time-based notations. Let e_k denote the

end of the appointed time window of the containers in batch *k*. Let d_k denote the completion time of retrieving the last container in batch *k*, and y_k denote the service starting time of batch *k*. The secondary objective is optimised for each stage separately in a sequential way rather than considering global optimisation. This means that when optimising the secondary objective of stage *k*, the service for stage *k*-1 has been completed, that is, S_k is given. This enables us to treat the service completion time of the (*k*-1)th stage (d_{k-1}) as a constant when optimising stage *k*. Given e_k and d_{k-1}, we have y_k by

$$y_k = \max\{e_k, d_{k-1}\}, \quad k \in \{2,...,K\}; \quad y_1 = e_1. \tag{2.3}$$

By equation (2.3), if the service completion time of batch *k*-1 is later than the end of the appointed time window of batch *k*, the service starting time of batch *k* is d_{k-1}. Otherwise, the service starting time of batch *k* coincides with the end of its appointed time window, that is, e_k.

Under the given decisions $\mathbf{a}_k(S_k, \zeta_k) = \{\mathbf{a}_k^S(S_k, \zeta_k), \mathbf{a}_k^R(S_k, \zeta_k)\}$, we now derive the explicit expressions of d_k and the relevant waiting time of truck *i* in batch *k* ($i \in B_k$) under ζ_k, which is denoted by $w_{k,i}^{\zeta_k}$. The following notations are introduced to extract the relevant information implied in the decision variables.

$o_i^{\zeta_k}$: the service order of truck *i*, $i \in B_k$, under ζ_k. Note that $o_i^{\zeta_k}$ is implied in the service sequence decision $\mathbf{a}_k^S(S_k, \zeta_k)$.

$r_i^{\zeta_k}$: the number of relocation moves needed when serving truck *i*, $i \in B_k$, under ζ_k. Note that $r_i^{\zeta_k}$ is implied in the relocation decision $\mathbf{a}_k^R(S_k, \zeta_k)$.

t^{ret} : the handling time per retrieval move.

t^{rel} : the handling time per relocation move.

By A8, all the trucks in a batch have already waited at the yard stack when the service of this batch begins, and thus there is no idle time between the services of any two trucks in the batch. Therefore, d_k is calculated by

$$d_k = y_k + \sum_{i \in B_k} (t^{rel} \cdot r_i^{\zeta_k} + t^{ret}) \tag{2.4}$$

$w_{k,i}^{\zeta_k}$ is calculated by

$$w_{k,i}^{\zeta_k} = (y_k - e_k) + \sum_{j \in B_k, o_j^{\zeta_k} < o_i^{\zeta_k}} (t^{rel} \cdot r_j^{\zeta_k} + t^{ret}) + t^{rel} \cdot r_i^{\zeta_k}, \tag{2.5}$$

The first term on the right side in equation (2.5) is the waiting time between the end of the appointed time window and the start of the service of batch *k*, the second term is the total handling time of the trucks in the batch that are served before truck *i*, and the last term is the relocation time for retrieving the container requested by truck *i*. Equation (2.5) is illustrated in Fig. 2. 3 using the instance in Fig. 2. 2, where the first batch of trucks is served in the sequence: $u_1 \rightarrow u_5 \rightarrow u_7$. By equation (2.5), the waiting time of truck u_1, u_5, and u_7, is respectively $y_k - e_k$, $y_k - e_k + t^{ret} + t^{rel}$, and $y_k - e_k + 2t^{ret} + 2t^{rel}$. It is observed that the handling time of u_1 contributes t^{ret} to the waiting time of both u_5 and u_7 and the handling time of u_5 contributes $t^{ret} + t^{rel}$ to the waiting time of u_7.

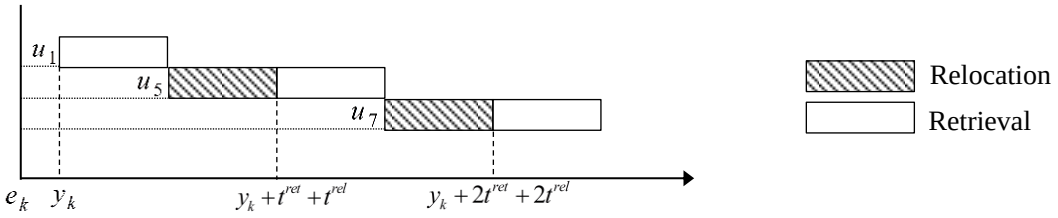


Fig. 2. 3 Illustration of trucks waiting time

Let $W_k^{\zeta_k}$ denote the total waiting times of all the trucks in batch *k* under ζ_k with the given decisions $\mathbf{a}_k(S_k, \zeta_k)$. Then we have

$$\begin{aligned} W_k^{\zeta_k} &= \sum_{i \in B_k} w_{k,i}^{\zeta_k} = \sum_{i \in B_k} \left((y_k - e_k) + \sum_{j \in B_k, o_j^{\zeta_k} < o_i^{\zeta_k}} (t^{rel} \cdot r_j^{\zeta_k} + t^{ret}) + t^{rel} \cdot r_i^{\zeta_k} \right), \\ &= (y_k - e_k) \cdot C_k + \sum_{i \in B_k} (t^{rel} \cdot r_i^{\zeta_k} + t^{ret}) \cdot \left(C_k - o_i^{\zeta_k} \right) + \sum_{i \in B_k} t^{rel} \cdot r_i^{\zeta_k} \end{aligned} \tag{2.6}$$

where the first term on the right hand represents the total waiting time of all trucks in the batch before the service of this batch commences, the second term represents the sum of the handling times of the predecessors of each truck in the batch, and the third term is the total relocation times for retrieving all the containers in the batch. Because the service time of truck *i* adds to the waiting times of all its successors in the batch, $(t^{rel} \cdot r_i^{\zeta_k} + t^{ret})$ is weighted by $(C_k - o_i^{\zeta_k})$. In other words, a part of the waiting time of a truck is a cumulative service time of all its predecessors in the batch (see Fig. 2. 3).

Let $\gamma_k^{(1)}(S_k)$ denote the primary objective for stage *k*, which is the expected minimum total number of relocations to retrieve the remaining *K*-*k*+1 batches of containers from the state S_k, and $\gamma_k^{(2)}(S_k)$ denote the secondary objective for stage *k*, which is the expected minimum total waiting times for the trucks in batch *k* with the state S_k. Given S_k, the Sooo extension model aims to find the solution that minimises the secondary objective $\gamma_k^{(2)}(S_k)$ among the set of solutions that minimise the primary objective $\gamma_k^{(1)}(S_k)$. Then, the Sooo extension model for stage *k* with the state S_k, $k \in \{1, ..., K\}$, is formulated as follows:

$$\gamma_k^{(1)}(S_k) = f_k(S_k), \text{ which is defined in Eq. (2.2)}$$

$$\gamma_k^{(2)}(S_k) = \sum_{\zeta_k} p(\zeta_k) \min_{\mathbf{a}_k} W_k^{\zeta_k} \tag{2.7}$$

Derivation of optimality

Observation 1. The optimal solution of the Sooo extension model at stage *k* from the state S_k under ζ_k is the one that minimises $\sum_{i \in B_k} r_i^{\zeta_k} \cdot \left(C_k - o_i^{\zeta_k} + 1 \right)$ among the set of optimal solutions $\mathbf{a}_k(S_k, \zeta_k)$ with regard to the primary objective.

Proof. At each stage *k*, d_{k-1} is known and thus y_k can be calculated by equation (2.3). Besides, e_k, C_k, t^{ret} and t^{rel} are constant. Therefore, $\min_{\mathbf{a}_k} W_k^{\zeta_k}$ in equation (2.7) is equivalent to

$$\min_{\mathbf{a}_k} \sum_{i \in B_k} r_i^{\zeta_k} \cdot \left(C_k - o_i^{\zeta_k} + 1 \right), \tag{2.8}$$

where $r_i^{\zeta_k} \cdot \left(C_k - o_i^{\zeta_k} + 1\right)$ represents the contribution of the number of relocations for retrieving container i on the waiting time of truck i itself and on the waiting times of all its successors in the batch. □

Observation 1 indicates that at each stage, the secondary objective is jointly determined by the number of relocations for each retrieval in the batch and its retrieval sequence.

2.4 Exact solution algorithms based on decision tree

By applying the recursive equation of the Sooo model, the optimal solutions can be obtained backward from stage K to stage 1. This procedure is usually executed by a tree search-based algorithm in a state-space constructed by a decision tree. The state-of-the-art tree search-based algorithm for the SCRP in which information is revealed on a batch basis is the Pruning-Best-First-Search (PBFS) algorithm proposed by Galle et al. (2018b). To solve the Sooo model, we develop an Adapted Pruning-Best-First-Search (APBFS) algorithm by extending the PBFS algorithm; we then extend the APBFS algorithm to solve the Sooo extension model. The PBFS is developed for the FCFS rule and does not consider customer preference, which does not allow for a straightforward adaption. The main differences between our algorithms from existing ones are explained below. Firstly, our decision tree is more general, which can support two types of decision-making under flexible service policies and incorporate customer preference. In particular, our decision tree is composed of three classes of nodes and each node is represented by a dual-matrix configuration, which differs from the PBFS decision tree that is composed of two classes of nodes each represented by a single-matrix configuration. Secondly, the new structure of the decision tree necessitates several major adaptions in the search: first, a new decision layer is considered during searching and back-tracking; second, another two techniques used to decrease the search space are modified: the abstract technique and the lowest level to stop branching. Thirdly, a new lower bound for the expected number of relocations for the SCRP-FS is proposed to prune unnecessary nodes. We consider the new lower bound as one of the major contributions of this paper since this is the first lower bound for the SCRP-FS in the literature. Fourthly, our extended APBFS expands the PBFS by having the capability of minimising the total truck waiting times of each batch, which has been neglected in the relevant literature.

In sub-sections 2.4.1-2.4.3, we first introduce the elements of the proposed algorithms with a focus on the differences between the PBFS and the APBFS. Then, in sub-section 2.4.4 and 2.4.5, we present the APBFS for the Sooo model and the extended APBFS for the Sooo extension model respectively. The contribution of the extended APBFS is introduced in sub-section 2.4.5.

2.4.1 Constructing a decision tree

In a typical decision tree for the SCRP, the root node represents the initial configuration and the leaf nodes represent the empty configuration. Between the root node and leaf nodes, there are two types of intermediate nodes: chance nodes and decision nodes, which alternate in some way to form the tree. A chance node is to model the stochasticity of trucks' arrival while a decision node is to model possible actions. In the PBFS, each node is represented by a single matrix that corresponds to the truck arrival orders.

In the (extended) APBFS algorithm, each node is represented by a ***dual-matrix configuration*** that is composed of a priority matrix and a preference matrix (see Fig. 2. 4). The ***priority matrix*** represents the priority labels of containers and the ***preference matrix*** represents the probability of trucks arriving in the

first sub-time window (i.e. the customer preference). The structure of the decision tree consists of ***three classes of nodes***, which are ***chance nodes***, ***SD nodes*** (sequencing decision nodes) and ***RD nodes*** (relocating decision nodes). The SD nodes create a new decision layer between the chance nodes and the RD nodes to sequence trucks. In the following, we define these nodes with the introduction of relevant notations.

A ***chance node*** corresponds to S_k, $k \in \{1,...,K\}$, in which the scenario of sub-batches of batch *k* (i.e., ζ_k) is to be revealed (see e.g., node 0 in Fig. 2. 4). From a chance node, descendant nodes are created by ζ_k, denoted by S_k' (see e.g., node 1). Let $\xrightarrow{\zeta_k}$ represent the revelation of ζ_k, we have $S_k \xrightarrow{\zeta_k} S_k'$, $k \in \{1,...,K\}$.

After the revelation of the random variable, actions are taken to retrieve the containers in batch *k* from S_k'. A ***SD node*** corresponds to S_k', in which the retrieval sequence for the *k*th batch, denoted by D_k^S, is to be determined. From a SD node, descendant nodes, denoted by X_{L_k} (e.g., node 5, node 6), are created by applying D_k^S. Recall that as defined in section 2.3.1.3, L_k represents the batch priority such that each batch *k* corresponds to an unique L_k. Let $\xrightarrow{D_k^S}$ represent the application of D_k^S, we have $S_k' \xrightarrow{D_k^S} X_{L_k}$. In a ***RD node***, the target container (the container with the smallest label) is to be retrieved and the relocation decisions to retrieve the container is to be determined. For batch *k*, starting from the SD node corresponding to S_k', C_k levels of RD nodes are created sequentially, denoted by X_i, $i \in \{L_k,...,L_k + C_k - 1\}$ (e.g., node 5, 7). Let D_i^R denote a sequence of moves (relocation moves and retrieval move) to retrieve the *i*th container, and X_{i+1} denote the configuration after applying action D_i^R to X_i and before applying action $D_{i+1}^R, i \in \{L_k,...,L_k + C_k - 1\}$. Let $\xrightarrow{D_i^R}$ represent the application of D_i^R, we have $X_i \xrightarrow{D_i^R} X_{i+1}$, $i \in \{L_k,...,L_k + C_k - 2\}$.

After the retrieval of the last container in the *k*th batch whose retrieval sequence is $L_k + C_k - 1$, $X_{L_k + C_k - 1}$ transits to the next chance node corresponding to S_{k+1} (e.g., node 10), which is represented by $X_{L_k + C_k - 1} \xrightarrow{D_{L_k + C_k - 1}^R} S_{k+1}$.

To summarize, for $k \in \{1,...,K\}$, the state transitions from S_k to S_{k+1} in the tree search are modelled by:

$$\text{if } C_k > 1, \begin{cases} S_k \xrightarrow{\zeta_k} S_k' \\ S_k' \xrightarrow{D_k^S} X_{L_k} \\ X_i \xrightarrow{D_i^R} X_{i+1}, \forall i \in \{L_k,...,L_k + C_k - 2\} \\ X_{L_k + C_k - 1} \xrightarrow{D_{L_k + C_k - 1}^R} S_{k+1} \end{cases}$$

$$\text{if } C_k = 1, \; S_k \xrightarrow{D_{L_k}^R} S_{k+1} \tag{2.9}$$

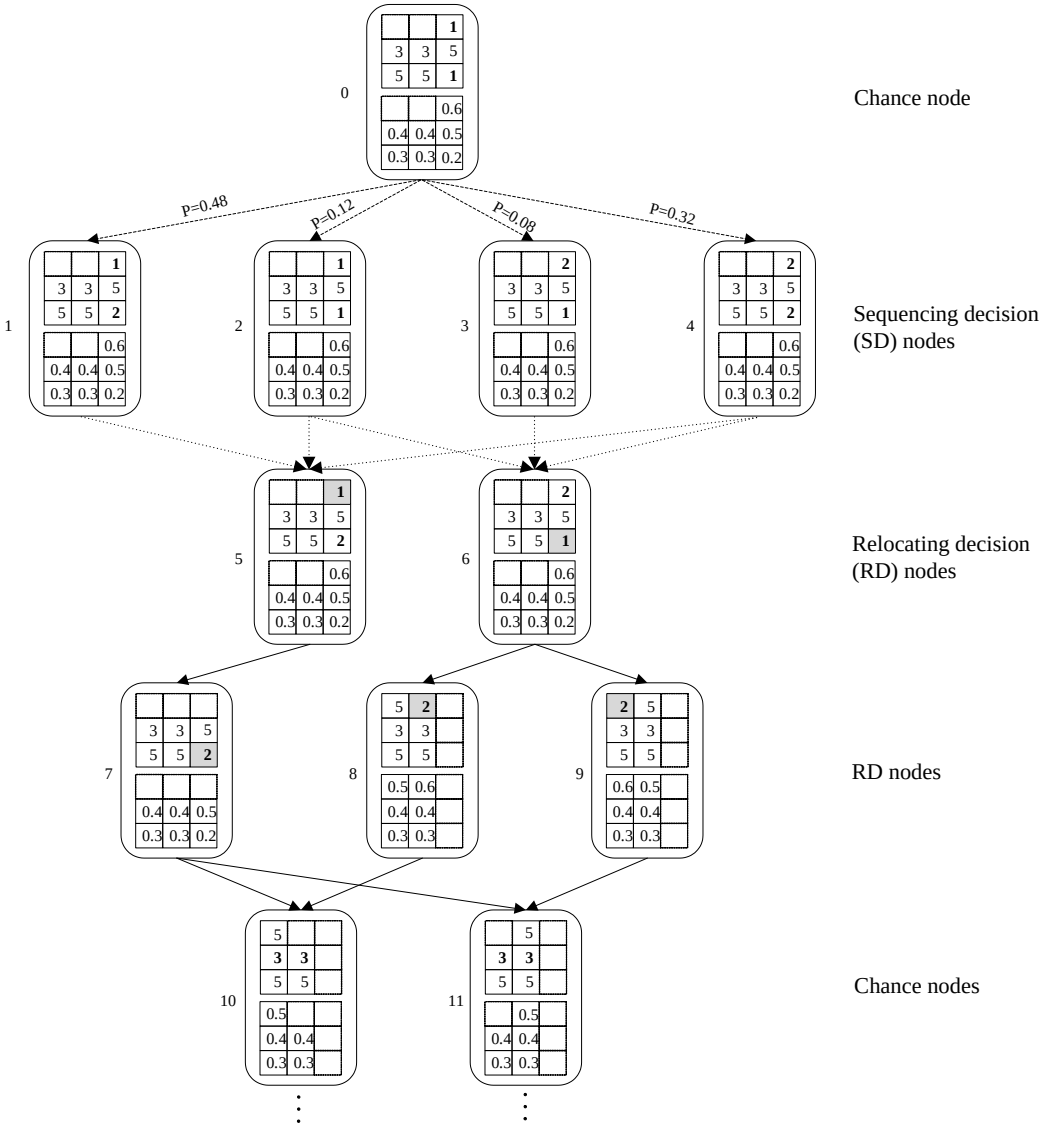


Fig. 2. 4 A sample decision tree

Dashed-lines: the revelation of the scenario of sub-batches; Dotted-lines: applying the sequencing action on retrieval order; Solid-lines: applying the relocating action and retrieving the containers; Containers in bold font: target batch of containers; Containers in the shaded slot: target container to be retrieved.

2.4.2 Back-tracking in the decision tree

Given a full decision tree, the optimal objective value is calculated by back-tracking. In this section, we only focus on the primary objective. Let n be a node in the decision tree. Each node has a cost-to-go function, denoted by $f(n)$, which represents the expected cost of the cheapest path from node n to the leaf node. Let n=0 denote the root node, and then the objective function is denoted by $f(0)$. The basic idea of back-tracking is to compute the cost-to-go function $f(n)$ for each node n recursively from the bottom up of the tree with the ultimate goal to obtain $f(0)$. To calculate $f(n)$, we need the immediate cost function, denoted by $r(n)$, which represents the cost incurred by the action taken to transit node n to its offspring. In the SCRP-FS, $f(n)$ is defined for all three types of nodes, which represents the minimum expected number of relocations required to retrieve all remaining containers from node n. $r(n)$is only defined for RD nodes, which represents the number of relocations required in order to retrieve the target container in node n. The following notations are used to calculate $f(n)$.

λ_n : the number of remaining containers in node *n*, which is defined as the level of *n*. If *n* is a chance node or a SD node, there exists a unique $k \in \{1,...,K\}$ such that $L_k = C - \lambda_n + 1$.

Ω_n : the set of offspring of a chance node *n*. Each node in Ω_n is a SD node that corresponds to a realisation of the random variable ζ_k, and thus $|\Omega_n| = 2^{C_k}$.

Φ_n : the set of offspring of a SD node *n*. Each node in Φ_n corresponds to a feasible retrieval order for the containers in batch *k*, wherein $L_k = C - \lambda_n + 1$, and thus $|\Phi_n| = |\zeta_k^1|!|\zeta_k^2|!$.

Δ_n : the set of offspring of a RD node *n*. Δ_n is constructed greedily by considering all feasible combinations of the relocation positions of the $r(n)$ blocking containers in node *n*. The maximum value of $|\Delta_n|$ is given by $(S-1)^{r(n)}$ when the number of empty slots in each candidate stack is no less than $r(n)$, wherein *S* is the number of stacks.

p_{n_i}: the probability of a SD node $n_i \in \Omega_n$, which is calculated by our probabilistic model of truck arrivals.

Given the above definitions, for each node *n*, we have:

$$f(n) = \begin{cases} \sum_{n_i \in \Omega_n} p_{n_i} f(n_i), \text{if (i) } n \text{ is a chance node,} \\ \min_{n_i \in \Phi_n} \{ f(n_i) \}, \text{if (ii) } n \text{ is a SD node,} \\ r(n) + \min_{n_i \in \Delta_n} \{ f(n_i) \}, \text{if (iii) } n \text{ is a RD node.} \end{cases} \tag{2.10}$$

In equation (2.10), the "if (ii)" condition is a new decision layer to the PBFS algorithm. We use the example in Fig. 2. 4 to illustrate equation (2.10). Suppose $f(10) = f(11) = 1$ is given. Then the $f(n)$ of other nodes are calculated as follows: $f(7) = r(7) + \min\{ f(10), f(11)\} = 1 + 1 = 2$; $f(5) = r(5) + \min\{ f(7)\} = 0 + 2 = 2$; $f(1) = \min\{ f(5)\} = 2$; $f(8) = r(8) + \min\{ f(10)\} = 0 + 1 = 1$; $f(9) = r(9) + \min\{ f(11)\} = 0 + 1 = 1$; $f(6) = r(6) + \min\{ f(8), f(9)\} = 2 + 1 = 3$; $f(2) = \min\{ f(5), f(6)\} = 2$. It confirms that the optimal offspring of node 2 with regard to the primary objective is node 5. $f(3) = \min\{ f(6)\} = 3$; $f(4) = \min\{ f(5), f(6)\} = 2$. It confirms that the optimal offspring of node 4 with regard to the primary objective is node 5. $f(0) = p_1 * f(1) + p_2 * f(2) + p_3 * f(3) + p_4 * f(4) = 0.48*2 + 0.12*2 + 0.08*3 + 0.32*2 = 2.08$.

2.4.3 Techniques to decrease the size of the decision tree

For larger problems, considering a full decision tree in the tree search becomes computationally cumbersome due to the exponential growth of the search space with the growing size of the problem. In the PBFS, a combination of four techniques has been proposed to reduce the size of the tree, while ensuring the optimality of the solution. The first one is the BFS (Best-First-Search) exploration strategy based on a valid lower bound, which determines the search direction of the tree. The BFS first explores the nodes with smaller lower bound, because these nodes are the most promising nodes that are most likely to return small $f(.)$. The second technique is pruning with a lower bound. By using the pruning strategy, a node is fathomed if its lower bound is greater than or equal to the best $f(.)$ of the explored nodes. The third technique is stopping the search at a level λ^* at which $f(.)$ can be obtained using specific techniques without the need for further branching. Finally, the abstracting technique is used to avoid re-generating and re-computing identical nodes.

To use these techniques in the APBFS algorithm, we make the following major adaptions. Firstly, we extend the abstract technique by using dual-matrix configurations. Secondly, we derive a new lower bound

for the SCRP-FS. Thirdly, we use $\lambda^* = S$ to stop further branching. The following three sub-sections present these adaptions respectively.

2.4.3.1 Abstraction technique

The abstraction technique is first studied by Ku and Arthanari (2016b) to reduce the search space of the CRP and then is used by Ku and Arthanari (2016a) and Galle et al. (2018b) for the SCRP. The rationale behind this technique is that some configurations are actually equivalent in terms of their contributions to the objective function, and thus duplicate nodes can be avoided. Generally, each newly generated node is abstracted by using a projection rule, after which we determine whether to keep this node or not by comparing it with the nodes that have been explored in the same level. In the (extended) APBFS algorithm, two nodes are regarded as equivalent only when both their abstract priority configurations and abstract preference configurations are identical. This is different from the PBFS. The projection procedure of the abstraction technique for the (extended) APBFS algorithm is as follows, and an illustration is provided in Appendix A.1. We denote the application of this procedure to node *n* as $Abstract(n)$.

Step 1: Rank the stacks within the priority configuration according to the heights of stacks in ascending order. Ties are broken by ranking them lexicographically in ascending order according to the priority labels of the containers from top tier to bottom tier.

Step 2: (obtain abstract priority configuration): Re-arrange the stacks within the priority configuration according to their rankings so that lower-ranked stacks are located on the left and higher-ranked ones on the right.

Step 3: (obtain abstract preference configuration): Re-arrange the stacks within the preference configuration in the same order of the rankings of the stacks in the priority configuration.

Remark: we observe that executing the abstract technique could be time-consuming as a newly generated node has to be compared with the configurations of all the existing nodes at the same level. For example, it takes about 100 seconds to implement the abstract technique on a node that needs to be compared with 7722 nodes at the same level. Future research may seek more efficient abstract techniques that allow efficient checking for repeated states, such as using a hash table that compares the hash value of two nodes instead of their exact configurations (Russell and Norvig, 2016).

2.4.3.2 Lower bound

A new lower bound for the SCRP-FS is proposed to prune unpromising nodes. Here we only consider the lower bound on the blocking containers, which is the expected number of containers that must be relocated at least once in order to retrieve all the containers from a node *n*, denoted by *lb*(*n*). We care about the blocking lower bound for RD nodes and chance nodes, while a lower bound for SD nodes is unnecessary because each SD node needs to be explored to return the cost-to-go function of a chance node. In the SCRP-FS, due to the application of the SOOO policy and the incorporation of customer preference, the probability of a container being blocking is different from the conventional SCRP. From now on throughout the paper, we refer to the method of calculating the blocking lower bound for the SCRP-FS as *LB-FS*. In the following, we explain how to compute *lb*(*n*) by *LB-FS*.

Lemma 1. Let *n* be a node with *S* stacks and *T* tiers, each stack containing H_s containers ($0 \le H_s \le T$). Let n_h^s denote the container located at stack s ($s \in \{1,...,S\}$) and tier h ($h \in \{1,...,H_s\}$), $l_{n_h^s}$ denote the priority label of container n_h^s, and $p_{n_h^s}$ denote the customer preference for container n_h^s, then we have:

$$lb(n)=\sum_{\substack{s=1,\\ H_s>1}}^{S}\sum_{h=2}^{H_s}\left(1\left\{l_{n_h^s}>\min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}\right\}\times 1+1\left\{l_{n_h^s}=\min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}\right\}\times\left(1-p_{n_h^s}\right)\times\left(1-\prod_{\substack{i=1\\ l_{n_i^s}=l_{n_h^s}}}^{h-1}\left(1-p_{n_i^s}\right)\right)\right) \tag{2.11}$$

where $1\{A\}$ is the indicator function of A: $1\{A\}$ = 1 if condition A is true; and 0 otherwise.

Proof. The basic idea of computing $lb(n)$ is to compute the expectation of a single container being blocking and then sum up the expectation for all the containers in node n. Let us fix s and compute the probability that container n_h^s is blocking. Clearly, for $H_s=0$, container n_h^s is not blocking for sure. Now we consider $H_s>1$. Obviously, if $l_{n_h^s}<\min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}$, container n_h^s is not blocking. Then, we consider the following two cases in which container n_h^s may be blocking.

(i) If $l_{n_h^s}>\min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}$, then container n_h^s is surely blocking. In this case, the probability that container n_h^s is blocking is equal to 1. In the example of Fig. 2. 5 (a chance node), container n_5^s meets this condition as $l_{n_5^s}=2>\min_{i\in\{1,\ldots,4\}}\left\{l_{n_i^s}\right\}=1$, and thus container n_5^s contributes one blocking container to $lb(n)$.

n_5^s	2	0.6
n_4^s	1	0.0
n_3^s	2	0.2
n_2^s	2	0.4
n_1^s	2	0.5

Fig. 2. 5 Illustration of a single stack configuration for computing the blocking lower bound

(ii) Otherwise, $l_{n_h^s}=\min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}$, which means there are containers below n_h^s with the same label, then container n_h^s is blocking with probability. This case makes the probability a container being blocking different from that in the Galle et al. (2018b). Recall the SOOO policy: the first sub-batch of trucks is given higher service priority over the second sub-batch of trucks; the trucks in the same sub-batch are given the same service priority. Therefore, n_h^s is surely blocking only in the situation where n_h^s belongs to the second sub-batch and there is at least one container with the same label below n_h^s belonging to the first sub-batch. In this condition, the probability that container n_h^s is blocking is equal to $\left(1-p_{n_h^s}\right)\times\left(1-\prod_{\substack{i=1\\ l_{n_i^s}=l_{n_h^s}}}^{h-1}\left(1-p_{n_i^s}\right)\right)$, where $1-p_{n_i^s}$ is the probability that container n_h^s belongs to the second sub-batch, and $1-\prod_{\substack{i=1\\ l_{n_i^s}=l_{n_h^s}}}^{h-1}\left(1-p_{n_i^s}\right)$ is the probability that at least one container with label $l_{n_h^s}$ below n_h^s belong to the first sub-batch. In Fig. 2. 5, container n_2^s and n_3^s meet this condition. $\mathbb{P}\left[n_2^s \text{ is blocking}\right]=(1-0.4)\times(1-(1-0.5))=0.3$; $\mathbb{P}\left[n_3^s \text{ is blocking}\right]=(1-0.2)\times(1-(1-0.4)\times(1-0.5))=0.56$.

Combining the above two cases, we have,

$$\mathbb{P}\left[n_h^s \text{ is blocking}\right] = \mathbb{1}\left\{l_{n_h^s} > \min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}\right\}\times 1 + \mathbb{1}\left\{l_{n_h^s} = \min_{i\in\{1,\ldots,h-1\}}\left\{l_{n_i^s}\right\}\right\}\times\left(1-p_{n_h^s}\right)\times\left(1-\prod_{\substack{i=1\\ l_{n_i^s}=l_{n_h^s}}}^{h-1}\left(1-p_{n_i^s}\right)\right).$$

Summing the above equation for $h\in\{2,\ldots,H_s\}$ and $s\in\{1,\ldots,S\}$, we have equation (2.11), which completes the proof. □

By equation (2.11), the $lb(\cdot)$ of the single stack in Fig. 2.5 is calculated as:

$$\mathbb{P}\left[n_2^s \text{ is blocking}\right] + \mathbb{P}\left[n_3^s \text{ is blocking}\right] + \mathbb{P}\left[n_5^s \text{ is blocking}\right] = 0.3+0.56+1=1.86.$$

2.4.3.3 Lowest level to stop branching

Another technique to decrease the size of the decision tree is stopping branching at an early level λ^* without the need for traversing to the leaf node. λ^* is regarded as the lowest level of the tree. The PBFS algorithm stops branching at $\lambda^* = \max\{S, C_K\}$ and computes *f*(.) either using a lower bound or A^* algorithm (an efficient algorithm for the classical CRP). Here we use λ^*=*S* and compute *f*(.) using *LB-FS*. Noticing that the number of containers in a chance node and in its offspring (a SD node) is equal, it does not make sense to stop the search at a SD node, because we can stop the search at the chance node as soon as λ^* is satisfied. In other words, the lowest level of the tree will be reached at either a chance node or a RD node.

Lemma 2. Let *n* be a chance node or a RD node with *S* stacks, *T* tiers, and *S* containers, and $\lambda_n = \lambda^* = S$, then $lb(n) = f(n)$.

Proof. Since there are only *S* containers remaining to be retrieved and there are *S* stacks, any blocking container can be relocated to an empty stack. As a result, the relocated container will never block other containers again. Therefore, each blocking container at node *n* will have only one relocation in the optimal solution. In addition, each relocation in the optimal solution is unavoidable according to the definition of *lb*(*n*). This implies that the optimal solution *f*(*n*) = *lb*(*n*), which completes the proof. □

2.4.4 The APBFS algorithm for the Sooo model

Built upon the elements introduced above, we present the whole framework of the Adapted Pruning-Best-First-Search (APBFS) algorithm for the Sooo model. The following notations are used for describing the algorithm.

Ω_n^{APBFS}: the set of offspring of chance node *n* that is used to compute $f(n)$, which is the subset of Ω_n.

Φ_n^{APBFS}: the set of offspring of SD node *n* that is used to compute $f(n)$, which is the subset of Φ_n.

Δ_n^{APBFS}: the set of offspring of RD node *n* that is used to compute $f(n)$, which is the subset of Δ_n.

Give a configuration *n* and lower bound *LB-FS*, the steps of the APBFS algorithm to return $f(n)$ is as follows.

Algorithm 1. APBFS algorithm $f(n) = APBFS(n, LB\text{-}FS)$

Step 1. Identify the class of node *n*. If *n* is a SD node, go to Step 4. Otherwise, go to Step 2.

Step 2. If the level of *n* is not greater than *S*, compute $f(n)$ using $lb(n)$. Otherwise, go to Step 3.

Step 3. If *n* is a chance node, compute $f(n)$ following Step 3.1~3.3.

Step 3.1. Construct Ω_n by considering all possible scenarios of sub-batches to retrieve the target batch of containers in node *n*. Compute the probability of each node n_i in Ω_n.

Step 3.2. Apply *Abstract*(.) to each node n_i in Ω_n. If the abstract configuration is new, add n_i to Ω_n^{APBFS} and compute $f(n_i)$. If the abstract configuration is identical to a node *m* that is already in Ω_n^{APBFS}, add the probability of n_i to the probability of *m*.

Step 3.3. Compute $f(n)$ by summing up the expectation of $f(n_i)$ for each n_i, $n_i \in \Omega_n^{APBFS}$.

Step 4. If *n* is a SD node, compute $f(n)$ following Step 4.1~4.6.

Step 4.1. Construct Φ_n by considering all feasible retrieval sequences to retrieve the target batch of containers in node *n*.

Step 4.2. Apply *Abstract*(.) to each node n_i in Φ_n to avoid duplicate configurations, which leads to Φ_n'.

Step 4.3. Compute the lower bound $lb(n_i)$ for each node n_i in Φ_n'. Sort the nodes in Φ_n' in non-decreasing order of $lb(.)$.

Step 4.4. Compute $f(n_{(1)})$, wherein $n_{(1)}$ is the node with the smallest lower bound in Φ_n', and add $n_{(1)}$ to Φ_n^{APBFS}.

Step 4.5. Repeat for each of the remaining nodes in Φ_n' in non-decreasing order of $lb(.)$ to construct Φ_n^{APBFS}: If the lower bound of the considered node $n_{(k)}$ is less than the smallest *f*(.) of the nodes in Φ_n^{APBFS}, apply *Abstract*(.) to node $n_{(k)}$. If the abstract configuration is identical to a node *m* that is already in the decision tree, then add *m* to Φ_n^{APBFS}; otherwise, add the considered node to Φ_n^{APBFS} and compute the cost-to-go function of the considered node $f(n_{(k)}) = APBFS(n_{(k)}, LB\text{-}FS)$.

Step 4.6. Determine $f(n)$ by taking the minimal value of $f(n_i)$, $n_i \in \Phi_n^{APBFS}$.

Step 5. If *n* is a RD node, compute $f(n)$ following Step 5.1~5.6.

Step 5.1. Construct Δ_n by considering all feasible relocation moves to retrieve the target container in node *n*.

Step 5.2. Apply *Abstract*(.) to each node n_i in Δ_n to avoid duplicate configurations, which leads to Δ_n'.

Step 5.3. Compute the lower bound $lb(n_i)$ for each node n_i in Δ_n'. Sort the nodes in Δ_n' in non-decreasing order of $lb(.)$.

Step 5.4. Compute $f(n_{(1)})$, wherein $n_{(1)}$ is the node with the smallest lower bound in Δ_n', and add $n_{(1)}$ to Δ_n^{APBFS}.

Step 5.5. Repeat for each of the remaining nodes in Δ_n' in non-decreasing order of $lb(.)$ to construct Δ_n^{APBFS}: If the lower bound of the considered node $n_{(k)}$ is less than the smallest *f*(.) of the nodes in Δ_n^{APBFS}, apply *Abstract*(.) to node $n_{(k)}$. If the abstract configuration is identical to a node *m* that is already in the decision tree, then add *m* to Δ_n^{APBFS}; otherwise, add the considered node to Δ_n^{APBFS} and compute the cost-to-go function of the considered node $f(n_{(k)}) = APBFS(n_{(k)}, LB\text{-}FS)$.

Step 5.6. Determine $f(n)$ by taking the minimal value of $f(n_i)+r(n)$, $n_i \in \Delta_n^{APBFS}$.

2.4.5 The extended APBFS algorithm for the Sooo extension model

In this section, the APBFS algorithm is extended to solve the Sooo extension model optimally. The basic

idea of the extended APBFS algorithm is, for each SD node *n*, to find its best offspring that minimises the secondary objective among its offspring that minimise the primary objective, i.e., *f*(*n*). Because of Observation 1 in Section 2.3.2.2, the secondary objective can be substituted by a waiting time indicator. With Observation 1, we define $\sum_{i \in B_k} r_i^{\zeta_k} \cdot \left(C_k - o_i^{\zeta_k} + 1\right)$ as the ***waiting time indicator*** of batch *k* under ζ_k, which is jointly determined by the container retrieval sequence in batch *k* and the number of relocations for each retrieval in batch *k*. Recalling in the APBFS algorithm, the container retrieval sequence of batch *k* is included in the immediate offspring (RD node) of the SD node *n* ($\lambda_n = C - L_k + 1$), denoted by node *m*. Given such a RD node *m*, the optimal number of relocations for each retrieval from batch *k* can be obtained by tracing the series of optimal offspring of node *m*. Therefore, in the extended APBFS algorithm, the focus is on selecting the optimal immediate offspring of SD nodes by using the waiting time indicator. To this end, the APBFS is extended from three perspectives. First, the pruning strategy is adjusted to explore each candidate node that is promising with regard to the secondary objective. Specifically, for each SD node *n*, its offspring whose lower bounds are equal to the best *f*(*n*) found so far are not pruned (Step 4.5 in Algorithm 3). Second, for the immediate offspring of SD node *n* whose cost-to-go functions are equal to *f*(*n*), we compute their waiting time indicators and then choose the one with the minimum waiting time indicator as the best offspring of node *n* (Step 4.7-4.8 in Algorithm 3). For a SD node *n*, the waiting time indicator of its immediate offspring n_i, denoted by $w(n_i)$, is given by *WaitTimeIndic*(n_i, *n*) in Algorithm 2. Lastly, the decision tree is traversed to level one, i.e., $\lambda^* = 1$ (Step 2 of Algorithm 3), because our *LB-FS* does not apply to the secondary objective.

Algorithm 2. Waiting time indicator of the immediate offspring n_i of SD node *n*: $w(n_i)$ = $WaitTimeIndic(n_i, n)$

Step 1. Set C_n to be the number of containers in the target batch in node *n*.

Step 2. Set $m = n_i$ and $w(n_i) = 0$.

Step 3. For *j* from 1 to C_n, do Step 3.1~3.3.

Step 3.1. Set *r*(*m*) to be the number of blocking containers in node *m*.

Step 3.2. Add $r(m) \cdot (C_n - j + 1)$ to $w(n_i)$.

Step 3.3. Update *m* by letting the new *m* become the optimal offspring of the current *m*.

For the purpose of completeness, the steps of the extended APBFS algorithm for computing *f*(*n*) given a configuration *n* and lower bound *LB-FS* are presented below, and a sample decision tree developed by the extended APBFS algorithm is provided and explained in Appendix A.2. The differences from the APBFS algorithm are highlighted in bold font. Note that the *f*(*n*) returned by the APBFS algorithm and the extended APBFS algorithm is exactly the same, but the best offspring of SD node *n* may be different due to the consideration of the secondary objective function.

Algorithm 3. Extended APBFS algorithm $f(n) = Extended\ APBFS(n, LB\text{-}FS)$

Step 1. Identify the class of node *n*. If *n* is a SD node, go to Step 4. Otherwise, go to Step 2.

Step 2. If the level of *n* is not greater than **one, return zero to** $f(n)$. Otherwise, go to Step 3.

Step 3. If *n* is a chance node, compute $f(n)$ following Step 3.1~3.3.

Step 3.1. Construct Ω_n by considering all possible scenarios of sub-batches to retrieve the target batch of containers in node *n*. Compute the probability of each node n_i in Ω_n.

Step 3.2. Apply *Abstract*(.) to each node n_i in Ω_n. If the abstract configuration is new, add n_i to Ω_n^{APBFS} and compute $f(n_i)$. If the abstract configuration is identical to a node *m* that is

already in Ω_n^{APBFS}, add the probability of n_i to the probability of *m*.

Step 3.3. Compute $f(n)$ by summing up the expectation of $f(n_i)$ for each n_i, $n_i \in \Omega_n^{APBFS}$.

Step 4. If *n* is a SD node, compute $f(n)$ and **return the optimal offspring of *n* following Step 4.1~4.8.**

Step 4.1. Construct Φ_n by considering all feasible retrieval sequences to retrieve the target batch of containers in node *n*.

Step 4.2. Apply *Abstract*(.) to each node n_i in Φ_n to avoid duplicate configurations, which leads to Φ_n'.

Step 4.3. Compute the lower bound $lb(n_i)$ for each node n_i in Φ_n'. Sort the nodes in Φ_n' in non-decreasing order of $lb(.)$.

Step 4.4. Compute $f(n_{(1)})$, wherein $n_{(1)}$ is the node with the smallest lower bound in Φ_n', and add $n_{(1)}$ to Φ_n^{APBFS}.

Step 4.5. Repeat for each of the remaining nodes in Φ_n' in non-decreasing order of $lb(.)$ to construct Φ_n^{APBFS}: If the lower bound of the considered node $n_{(k)}$ is **not greater than** the smallest $f(.)$ of the nodes in Φ_n^{APBFS}, apply *Abstract*(.) to node $n_{(k)}$. If the abstract configuration is identical to a node *m* that is already in the decision tree, then add *m* to Φ_n^{APBFS}; otherwise, add the considered node to Φ_n^{APBFS} and compute the cost-to-go function of the considered node $f(n_{(k)})$ = *Extended APBFS*($n_{(k)}$, *LB-FS*).

Step 4.6. Determine $f(n)$ by taking the minimal value of $f(n_i)$, $n_i \in \Phi_n^{APBFS}$.

Step 4.7. **For each node in Φ_n^{APBFS}, compute its waiting time indicator by Algorithm 2.**

Step 4.8. **Return the node in Φ_n^{APBFS} that has the minimum waiting time indicator as the optimal offspring of node *n*.**

Step 5. If *n* is a RD node, compute $f(n)$ and **return the optimal offspring of *n* following Step 5.1~5.7**.

Step 5.1. Construct Δ_n by considering all feasible relocation moves to retrieve the target container in node *n*.

Step 5.2. Apply *Abstract*(.) to each node n_i in Δ_n to avoid duplicate configurations, which leads to Δ_n'.

Step 5.3. Compute the lower bound $lb(n_i)$ for each node n_i in Δ_n'. Sort the nodes in Δ_n' in non-decreasing order of $lb(.)$.

Step 5.4. Compute $f(n_{(1)})$, wherein $n_{(1)}$ is the node with the smallest lower bound in Δ_n', and add $n_{(1)}$ to Δ_n^{APBFS}.

Step 5.5. Repeat for each of the remaining nodes in Δ_n' in non-decreasing order of $lb(.)$ to construct Δ_n^{APBFS}: If the lower bound of the considered node $n_{(k)}$ is less than the smallest $f(.)$ of the nodes in Δ_n^{APBFS}, apply *Abstract*(.) to node $n_{(k)}$. If the abstract configuration is identical to a node *m* that is already in the decision tree, then add *m* to Δ_n^{APBFS}; otherwise, add the considered node to Δ_n^{APBFS} and compute the cost-to-go function of the considered node $f(n_{(k)})$ = *Extended APBFS*($n_{(k)}$, *LB-FS*).

Step 5.6. Determine $f(n)$ by taking the minimal value of $f(n_i)+r(n)$, $n_i \in \Delta_n^{APBFS}$.

Step 5.7. **Return the node in Δ_n^{APBFS} whose cost-to-go function equals $f(n)$ as the optimal offspring of node *n*.**

2.5. Heuristic solution methods

The (extended) APBFS algorithms are very time-consuming for larger problems. In this section, we propose two efficient heuristic algorithms for the SCRP-FS: the SEM (Sequencing based Expected Minmax) heuristic and the SEML (Sequencing based Expected Minmax with Look-ahead horizon) heuristic. In addition, in order to make the results of the Sooo (extension) model comparable to that of the batch model proposed by Galle et al. (2018b) in terms of the influence of service policies, we extend the EM (Expected Minmax) heuristic used in Galle et al. (2018b) to solve the batch model in the new context with customer preference-based arrivals. From now on, we use the "base model" to refer to the batch model that considers the customer preference-based arrivals. The contributions of our heuristics are summarized below.

First, the EM extension heuristic generalizes the EM heuristic to the SCRP with customer preference information. The main adaptation we made to the EM heuristic is the introduction of the concepts of the Blocking Index (BI) and the Delay Index (DI) that calculate the stack score. The BI and DI are not needed in the EM heuristics and they cannot be easily inferred from the case of equal arrival probability. As presented in Appendix B.1, great effort has been made to calculate the BI and DI, which are used to make a more accurate decision in case of a tie in the context of non-equal arrival probability. This extension is especially useful for the situation of large batch size as the occurrence of ties will be more frequent.

Second, we develop two new fast and efficient heuristics to solve the SCRP-FS – the SEM heuristic and the SEML heuristic. The main ingredients of these two heuristics are: sequencing rule and relocation rule. Regarding the relocation rule, we derive a new blocking index and a new delay index – BIS and DIS, to calculate the stack score by considering the SOOO policy, as shown in Appendix B.3. This generalizes the EM extension heuristic to a more flexible case. In addition, we make a contribution in terms of the sequencing rule, which is an important element of the SCRP-FS. Even though the SEM heuristic uses an intuitive sequencing rule, it has been shown to be effective in the computational experiments. The SEML heuristic further improves the SEM by using a more complex sequencing rule that applies a look-ahead strategy dedicated to performing the most promising retrieval sequence.

We present these three heuristics respectively in the following three sub-sections.

2.5.1 EM extension heuristic

The EM heuristic has been computationally demonstrated to be the equal best heuristic for the batch model. The idea of the EM heuristic is from that of the Min-Max heuristic in the earlier literature (Caserta et al., 2012), which is based on the computation of a stack score that determines which stack a blocking container should be placed. In this paper, the EM extension heuristic is adapted from the EM heuristic to obtain sub-optimal solutions of the base model. Although the EM extension heuristic is not the focus of this study, its description provides a basis for explaining the SEM heuristic and the SEML heuristic that we will design to solve the SCRP-FS.

Before describing the EM extension heuristic, we first briefly introduce the EM heuristic for the batch model. In the batch model, once the truck arrival order of a batch is revealed, the retrieval sequence for this batch is confirmed. The EM heuristic only focuses on the heuristic rules for relocating. Let l_c be the

priority label of container *c* to be relocated, and $m(s)$ be the smallest label of a container in stack *s*, $s \in \{1,...,S\}$. For an empty stack, $m(s)$ is defined as *C*+1. The heuristic rules that determine the storage position of a blocking container *c* from stack *s* are described below.

[Condition 1] There is an available stack $s' \neq s$ such that $m(s') > l_c$.

Let $M = \min_{s' \in \{1,...,S\} \setminus s} \{m(s') : m(s') > l_c\}$. Select the stack that satisfies $m(s') = M$. Break ties by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

[Condition 2] For all stack $s' \neq s$, $m(s') \leq l_c$.

Let $M = \max_{s' \in \{1,...,S\} \setminus s} \{m(s')\}$. Select the stack that satisfies $m(s') = M$. Break ties by choosing from the ones with the minimum number of containers labelled *M*. Further ties are broken by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

The EM heuristic is relatively intuitive. The idea is to minimise the expected number of blocking containers after each relocation move. In Condition 1, *c* can surely avoid being relocated again, and we say a 'good' move is possible. In this condition, EM chooses the stack with the minimum $m(s')$, since the stacks with larger $m(s')$ can be saved as candidate stacks for positioning blocking containers with greater labels. In condition 2, we say a 'good' move is impossible. There are two cases. If $M = l_c$ (which implies that *c* will be relocated again in the future with probability), the stack with the minimum number of containers labelled *M* is chosen, which can minimise the probability of *c* being relocated again. The rationale behind it is that there is an equal chance for any container being the first one to be retrieved among the containers labelled *M*. On the other hand, if $M < l_c$ (which means that *c* will surely be relocated again in the future), the stack with the maximum $m(s')$ is chosen to delay the next relocation of *c* as much as possible. Ties are broken by selecting the stack with the minimum number of containers labelled *M* to delay *c* being relocated again, as there is an equal chance for any container being the first one to be retrieved among containers labelled *M*.

Now we extend the EM as an application to the base model. The EM extension follows the heuristic rule for Condition 1 but applies new rules for Condition 2. The following rules are used in the EM extension for Condition 2:

Let $M = \max_{s' \in \{1,...,S\} \setminus s} \{m(s')\}$. Select the stack that satisfies $m(s') = M$. In case of ties, if $M = l_c$, choose from the ones with the minimum $BI(s')$; if $M < l_c$, choose from the ones with the minimum $DI(s')$. Further ties are broken by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

The main difference between the EM and the EM extension is the way of breaking ties in the case where more than one stack satisfies $m(s') = M$ in Condition 2. With the consideration of customer preference, more accurate criteria are required to break the tie. For this purpose, we introduce two indexes to calculate the stack stores: Blocking Index (BI) and Delay index (DI). The BI of a stack *s*, denoted by $BI(s)$, is defined as the probability of a container being blocking if relocated to *s*. The DI of a stack *s*, denoted by $DI(s)$, is defined as the probability of a container with the smallest label in stack *s* being the first one to be retrieved within its batch. The details of how to calculate the BI and the DI are given in Appendix B.1.

2.5.2 SEM (Sequencing based Expected Minmax) heuristic

The SEM heuristic is proposed to solve the SCRP-FS. Two decisions are to be made by the SEM: sequencing the trucks within the same sub-batch, and relocating the blocking containers. The main idea of the sequencing rule is to avoid as many current relocations as possible. The relocating rule is similar to the EM extension heuristic but new blocking index and delay index are introduced to consider the SOOO policy. In the following, we first introduce the outline of the SEM heuristic and then describe the heuristic rules in detail.

2.5.2.1 Outline of the SEM

In the SEM heuristic, the decision on the retrieval sequence is made one container at a time using a sequencing rule and then the consequent blocking containers are relocated using a relocating rule. The following notations are defined and used throughout Section 2.5.2 and 2.5.3:

t_i: the *i*th target container, $i \in \{1,...,C\}$.

r_c : the number of relocations needed for retrieving container *c*.

X : the bay configuration. Let X_0 represent the initial configuration.

lmin : the smallest label of containers in X .

Θ : the set of containers labelled *lmin* in X .

The general steps of the SEM heuristic are as follows:

Step 0. Let $\mathrm{X} = \mathrm{X}_0$. Set *k*=1 and *i*=1, i.e., the index of the first batch and the index of the first target container.

Step 1. If *k* > *K*, STOP – all containers have been retrieved; otherwise, given X and the truck arrival information of batch *k*, update X by adding the number of containers in the first sub-batch to the labels of the containers in the second sub-batch.

Step 2. Identify *lmin* and construct Θ . If there is only one container in Θ , let this container be t_i ; otherwise, determine t_i and update X accordingly using the ***Sequencing Rule***.

Step 3. Calculate r_{t_i} . If $r_{t_i} = 0$, go to step 4; otherwise, move the r_{t_i} number of blocking containers from top to bottom to the stacks determined by the ***Relocating Rule*** and X is updated as a result.

Step 4. Retrieve t_i from X . If $i = \sum_{j=1}^{k} C_j$, which means all containers in batch *k* have been retrieved, then set *k*=*k*+1 and go to step 1; otherwise, set *i*=*i*+1, go to step 2.

2.5.2.2 Heuristic rules in the SEM

The sequencing rule and the relocation rule are introduced here.

Sequencing rule

The SEM heuristic uses an intuitive sequencing rule, the main idea of which is choosing the container with the least number of blocking containers from the candidate containers.

Step 1. Given X , *lmin* , and Θ , compute the r_c of each container $c \in \Theta$.

Step 2. Sort $\{r_c : c \in \Theta\}$ in non-decreasing order of r_c . Choose the one with the lowest r_c from Θ as the target container t_i , breaking ties arbitrarily.

Step 3. Update X by increasing the labels of the containers in $\Theta \setminus t_i$ by one.

Appendix B.2 provides an example illustrating the above sequencing rule.

Relocating rule

The relocating rule used in the SEM heuristic follows the basic idea of the rule in the EM extension heuristic but uses a new blocking index and a new delay index - BIS (blocking index considering sequencing) and DIS (delay index considering sequencing) - to break ties. This is important because the blocking container in the batch model is not necessarily blocking in the SCRP-FS in which the container retrieval sequence is flexible. Therefore, in order to make correct decisions for the relocation positions, we need new indexes that can take into account the flexible service sequence. The idea behind the BIS is that container *c* being blocking if relocated to stack s' occurs only in the scenario where *c* is in the latter sub-batch and there is at least one container $c_i \in M_{s'}$ in the former sub-batch, where $M_{s'}$ is the set of containers labelled *M* and located in s'. The idea behind the DIS is that a container c_i is surely being the first one to be retrieved in its batch only in the situation that satisfies the following two conditions: 1) c_i is in the former sub-batch; 2) c_i has the lowest number of blocking containers among the containers in the former sub-batch. The details of computing $BIS(s)$ and $DIS(s)$ are given in Appendix B.3.

For the sake of completeness, the relocating rule of the SEM to determine the storage position of a blocking container *c* from stack *s* is presented as follows.

[Condition 1] There is an available stack $s' \neq s$ such that $m(s') > l_c$.

Let $M = \min_{s' \in \{1,\ldots,S\} \setminus s} \{m(s') : m(s') > l_c\}$. Select a stack that satisfies $m(s') = M$. Break ties by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

[Condition 2] For all stack $s' \neq s$, $m(s') \leq l_c$.

Let $M = \max_{s' \in \{1,\ldots,S\} \setminus s} \{m(s')\}$. Select a stack that satisfies $m(s') = M$. In case of ties, if $M = l_c$, choose from the ones with the minimum $BIS(s')$; if $M < l_c$, choose from the ones with the minimum $DIS(s')$. Further ties are broken by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

2.5.3 SEML (Sequencing based Expected Minmax with Look-ahead horizon) heuristic

The SEML improves the SEM by using a sophisticated sequencing rule that applies a look-ahead strategy dedicated to performing the most promising retrieval sequence. Recalling the sequencing rule of the SEM heuristic, in case of tie that more than one container has the lowest number of blocking containers among the containers with the smallest labels, i.e., there is more than one potential target container, the SEM chooses one arbitrarily as the next target container (Step 2 in Section 2.5.2.2). The idea of the SEML heuristic is to break this tie more precisely with look-ahead evaluation. The look-ahead horizon *H* is equal to the number of potential target containers in case of the tie. To be more specific, the SEML first evaluates the contribution of each feasible retrieval sequence of the potential target containers to the total number of relocations, and then, the sequence that contributes least is selected as the actual retrieval sequence of these potential target containers. The contribution is measured by the sum of the number of realised relocations during the retrievals of the potential target containers and the lower bound of the configuration after these retrievals. A new lower bound is proposed with minor modification of the *LB-FS*, because the SEML applies to both SD nodes and RD nodes while the *LB-FS* does not apply to SD nodes. In the new lower bound, the containers with the same priority label whose truck arrival sequence have been revealed are not considered blocking each other. The relocating rule used in the SEML is the same as that in the SEM. The

details of the sequencing rule are presented below, and an illustration is provided in Appendix B.4.

Step 1. Given X, $lmin$, and Θ, compute the r_c of each container $c \in \Theta$.

Step 2. Sort $\{r_c : c \in \Theta\}$ in non-decreasing order of r_c. Construct the set of potential target containers $\{c \mid r_c = \min\{r_c : c \in \Theta\}\}$. Set $H = |\{c \mid r_c = \min\{r_c : c \in \Theta\}\}|$. If H=1, then choose the only potential target container as the target container t_i. Update X by increasing the labels of the containers in $\Theta \setminus t_i$ by one. Otherwise, go to Step 3.

Step 3. Look-ahead evaluation

Step 3.1. Update X by increasing the labels of the non-potential containers in Θ by H.

Step 3.2. Enumerate all feasible retrieval sequences (H! number in total) for the potential target containers.

Step 3.3. Update the labels of the potential target containers according to one feasible retrieval sequence that has not been evaluated and obtain a tentative configuration to be evaluated.

Step 3.4. Given the tentative configuration, retrieve the potential target containers, move the blocking containers according to the relocating rule, and count the number of relocations incurred. Compute the lower bound of the consequent configuration and the contribution. If all retrieval sequences have been evaluated, choose the one with the least contribution as the determined retrieval sequence for the potential target containers, breaking ties arbitrarily. Then, update X according to the determined retrieval sequence of the potential target containers; otherwise, go to Step 3.3.

Step 3.5. The container with $lmin$ is selected as the target container t_i.

2.6 Simulation model

In this section, we develop a discrete-event simulation model to evaluate the effectiveness of the exact algorithms and the heuristics respectively in terms of the two performance metrics: the total number of relocations and the average relevant truck waiting time. Simulation is needed for evaluating heuristics because the solutions of the heuristics depend on the scenario of truck arrivals. The necessity of a simulation model for evaluating exact algorithms is because the exact algorithms to be evaluated (the (extended) APBFS algorithm and the PBFS algorithm) do not record the service completion time for each batch. Hence, in order to evaluate the relevant truck waiting time, we need to simulate the complete retrieval process by using the optimal solutions. To the best of the authors' knowledge, this study is the first one that implements a simulation model to evaluate SCRP's optimal solutions that are derived from in a decision tree. Our main focus in this section is to show how to evaluate the solutions of the exact algorithms by using the developed discrete-event simulation model.

2.6.1 Input and output data

The input data of the simulation model includes: i) the problem instance that consists of the container stacking configuration, the batch information and the customer preference, ii) the truck arrival times, and iii) the handling time per relocation move and the handling time per retrieval move. The direct output for each container/truck includes: i) the number of relocations, ii) the service starting time, and iii) the service completion time. Then, we can output the total number of relocations and the average relevant truck waiting time. By definition, the relevant waiting time of a truck is calculated by: service completion time – service starting time – the handling time per retrieval move. The average relevant waiting time for a sample

is obtained by taking the average over the relevant waiting times of all trucks. In addition, we also output the average delay and the average turn time, which will be explained in Section 2.7.3.2.

2.6.2 Model structure and functions

The simulation model consists of three major programs: a truck generator, an optimiser, and a simulator, which are subsequently described in detail. All programs are implemented in Matlab.

2.6.2.1 Truck generator

The truck generator program creates truck arrival times. Given a problem instance with an initial priority matrix and a customer matrix, N samples of truck arrival times are generated by respecting the appointed time windows and customer preferences. First, the sub-batch of a truck is generated based on the probability given by its customer preference p, such that on expectation each truck is allocated to the first sub-batch for $N*p$ times and the second sub-batch for $N*(1-p)$ times. Second, the sub arrival time window of the truck is generated by using its sub-batch, its appointed time window, and the length of the implemented appointment time window. Last, the specific arrival time of the truck is uniformly generated within its sub arrival time window. To ensure a fair comparison of different algorithms, seed initialized distribution is used. Thus, identical random truck arrival times can be used in simulating different algorithms and the simulation results are repeatable by applying identical problem instances.

2.6.2.2 Optimiser and simulator

The optimiser program generates the decisions on retrieval sequence and relocation positions, which feeds the simulator to perform tasks. The simulator is the core of the simulation model. Its main task is to perform the moves specified by the output of the optimiser, keep track of the state of the container stack, count the number of relocations, and record the time-related performance. The simulation model can evaluate both the exact algorithms and the heuristics but differs in the optimisers and the way the simulators extract the decisions from the output of the optimisers. When evaluating heuristics, the relevant heuristic is used as an optimiser; and the simulator reads both the problem instance and the output of the truck generator. When evaluating the exact algorithms, the exact algorithm is used as an optimiser to produce the optimal solutions; and the simulator reads the problem instance and executes the optimal solutions. Details of the simulation model for evaluating exact algorithms are described below.

The simulation model contains three types of discrete events: revealing the truck arrival information for a batch, relocating a container, and retrieving a container. Given a problem instance, first, the optimiser is invoked, that is, an exact algorithm is executed to obtain the optimal solution. The obtained optimal solution is cached in a tree structure, which we call ‘solution tree’. The simulator reads a truck arrival sample output by the generator and reveals it batch by batch. Once a batch is revealed, the simulator looks up the solution tree to extract the decisions for that batch and performs retrieval moves and relocation moves accordingly. An overview of the architecture of the simulation model is presented in Fig. 2. 6.

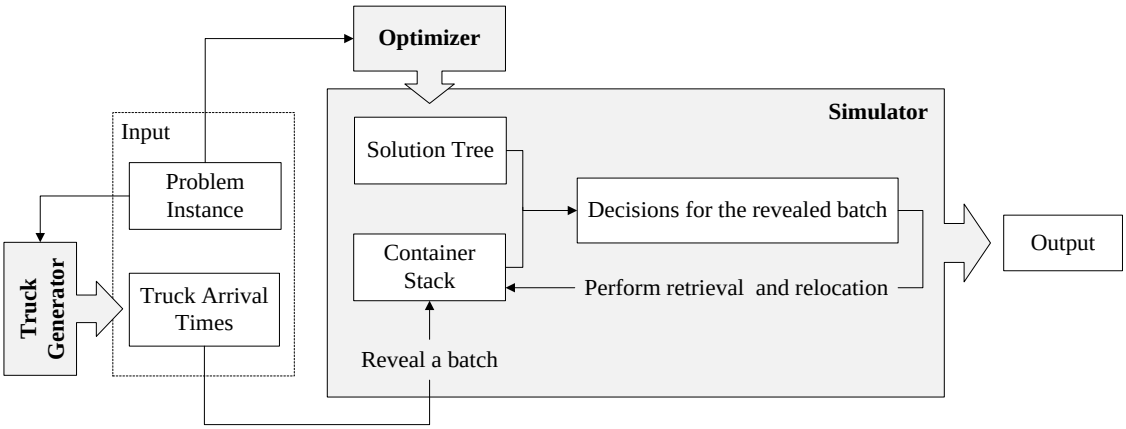


Fig. 2. 6 The architecture of the simulating model for evaluating exact algorithms

We use the example in Fig. 2. A.2 in Appendix A to illustrate the simulation process. Given the problem instance, i.e., the initial node in Fig. 2. A.2, the optimiser is first invoked to generate the solution tree. Then, given a sample of truck arrival times, the simulator reads the sample and reveals it batch by batch. Once a batch is revealed, the simulator looks up the solution tree to identify the SD that matches the revealed container stack and extracts the optimal retrieval sequence for that batch. The decision on the retrieval sequence of a batch is included in the best offspring (a RD node) of the identified SD node. For example, let us consider the scenario in which the truck arrival information of the first batch is revealed as that in node n. Firstly, the optimal retrieval sequence for the first batch is extracted to be the one indicated in node n_2. The simulator then retrieves container 3 and records its service start time and service completion time. Secondly, the container stack is changed to n_3. Notice that the next container to be retrieved (container 4) has a blocking container. When there are blocking containers to be relocated, the simulator looks up the solution tree to identify the optimal relocation positions. The decisions on the relocation positions are obtained by tracing the best offspring (e.g., n_4) of the node in which the blocking containers are located (e.g., n_3). In the considered scenario, the best relocation position for the blocking container above container 4 is identified to be the empty stack. The simulator then relocates the blocking container to the empty stack, retrieve container 4, and records the service starting time and service completion time of container 4. Finally, the container stack is changed to node n_4. After that, the simulator continues to reveal the truck arrival information for the second batch and perform tasks in the same way. It should be noted that after each event, the container stack needs to be abstracted to ensure that it can be matched with one of the nodes in the solution tree.

2.7. Computational experiments

In this section, we test the proposed models and solution methods through enormous numerical experiments using the simulation model introduced in Section 2.6. We present four sets of experiments. Firstly, we test the solving capabilities of the two proposed exact solution algorithms; and we show the improvement of the Sooo extension model over the Sooo model on the relevant truck waiting time. Secondly, we evaluate the effectiveness and the efficiency of the two proposed heuristics by comparing them with the exact solutions of the Sooo extension model; and we compare the performances of the two heuristics to conclude a superior one. Thirdly, we evaluate the effect of the proposed flexible service policy as opposed to the FCFS policy and analyse the impacts of the combinations of different bay layouts and fill rates, average batch sizes and customer preferences on the effect of the flexible service policy. Lastly, we

analyse the influence of customer preference on the Sooo extension model.

All algorithms and simulation models are coded in MATLAB 2018a, partially based on the source code of Galle et al. (2018b) which is available at https://github.com/vgalle/StochasticCRP. All experiments are performed on a desktop with Intel® Core ™ i5-7500 3.40 GHz CPU, 8 GB of RAM, and 64-bit Windows 10 Enterprise. The time limit for running the exact algorithm for each instance is set to one hour (3600 seconds) because some instances are extremely time-consuming.

Our experiment dataset is adapted from the set of CRPTW instances in the literature (Ku and Arthanari, 2016a) which is available at http://crp-timewindow.blogspot.com. The existing instance set is composed of 1440 instances forming 48 classes. The problem classes are characterised by the size ($T\times S$) and the fill rate (μ) of the bay, with T varying from three to six tiers, S varying from five to ten stacks and two μ being considered: 50% and 67%. Given a bay size and a fill rate, the number of containers in the bay is calculated by C = round(μ*T*S), where round(x) rounds x to its closer integer. There are on average two containers per batch, i.e., the average batch size is two. For each such class setting, 30 instances are included, varying in the stacking positions of the containers and the number of containers of each batch. To provide a meaningful interpretation for our model, we consider larger batches with up to on average six containers per batch. The instances of larger batches are obtained by slightly modifying the existing instance set following the method in Galle et al. (2018b), which merges r batches using $w' = \lceil w/r \rceil$, where w is the original batch of a container and w' is its modified batch. As a result, we have instances with small batches (on average 2 containers per batch), large batches (on average 4 containers per batch), and ultra-large batches (on average 6 containers per batch). We use 'the number of tiers (T) – the number of stacks (S) – the fill rate (μ) – the average batch size (B)' to represent our *problem class*. We do not distinguish problem scales accurately because all the relevant factors – T, S, μ, B – have an influence on the computation times of the exact algorithms and the random initial configuration of the container stack also has a great influence. Instead, we consider a problem as a larger problem if it has a larger rate and/or larger batches while other factors (T and S) are the same. Because the instances with ultra-large batches are very hard to be solved optimally, we only use their near-optimal solutions obtained by heuristics to show a positive difference between the FCFS policy and the SOOO policy in Section 2.7.3.2.

Regarding the customer preference, we consider three scenarios of homogenous preference, in which the preferences of all trucks are respectively 0%, 50%, and 100%, and a scenario of heterogeneous preference, in which the preference of each truck is randomly generated and thus differs from each other. The instances with scenario '50%' are referred to as the benchmark set, as they are equivalent to the instance set of the batch model. In this scenario, the probability of each SD node is the same (i.e., 0.25). In the scenario '0%' and '100%', all trucks will arrive at the second sub-time window and the first sub-time window respectively. The truck generator program introduced in Section 2.6.1 is used to generate 1000 samples of truck series for each instance associated with a scenario of customer preference. The appointment time window is set to be 30 minutes. The handling times per relocation move and per retrieval move are calibrated according to the technical capabilities of yard cranes. A Rubber-Tyred Gantry Crane (RTG) can perform 20–25 moves on average per hour, but its realised performance in practice is typically less than 12 moves per hour (Saanen, 2011), which indicates that each move takes on average 2.4 - 5 minutes. Considering that a retrieval move usually takes longer to complete than a relocation move due to the need for coordination with the truck drivers, we set 2 minutes for a relocation move and 4 minutes for a retrieval move.

2.7.1 Performance of the proposed models and exact algorithms

In this section, we evaluate the performances of the exact algorithms, test the tightness of *LB-FS*, and

compare the average relevant truck waiting time between the Sooo model and the Sooo extension model. All results are obtained by simulating optimal solutions.

2.7.1.1 Performances of the exact algorithms

Table 2. 2 shows the results of instances with small batches and the 50% fill rate. The first three columns list the problem class, which is characterised by the number of stacks (*S*), the number of tiers (*T*), and the number of containers (*C*). The average batch size is omitted here as all instances have the same average batch size (i.e., two). Column 'lb' gives the value of lower bound obtained by our proposed lower bound *LB-FS*. Columns five to nine and Columns ten to fourteen respectively report the simulation results of the APBFS algorithm and the extended APBFS algorithm. Column 'Opt' gives the average of the expected total number of relocations over 30 instances, which is the theoretically optimal solution obtained by the exact algorithm. Colum 'Solved' reports the number of instances that the relevant exact algorithm is able to solve to optimality within the time limit (1 hour), where '√' indicates that all 30 instances for a problem class are solved to optimality. Column 'CPU(s)' reports the average computation time for the solvable instances in seconds. Column 'Rel' and column 'AveWait' respectively give the average of the total number of relocations and the average of the relevant truck waiting time over 30 instances, each one based on 1000 samples, which are obtained by simulation.

Table 2. 2 Results of the APBFS and the extended APBFS algorithm for instances with small batches and 50% fill rate

T	*S*	*C*	lb	Sooo - APBFS					Sooo extension - extended APBFS				
				Opt	Solved	CPU(s)	Rel	AveWait (min)	Opt	Solved	CPU(s)	Rel	AveWait (min)
3	5	8	1.454	1.478	√	0.02	1.478	3.319	1.478	√	0.03	1.478	3.207
	6	9	1.558	1.582	√	0.02	1.581	3.428	1.582	√	0.03	1.581	3.303
	7	11	2.608	2.654	√	0.02	2.654	3.485	2.654	√	0.03	2.654	3.356
	8	12	2.163	2.169	√	0.01	2.169	3.091	2.169	√	0.03	2.169	3.014
	9	14	2.875	2.884	√	0.02	2.885	3.226	2.884	√	0.04	2.885	3.106
	10	15	3.092	3.094	√	0.03	3.093	3.044	3.094	√	0.06	3.093	2.930
4	5	10	2.725	2.856	√	0.02	2.855	3.534	2.856	√	0.03	2.855	3.411
	6	12	3.371	3.461	√	0.03	3.466	3.532	3.461	√	0.05	3.466	3.351
	7	14	3.817	3.944	√	0.04	3.941	3.485	3.944	√	0.05	3.941	3.324
	8	16	4.475	4.555	√	0.16	4.559	3.556	4.555	√	0.20	4.559	3.390
	9	18	5.467	5.526	√	0.29	5.523	3.691	5.526	√	0.34	5.523	3.474
	10	20	5.983	6.016	√	0.72	6.015	3.334	6.016	√	0.80	6.015	3.181
5	5	13	4.371	4.883	√	0.16	4.883	4.042	4.883	√	0.19	4.884	3.900
	6	15	5.138	5.546	√	2.44	5.544	3.708	5.546	√	2.98	5.544	3.553
	7	18	6.246	6.575	√	0.72	6.573	3.993	6.575	√	0.77	6.573	3.777
	8	20	7.017	7.519	√	7.57	7.516	3.695	7.519	√	8.16	7.516	3.482
	9	23	8.358	8.699	29	67.35	8.696	3.835	8.699	29	68.25	8.696	3.606
	10	25	8.883	9.237	29	39.40	9.238	3.719	9.237	29	49.37	9.238	3.517
6	5	15	5.975	7.004	√	4.56	6.999	4.034	7.004	√	5.06	6.999	3.886
	6	18	6.900	7.729	√	5.48	7.728	4.162	7.729	√	6.18	7.728	3.959
	7	21	8.575	8.925	23	294.96	8.991	4.026	8.925	22	160.87	8.923	3.787
	8	24	9.258	9.886	22	150.07	9.881	3.998	9.886	22	164.07	9.882	3.748
	9	27	10.275	10.538	18	100.24	10.538	3.661	10.538	18	104.47	10.538	3.419
	10	30	11.692	11.576	19	285.31	11.599	3.785	11.576	18	108.52	11.576	3.580

*Note: customer preference scenario: 50%

From Table 2. 2, we can see that the solution capacity of the two algorithms is quite similar. Both of them are capable of solving all the instances with T= 3 and T=4 in less than one second and 98.9% (178/180) of the instances with T=5 within the time limit. As T increases to six, some instances are extremely time-consuming. We call the instances that cannot be solved within one hour 'hard instances'. The number of hard instances for each problem class is basically the same as that for the PBFS algorithms, which indicates that our proposed exact algorithms are effective for the SCRP-FS. We observe that the hard instances for the two proposed algorithms are the same except the classes of (T=6, S=7) and (T=6, S=10). This is because the search rules used in the two algorithms are basically the same. The only difference is that in the extended APBFS algorithm the nodes that perform equally in terms of the primary objective are further explored in order to find the node that is optimal to the secondary objective. This leads to that the extended APBFS algorithm requires a longer CPU time to prove optimality, which can be observed from the CPU columns. It should be pointed out that for the problem classes of (T=6, S=7) and (T=6, S=10), the CPU times for Sooo extension are much shorter than that of Sooo. This is because the Sooo model is able to solve one more instance than the Sooo extension model within the allowed computational time limit (i.e. 3600 seconds), and this extra instance is too time-consuming to solve for the Sooo extension model. A fair comparison of two models can be referred to Section 2.7.1.2 and Table 2. C.4. Due to the unavailability of the optimal solutions for hard instances, these hard instances are excluded from the simulation. For the problem class that includes any hard instance, Table 2. 2 only reports the average over the solved instances in columns 'CPU', 'Rel' and 'AveWait'. In addition, as expected, the optimal solutions in terms of the total expected number of relocations (Opt) of the two models are the same. Besides, the gap between the 'Opt' and the 'Rel' in both models is insignificant, which is within [-0.08%, 0.09%], indicating that our samples are large enough to approximate the actual values.

The results of the extended APBFS algorithm for larger instances are given in Appendix C.1. From Table 2. 2 and Appendix C.1, we can conclude that the extended APBFS can solve 87.5% (42/48) of the instances with T=3,4 within 30 seconds. In the tables in Appendix C.1, lb* represents the calibrated lower bound, which takes the average of the lb of the instances that are solved optimally. By comparing lb*and Opt, we can find that the relative difference between the lower bound and the optimal solution for instances with higher stacks (larger T) is greater than that with lower stacks (smaller T). This can be explained by the fact that the chance of a container being relocated more than once in a bay with higher stacks is greater than that in a bay with lower stacks. Since our lower bound only counts the number of blocking containers that are relocated at least once, it is tighter for lower bays. For all the instances with lower bays (T=3, 4) in Table 2. 2 and Appendix C.1, our lower bound is within 13.11% of the optimal solution, and in about 73% (35/48) cases our lower bound is very close to the optimal solution with a gap within 5%. Since in many container terminals, laden containers are stacked up to four tiers due to safety issues and efficiency considerations, our lower bound can efficiently evaluate the least number of relocations needed to empty a bay, which could help to determine a favorable container stacking configuration.

Remark: it is observed that the CPU time deviates greatly for different instances even their problem classes are the same. After a closer check, we find that the initial configuration of the container stack has a great influence on the solution computational efficiency.

2.7.1.2 Comparison of the Sooo model and the Sooo extension model

For a fair comparison of the relevant truck waiting time between the two models, we calibrate the results of 'CPU', 'Rel' and 'AveWait' in Table 2. 2 to ensure that only the instances that are solved optimally by both algorithms are included into the comparison. The calibrated results are presented in italic in Table 2. C.4 of Appendix C.2. Besides, we compare the results of the two models for instances with a 67% fill rate,

which are shown in Table 2. 3 and Table 2. 4 (the numbers in italic represent the calibrated results). Comparing the 'Rel' of the two models, it is found that the simulated total number of relocations of the two models are the same in most cases, with only five problem class occurring a difference of 0.001 number of relocations. The occurrence of this difference seems counterintuitive, but it might happen only because we are sampling. However, as the difference is quite trivial, it is fair to compare the 'AveWait' of the two models based on our samples. Column 'Gap[AveWait]' reports the gap between the average relevant truck waiting time of the two models, which represents the benefits of taking into account the truck waiting time.

Comparing Table 2. 3 with Table 2. C.4, we can find that the gaps in the average relevant truck waiting time between the two models are more significant for instances with a larger fill rate. Besides, in a comparison between Table 2. 3 and Table 2. 4, it can be found that the instances with more concentrated truck arrival patterns, that is, '0%' customer preference scenario, benefit more from the Sooo extension model. In addition, the results of small batches and large batches indicate that the batch size does not have a significant influence on the relative difference of the relevant truck waiting time between the two models. From Table 2. C.4 and Tables 2.3-2.4, we can conclude that the reduction in the average relevant truck waiting time in the Sooo extension model over the Sooo model is between 2.5% and 11%. Moreover, the CPU results confirm that the extended APBFS algorithm takes a longer time to obtain the optimal solution than the APBFS algorithm.

Table 2. 3 Comparison between the Sooo model and the Sooo extension model for instances with 67% fill rate and '50%' customer preference scenario

T	*S*	*C*	Sooo				Sooo extension				Gap [AveWait]
			Solved	CPU(s)	Rel	AveWait (min)	Solved	CPU(s)	Rel	AveWait (min)	
Small batches											
3	5	10	√	0.04	2.786	3.387	√	0.05	2.786	3.224	4.81%
	6	12	√	0.04	3.620	3.710	√	0.05	3.620	3.524	5.01%
	7	14	√	0.03	3.759	3.655	√	0.06	3.759	3.506	4.08%
	8	16	√	0.05	4.382	3.577	√	0.08	4.382	3.385	5.37%
	9	18	√	0.07	4.816	3.580	√	0.11	4.816	3.408	4.80%
	10	20	√	0.07	5.066	3.327	√	0.11	5.066	3.176	4.54%
4	5	13	√	0.11	5.071	4.130	√	0.14	5.071	3.976	3.73%
	6	16	√	1.10	6.931	4.109	√	1.19	6.931	3.898	5.14%
	7	19	√	1.01	6.929	3.646	√	1.16	6.929	3.465	4.96%
	8	21	√	13.98	7.969	3.829	√	20.25	7.969	3.601	5.95%
	9	24	√	11.08	9.257	3.825	√	11.80	9.257	3.619	5.39%
	10	27	27	69.81	9.622	3.620	27	92.17	9.622	3.426	5.36%
Large batches											
3	5	10	√	0.51	2.403	7.208	√	0.90	2.403	6.945	3.65%
	6	12	√	0.58	3.245	7.804	√	1.21	3.245	7.468	4.31%
	7	14	√	1.41	3.518	8.119	√	3.58	3.518	7.720	4.91%
	8	16	√	1.13	4.180	7.671	√	3.36	4.179	7.281	5.08%
	9	18	√	1.46	4.484	7.680	√	4.05	4.485	7.349	4.31%
	10	20	√	0.93	4.755	7.338	√	3.40	4.755	6.937	5.46%
4	5	13	√	8.60	4.632	8.618	√	27.60	4.631	8.342	3.20%
	6	16	28	13.31	6.167	8.502	28	39.35	6.167	8.091	4.83%
	7	19	28	36.99	6.307	7.681	28	175.06	6.307	7.310	4.83%
	8	21	28	*12.32*	*7.027*	*8.036*	26	73.15	7.027	7.580	5.67%

	9	24	27	*71.98*	*8.208*	*8.086*	24	320.01	8.208	7.655	5.33%
	10	27	23	*153.87*	*8.898*	*7.673*	21	441.13	8.898	7.233	5.73%

Table 2. 4 Comparison between the Sooo model and the Sooo extension model for instances with 67% fill rate and '0%' customer preference scenario

T	*S*	*C*	Sooo				Sooo extension				Gap [AveWait]
			Solved	CPU(s)	Rel	AveWait (min)	Solved	CPU(s)	Rel	AveWait (min)	
Small batches											
3	5	10	√	0.03	2.500	3.307	√	0.04	2.500	2.993	9.48%
	6	12	√	0.03	3.367	3.672	√	0.04	3.367	3.289	10.44%
	7	14	√	0.03	3.567	3.595	√	0.05	3.567	3.329	7.42%
	8	16	√	0.05	4.100	3.546	√	0.07	4.100	3.171	10.58%
	9	18	√	0.07	4.533	3.533	√	0.10	4.533	3.204	9.33%
	10	20	√	0.07	4.867	3.320	√	0.10	4.867	3.023	8.94%
4	5	13	√	0.06	4.600	3.944	√	0.09	4.600	3.667	7.02%
	6	16	√	0.18	6.600	4.067	√	0.21	6.600	3.633	10.66%
	7	19	√	0.20	6.567	3.589	√	0.25	6.567	3.239	9.77%
	8	21	√	1.06	7.633	3.803	√	1.31	7.633	3.384	11.02%
	9	24	√	2.96	8.967	3.836	√	3.26	8.967	3.422	10.79%
	10	27	29	7.71	9.379	3.630	29	8.56	9.379	3.246	10.56%
Large batches											
3	5	10	√	0.46	1.800	6.887	√	0.83	1.800	6.327	8.13%
	6	12	√	0.50	2.700	7.683	√	1.07	2.700	6.861	10.70%
	7	14	√	1.23	2.967	7.990	√	3.48	2.967	7.152	10.49%
	8	16	√	0.94	3.533	7.471	√	2.33	3.533	6.704	10.26%
	9	18	√	1.08	3.833	7.452	√	2.88	3.833	6.852	8.05%
	10	20	√	0.82	4.233	7.287	√	2.99	4.233	6.503	10.75%
4	5	13	√	1.82	3.667	8.103	√	6.19	3.667	7.441	8.16%
	6	16	√	8.28	5.467	8.283	√	132.93	5.467	7.433	10.26%
	7	19	√	10.48	5.533	7.484	√	173.77	5.533	6.789	9.28%
	8	21	√	*9.28*	*6.214*	*7.762*	28	86.46	6.214	6.986	9.99%
	9	24	√	25.06	7.733	8.042	√	331.17	7.733	7.200	10.47%
	10	27	29	*27.63*	*8.429*	*7.569*	28	84.01	8.429	6.788	10.31%

2.7.2 Effectiveness of the proposed heuristics

In this section, we evaluate the effectiveness of the heuristic algorithms. First, we compare the results of the proposed two heuristics with that of the extended APBFS algorithm. Second, we compare the performances of the two heuristics.

2.7.2.1 Comparison of the exact solutions and heuristic solutions

Table 2. 5 compares the SEM heuristic and the SEML heuristic with that of the extended APBFS algorithm on instances with small batches and a 50% fill rate. For a fair comparison, the heuristic results are calibrated (in italic) to ensure that the comparison is based on the instances that are solved to optimality by the extended APBFS algorithm. Because the CPU times of both heuristics are less than 1 second, they are not presented here. The best heuristic result for each problem class is highlighted in bold, from which it can be observed that in almost all cases the SEML heuristic outperforms the SEM heuristic. In addition,

from the column Gap[Rel], we can see that for instances with T=3,4, our proposed SEML heuristic performs quite well, with only at most 0.64% difference from the optimal solutions in terms of the total number of relocations. Overall, the SEML heuristic is at most 3.41% more than the optimal total number of relocations. Regarding the average relevant truck waiting time (Gap[AveWait]), the result of the SEML heuristic is at most 1.49% more than that of the extended APBFS algorithm. Besides, occasionally, the SEML heuristic even outperforms the extended APBFS algorithm slightly in terms of the relevant truck waiting time. This is not surprising, because the Sooo extension model aims to minimise the total waiting times of each batch sequentially rather than minimising the total waiting times of all the trucks. Because the total waiting time of all trucks is jointly determined by the number of relocations of and the service sequence of each truck, the solutions with the same total number of relocations may lead to different total waiting time and it might also happen that the solutions with more relocations lead to less total waiting time.

Table 2. 5 Comparison of the extended APBFS algorithm, SEM and SEML heuristics for small batches and 50% fill rate

T	S	C	Extended APBFS		SEM				SEML			
			Rel	AveWait	Rel	AveWait	Gap[Rel]	Gap[AveWait]	Rel	AveWait	Gap[Rel]	Gap[AveWait]
3	5	8	1.478	3.207	**1.478**	**3.208**	0.00%	0.02%	**1.478**	**3.208**	0.00%	0.02%
	6	9	1.581	3.303	1.582	**3.303**	0.07%	0.02%	**1.581**	**3.303**	0.00%	0.00%
	7	11	2.654	3.356	**2.654**	**3.359**	0.00%	0.08%	**2.654**	**3.359**	0.00%	0.08%
	8	12	2.169	3.014	2.174	3.015	0.22%	0.04%	**2.169**	**3.014**	0.00%	0.01%
	9	14	2.885	3.106	**2.886**	**3.106**	0.05%	0.01%	**2.886**	**3.106**	0.03%	0.01%
	10	15	3.093	2.930	**3.100**	**2.931**	0.22%	0.04%	**3.100**	**2.931**	0.22%	0.04%
4	5	10	2.855	3.411	2.891	3.408	1.26%	-0.09%	**2.874**	**3.403**	0.64%	-0.22%
	6	12	3.466	3.351	3.481	3.356	0.44%	0.15%	**3.471**	**3.353**	0.14%	0.06%
	7	14	3.941	3.324	3.986	3.332	1.15%	0.25%	**3.960**	**3.327**	0.49%	0.10%
	8	16	4.559	3.390	**4.578**	**3.389**	0.42%	-0.02%	**4.578**	**3.389**	0.42%	-0.02%
	9	18	5.523	3.474	**5.532**	**3.471**	0.15%	-0.09%	**5.532**	3.472	0.15%	-0.06%
	10	20	6.015	3.181	6.018	**3.183**	0.06%	0.06%	**6.016**	**3.183**	0.01%	0.06%
5	5	13	4.884	3.900	5.047	3.921	3.36%	0.55%	**5.031**	**3.920**	3.02%	0.51%
	6	15	5.544	3.553	5.633	3.558	1.61%	0.16%	**5.619**	**3.554**	1.35%	0.04%
	7	18	6.573	3.777	6.631	3.790	0.88%	0.33%	**6.619**	**3.785**	0.70%	0.20%
	8	20	7.516	3.482	7.636	3.500	1.60%	0.52%	**7.619**	**3.497**	1.37%	0.43%
	9	23	8.696	3.606	*8.733*	*3.610*	0.42%	0.11%	***8.710***	***3.607***	0.16%	0.04%
	10	25	9.238	3.517	*9.301*	*3.518*	0.68%	0.03%	***9.285***	***3.515***	0.50%	-0.05%
6	5	15	6.999	3.886	**7.237**	**3.944**	3.40%	1.48%	7.238	**3.944**	3.41%	1.49%
	6	18	7.728	3.959	7.918	3.964	2.47%	0.13%	**7.913**	**3.963**	2.40%	0.11%
	7	21	8.923	3.787	***9.011***	***3.772***	0.98%	-0.40%	***9.011***	***3.772***	0.98%	-0.40%
	8	24	9.882	3.748	***9.987***	***3.755***	1.06%	0.16%	*9.994*	***3.755***	1.13%	0.18%
	9	27	10.538	3.419	***10.652***	***3.429***	1.08%	0.28%	***10.652***	***3.429***	1.08%	0.28%
	10	30	11.572	3.580	***11.579***	***3.578***	0.06%	-0.06%	***11.579***	***3.578***	0.06%	-0.06%

*Note: customer preference scenario: 50%

We compare the SEML heuristic with the extended APBFS algorithm on larger instances in Appendix C.1. The comparisons are based on the instances that can be solved optimally by both the exact algorithm and the heuristic algorithm. It can be seen that for the problem classes in Table 2. 5 and in Appendix C.1 for which we have access to the optimal solutions of all the 30 instances, the maximum gaps for the total number of relocations and the average relevant truck waiting time are 4.28% and 1.49% respectively; and

in about 84% (41/49) cases, the total number of relocations obtained by the SEML heuristic is very close to the optimal solutions with gaps no more than 2%. Besides, for the problem classes for which we only have access to the optimal solutions of part of the 30 instances, the maximum gap for the total number of relocations and the average relevant truck waiting time are 9.80% and 1.53% respectively. With an enormous number of instances in a range of sizes being evaluated, our experiments show strong evidence that the SEML heuristic is a good solution to the SCRP-FS of practical sizes.

2.7.2.2 Comparison of the two heuristics

Furthermore, we compare the performance between the SME heuristic and the SMEL heuristic on all the instances with a 67% fill rate. Appendix D displays the gaps between the two heuristics for instances with three to six tiers respectively. The horizontal axis presents the characteristics of each instance: the customer preference scenario (*P*), the average batch size (*B*), and the number of stacks (*S*). Gap[Rel]=(SEM[Rel]–SEML[Rel])/SEM[Rel]×100%, and Gap[AveWait]= (SEM[AveWait]–SEML[AveWait])/SEM[AveWait]×100%. In most cases, the SEML heuristic shows superior performance on both measures to that of SEM heuristic, which confirms the importance of looking ahead on the decision making of retrieval sequence. Although in very few cases the good performance on 'Rel' of the SEML heuristic is at the expense of 'AveWait', the increases on 'AveWait' are no more than 1% compared with SEM. Given the better performance quality of the SEML heuristic, we use SEML as the heuristic solver for the SCRP-FS in the remaining experiments.

2.7.3 Effect of the flexible service policy

In this section, we first verify the effectiveness of our proposed flexible service policy by comparing the Sooo extension model with the base model. Then, various instances with different bay sizes and fill rates, batch sizes and customer preference scenarios are tested to investigate their impacts on the effect of the flexible service policy.

2.7.3.1 Comparison of the base model and the Sooo extension model on the benchmark

In order to evaluate the effect of the proposed flexible service policy as opposed to the FCFS policy, we compare the results of the Sooo extension model with the base model on the benchmark set. The benchmark set consists of the instances with a 50% fill rate, small batches, and the '50%' customer preference scenario. In order to obtain the results of the base model, we slightly adapt the PBFS algorithm in Galle et al. (2018b) by using a new lower bound that incorporates the characteristics of customer preference, which is similar to the idea of computing the BI in the EM extension algorithm. Table 2. 6 reports the calibrated results for comparison. Gap[Rel]=(Base model[Rel]–Sooo extension model[Rel])/Base model[Rel]×100%, and Gap[AveWait]=(Base model[AveWait]– Sooo extension model[AveWait])/Base model[AveWait]×100%. We can see that around 2% - 13% reduction in the total number of relocations can be achieved by the Sooo extension model compared with the base model on the benchmark set. The effectiveness of the flexible policy is also demonstrated by the around 4.3% - 8.4% reduction in the average relevant truck waiting time.

Table 2. 6 Comparison between the base model and the Sooo extension model on the benchmark instance set

T	*S*	*C*	Base model*					Sooo extension model					Gap	
			Opt	Solved	CPU(s)	Rel	AveWait (min)	Opt	Solved	CPU(s)	Rel	AveWait (min)	Gap[Rel]	Gap [AveWait]
3	5	8	1.703	√	0.02	1.700	3.436	1.478	√	0.03	1.478	3.207	13.03%	6.66%
	6	9	1.739	√	0.01	1.737	3.487	1.582	√	0.03	1.581	3.303	9.00%	5.28%
	7	11	2.878	√	0.02	2.878	3.544	2.654	√	0.03	2.654	3.356	7.79%	5.30%
	8	12	2.308	√	0.02	2.307	3.148	2.169	√	0.03	2.169	3.014	5.95%	4.26%
	9	14	3.004	√	0.02	3.006	3.235	2.884	√	0.04	2.885	3.106	4.02%	3.98%
	10	15	3.192	√	0.02	3.193	3.066	3.094	√	0.06	3.093	2.930	3.14%	4.45%
4	5	10	3.108	√	0.02	3.107	3.657	2.856	√	0.03	2.855	3.411	8.08%	6.74%
	6	12	3.675	√	0.03	3.676	3.559	3.461	√	0.05	3.466	3.351	5.70%	5.85%
	7	14	4.164	√	0.03	4.164	3.522	3.944	√	0.05	3.941	3.324	5.37%	5.62%
	8	16	4.819	√	0.16	4.820	3.592	4.555	√	0.20	4.559	3.390	5.42%	5.63%
	9	18	5.730	√	0.33	5.729	3.670	5.526	√	0.34	5.523	3.474	3.58%	5.35%
	10	20	6.275	√	0.75	6.272	3.353	6.016	√	0.80	6.015	3.181	4.10%	5.13%
5	5	13	5.323	√	0.15	5.325	4.198	4.883	√	0.19	4.884	3.900	8.29%	7.09%
	6	15	5.911	√	4.38	5.914	3.813	5.546	√	2.98	5.544	3.553	6.25%	6.83%
	7	18	6.965	√	1.11	6.969	4.046	6.575	√	0.77	6.573	3.777	5.68%	6.64%
	8	20	7.847	√	7.85	7.846	3.703	7.519	√	8.16	7.516	3.482	4.21%	5.98%
	9	23	8.999	28	70.96	9.005	3.829	*8.704*	29	*70.61*	*8.701*	*3.609*	3.38%	5.76%
	10	25	9.547	29	47.92	9.547	3.716	9.237	29	49.37	9.238	3.517	3.24%	5.34%
6	5	15	7.595	√	4.98	7.590	4.244	7.004	√	5.06	6.999	3.886	7.79%	8.43%
	6	18	8.232	√	17.45	8.231	4.249	7.729	√	6.18	7.728	3.959	6.11%	6.82%
	7	21	*9.394*	23	*251.90*	*9.393*	*4.077*	8.925	22	160.87	8.923	3.787	5.00%	7.11%
	8	24	10.318	22	134.48	10.313	3.989	9.886	22	164.07	9.882	3.748	4.18%	6.03%
	9	27	10.713	18	103.13	10.714	3.609	10.538	18	104.47	10.538	3.419	1.64%	5.26%
	10	30	11.804	17	234.62	11.809	3.774	*11.551*	18	*114.81*	*11.547*	*3.583*	2.22%	5.06%

*Note: The base model refers to the batch model of Galle et al. (2018b) in the new context of customer preference-based arrivals.

2.7.3.2 Effect of the flexible service policy in different scenarios

Based on all instances (including the instances with ultra-large batches), we analyse the impacts of the combinations of different bay sizes (*T***S*) and fill rates (μ), truck appointment patterns (the average batch size) and truck arrival behaviours (the customer preference scenario) on the effects of the flexible service policy. The results of all instances are obtained by simulating heuristic solutions except the results of the benchmark set which are from the optimal solutions in Table 2. 6.

Effect on the number of relocations

Fig. 2. 7 depicts the relative reduction in the total number of relocations. In each figure, six plots are presented, varying in the average batch size in the horizontal direction and the customer preference scenario in the vertical direction. Note that because the relative reductions for the '0%' and '100%' customer preference scenarios are the same, we only present the result of one scenario in the vertical direction. As shown in Fig. 2. 7, the effect on relocation reduction is more significant for the cases with larger batch sizes and the cases with more concentrated truck arrivals within the appointed time window (i.e., the '0%' customer preference scenario). The reason is that these cases provide more opportunities for out-of-order retrievals to reduce relocations as there are more trucks in the same sub-batch. Note that under the cases where the customer preference scenario is '0%', the SCRP-FS is equivalent to the deterministic CRP with flexible service policies (CRP-FS) in which all the trucks in the same batch are allowed to be

retrieved out-of-order. The effect of the flexible service policy is maximised in the context of the CRP-FS as the container retrieval order has the greatest flexibility and meanwhile, the truck arrival uncertainties are completely offset.

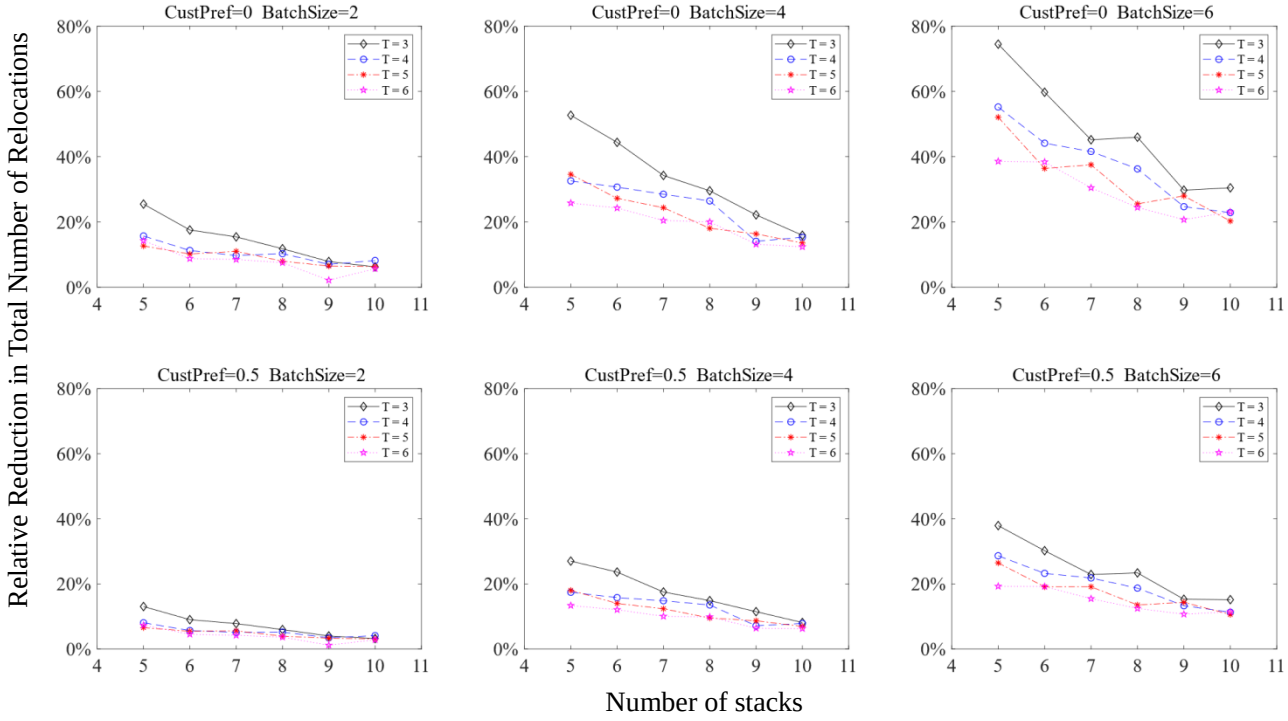


Fig. 2. 7(a) Effect of the flexible service policy on the total number of relocations for instances of 50% fill rate

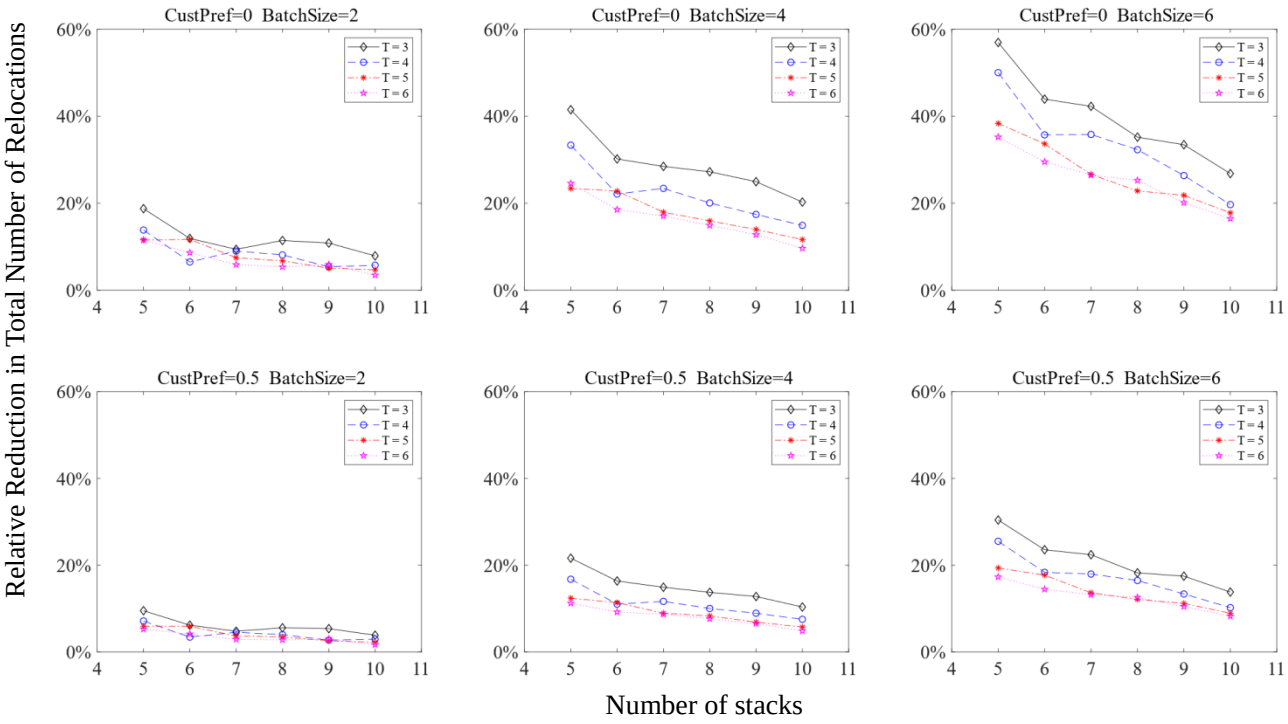


Fig. 2. 7(b) Effect of the flexible service policy on the total number of relocations for instances of 67% fill

rate

Furthermore, we can find that the relative reduction in the number of relocations depends on the bay size (T*S) and the fill rate (μ). In general, the percentage is decreasing as the bay size and the fill rate get larger. To understand this, let us consider the benefits of the flexible service policy. For each out-of-order retrieval, the direct benefit is avoiding one relocation, and the indirect benefit is avoiding future relocations that might be caused by the blocking container if it is not retrieved out-of-order. As T and μ increases, the likelihood of blocking become greater, but the increasing number of blocking containers cannot be offset completely by implementing the proposed flexible service policy as only the containers in the same sub-batch are allowed to be retrieved out-of-order. In addition, as S increases, it is more likely that a better relocating stack can be found for a relocated container, meaning that the relocated container being blocking again in the future is less likely to occur, and thus the benefit of out-of-order retrieval is diminishing. This indicates that the bay of smaller size and sparse stacking can benefit more from the flexible service policy. For the instances with on average six trucks per batch and the '50%' customer preference scenario, the peak relative reduction on the number of relocations is around 38% and 30% respectively for the bay of 50% and 67% fill rate. This leads to a 9.6% and 11.3% reduction in the average relevant waiting time respectively for the bay of 50% and 67% fill rate (see Appendix E.3).

Effect on the trucks waiting time

We also report the absolute reduction in the two performances. Note that the application of the flexible policy always leads to positive reductions, we use "absolute reduction" only to differentiate it from "relative reduction". The absolute reduction on the average relevant truck waiting times shows a similar pattern as that on the total number of relocations (see Appendix E.1 and Appendix E.2). However, in contrast to Fig. 2. 7, the bay's height and fill rate has a positive impact on the relative reduction in the average relevant truck waiting time (see Appendix E.3). This is because the total relevant truck waiting times include a fixed amount of time that is not influenced by the service policy. Recalling Section 2.3.2.2, for each batch k, no matter what the solution is, we have to add $t^{ret} \cdot \sum_{j=1}^{C_k} (C_k - j)$ to the total relevant waiting times, which is a fixed value. To obtain an accurate understanding of the effect of the flexible service policy on reducing trucks' waiting times, we deduct this fixed amount of time from the total relevant waiting times and then take the average, resulting in a new average waiting time. To differentiate, we call it average delay time. The average delay time represents the waiting time caused to each truck only due to relocation operations.

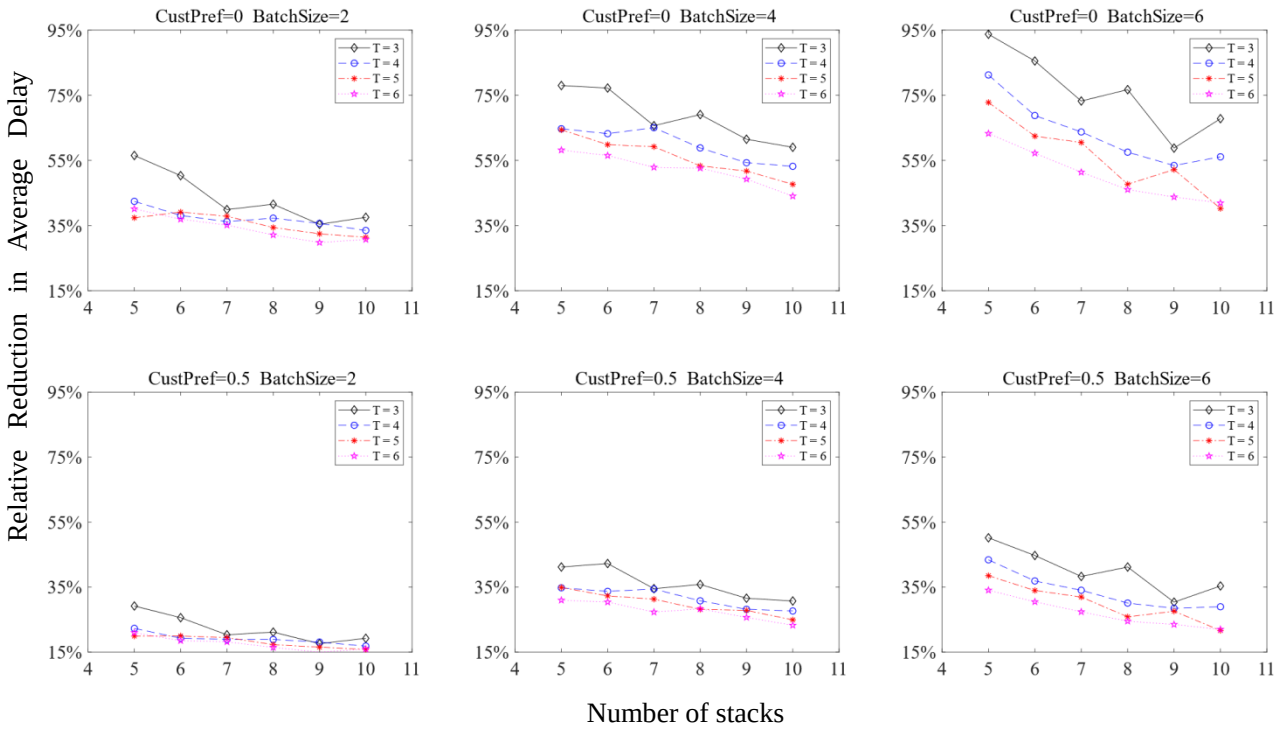


Fig. 2. 8(a) Effect of the flexible service policy on average delay time for instances of 50% fill rate

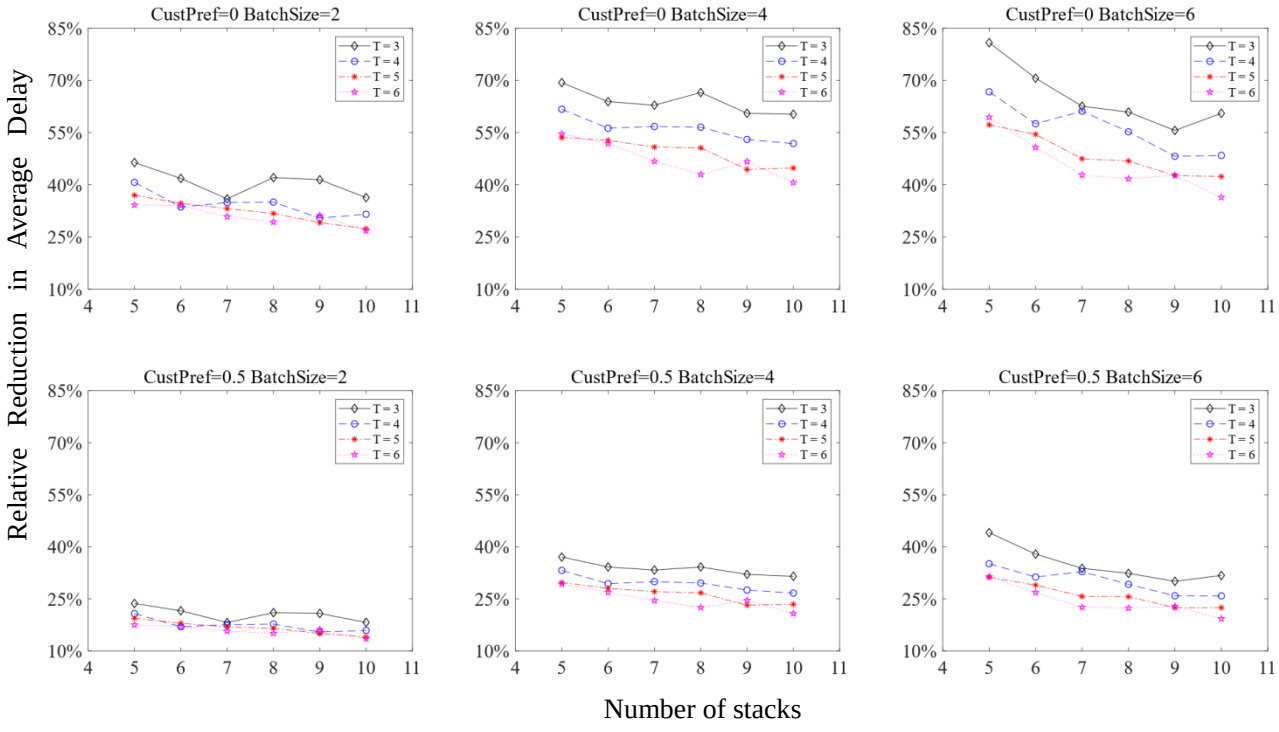


Fig. 2. 8(b) Effect of the flexible service policy on average delay time for instances of 67% fill rate

Fig. 2. 8 depicts the relative reduction in the average delay time. The similar trend between Fig. 2. 7 and Fig. 2. 8 indicates that the reduction in the number of relocations plays a direct role in reducing the average delay time. For the instances with ultra-large batches and the customer preference scenario being '50%',

about 50% and 44% of the average delay time can be reduced respectively for the bay of 50% and 67% fill rate as a result of out-of-order retrievals. The experiment results also demonstrate (not presented in the figure) that on average one reduction in the number of relocations results in 1.07 minutes and 0.93 minutes reduction in the average relevant truck waiting time across all instances respectively for the bay of 50% and 67% fill rate.

Moreover, we also measure the average turn time under the flexible service policy, which is shown in Appendix E.4. The turn time of a truck is defined as the elapse of time between its arrival time and its retrieval service completion time. Appendix E.4 shows an average difference of 15-minutes in the average turn time between the '0%' customer preference scenario and the '100%' customer preference scenario. This is only due to the difference between the truck arrival times that are generated for the two scenarios. Noticing that our appointment time window is set to be 30 minutes, the 15-minutes difference validates our simulation results.

Effect on the service equity

Out-of-order retrievals might make some trucks perceive unfair service due to the adjustment of the service sequence. To examine the equity of truck service, we use box plots to display the distributions of the truck turn time under the FCFS policy and the flexible policy respectively, which is contrasted in Fig. 2. 9. It can be observed that the maximum values of the truck turn time (among all trucks' turn times including the outliers) under the flexible service policy are generally greater than that under the FCFS policy. This is not surprising because the flexible service policy makes some trucks that arrive earlier being served at a later time due to the sequencing decision. However, because we restrict the out-of-order retrievals within the same sub-group, the trucks arriving in the first sub-window will always be serviced before the trucks arriving in the second sub-window, which means the service equity between two sub-groups of trucks is maintained. It can be seen that the difference of the maximum turn times between two policies is only about five minutes among the cases in Fig. 2. 9. Besides, the differences are not obvious for the cases with higher tiers (T = 5, 6), and in some cases, the flexible policy even has a shorter maximum turn time. The reason is that the instances with higher tiers require a higher average number of relocations to retrieve a container, while the flexible policy can significantly reduce the number of relocations and avoid the long waiting time compared to the FCFC service. Moreover, the flexible policy has a lower minimum value of the truck turn time; and more importantly, the median and the mean of the trucks' turn time under the flexible policy are always smaller than those under the FCFS policy.

These results demonstrate that when the FCFS policy is replaced by the flexible policy, although some trucks may experience a little longer turn time, on average the service each truck receives can be improved. This goal is consistent with most of the existing relevant literature, e.g., minimising the average waiting time (Borjian et al., 2015b; Zeng et al., 2019) or minimising total delay times (Borjian et al., 2013).

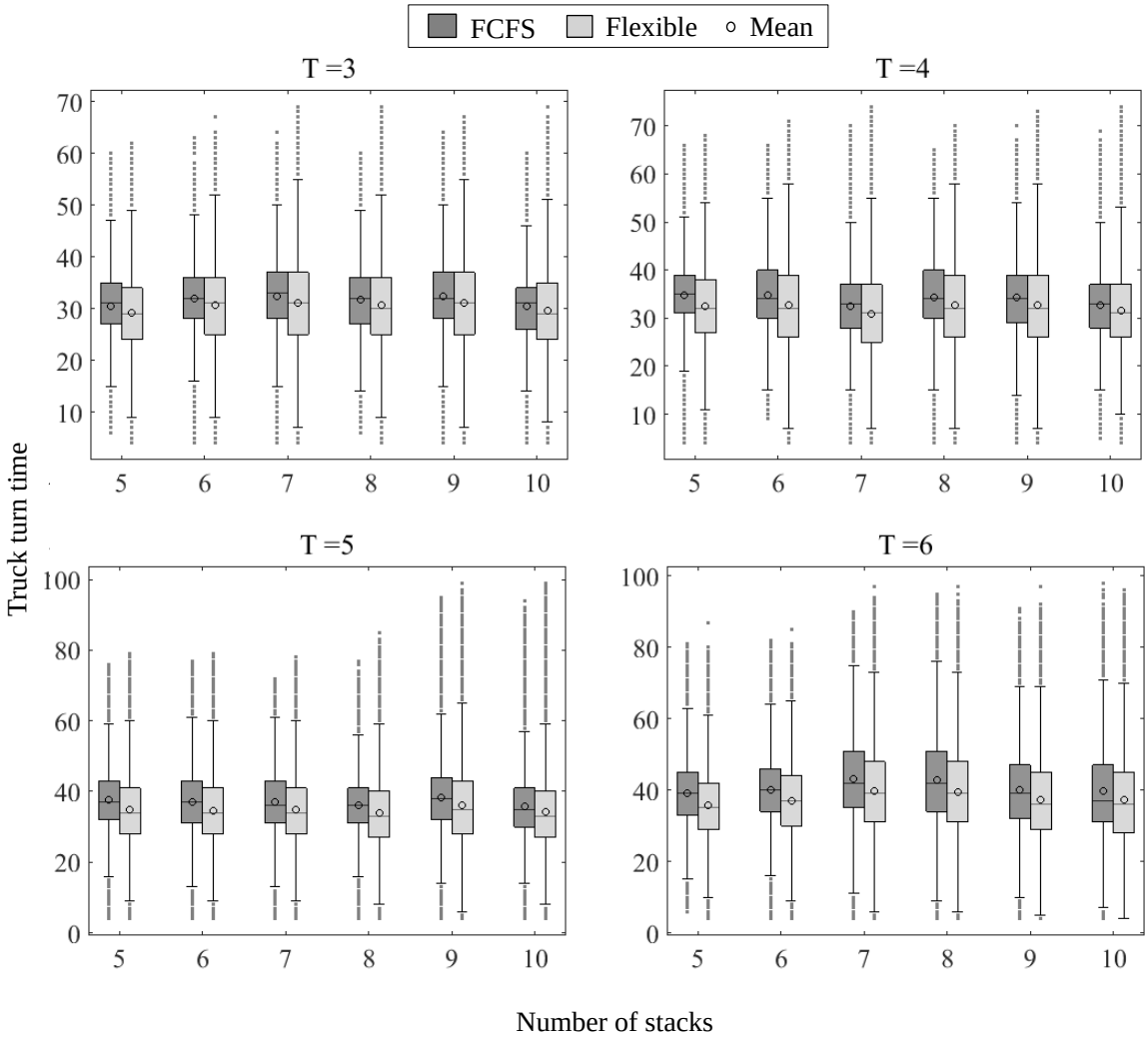


Fig. 2. 9 Grouped box plots of the truck turn time under two service policies for the instances with 67% fill rate, '50%' customer preference scenario and on average 6 containers per batch

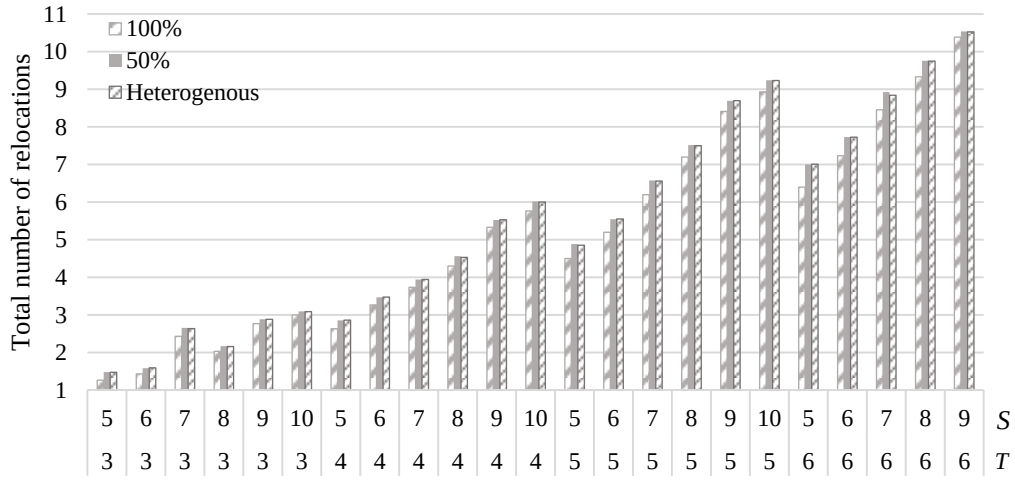
2.7.4 Influence of customer preference

In this section, we analyse the influence of customer preferences on the results of the Sooo extension model. We consider three sets of customer preference scenario: i) all trucks arrive at the first sub-time window with the probability of 100% ('100%'); ii) all trucks arrive at the first sub-time window with the probability of 50% ('50%'); iii) trucks arrive at the first sub-time window with different probabilities (heterogeneous). Appendix F reports the results obtained by the extended APBFS algorithm of these three sets of customer preference scenarios on the instances with small batches and a 50% fill rate. For the heterogeneous scenario, we generate 10 samples of customer preferences randomly for each of the 30 instances of each problem class, and hence, each problem class has 300 instances to be solved. The number of instances that are solved optimally is given in the form 'x/300' and '√' indicates that all instances out of 300 are solved optimally. Note that it takes about six days to obtain the results of the problem class with T=6 and S=9 for the heterogeneous scenario since 127 out of 300 instances cannot be solved optimally within one hour, we did not conduct the experiments of the problem class with T=6 and S=10 (because it would take much longer computational time than six days).

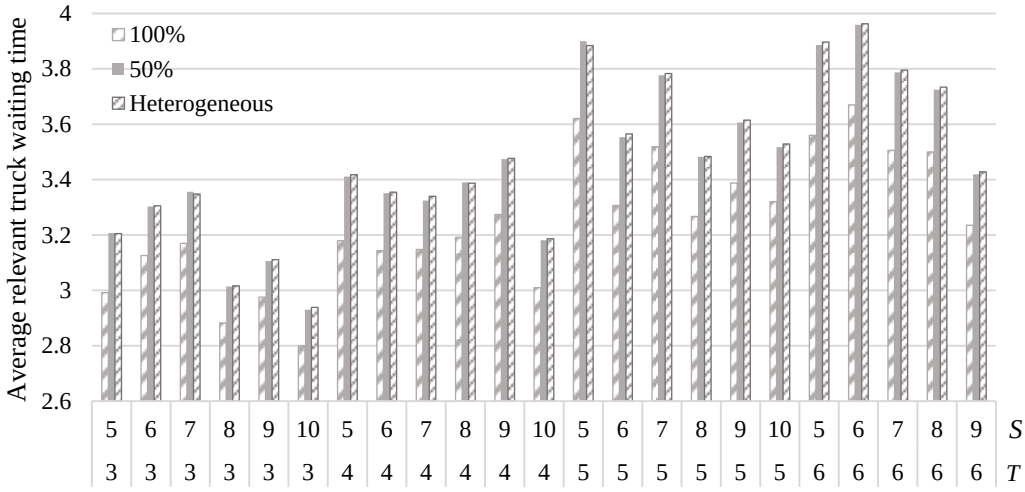
Concerning the computational efficiency, from Appendix F, it is observed that the instances with the

‘100%’ scenario take less time to obtain optimal solutions than the ‘50%’ scenario, and there are fewer hard instances for the ‘100%’ scenario. This can be explained by the fact that in the ‘100%’ scenario, each chance node only has one offspring, which reduces the burden of the decision tree. By contrast, the heterogeneous scenario takes more time to be solved. The reason is that the abstract technique does not work efficiently for the heterogeneous scenario as it rarely happens that two nodes are equivalent since the preferences of customers differ from each other. Even so, there is no obvious change in the number of hard instances, which can be seen from the ‘Solved’ columns of the ‘50’ scenario and the heterogeneous scenario.

In terms of the objective values, from Fig. 2. 10, we can see that the heterogenous scenario and the ‘50%’ scenario perform similarly, which have obvious differences from the ‘100’ scenario. This implies that if all customers tend to arrive at a specific sub-time window of their appointed time windows, the results will be influenced significantly. Besides, if customers have heterogeneous preferences, we can use the results of the ‘50%’ scenario as an approximation of the objective values of the heterogeneous scenario. However, this does not mean that the solution of the ‘50%’ scenario is feasible to the solution of the heterogeneous scenario.



(a) Comparison of the total number of relocations



(b) Comparison of the average relevant truck waiting time

Fig. 2. 10 Comparison between three sets of customer preference scenarios for instances with small batches and a 50% fill rate

We summarize the key findings of the experiments below. Firstly, the solution capacity of the two proposed exact solution algorithms is quite similar. The extended APBFS algorithm can solve 87.5% of the instances with T=3,4 within 30 seconds. Secondly, our proposed lower bound *LB-FS* is more effective for instances with lower tiers (T=3,4). In 73% of the instances with T=3,4, our lower bound is fairly close to the optimal solution with a gap within 5%. Thirdly, for the instances that can be solved optimally, the Sooo extension model can reduce the average relevant truck waiting time by 2.5% - 11% in comparison to the Sooo model, which indicates the significance of considering truck waiting time in addressing the SCRP-FS. Fourthly, the SEML heuristic outperforms the SEM heuristic in both performances, which demonstrates the importance of looking ahead on the decision-making of retrieval sequence. For the instances that we have access to the optimal solutions, in about 84% cases the total number of relocations obtained by the SEML heuristic is very close to the optimal solutions with a gap within 2%, and the maximum gap is 4.28%. Fifthly, the proposed flexible service policy can significantly reduce both the number of relocations and the relevant truck waiting times. Although some trucks may experience a little longer turn time, on average the service each truck receives can be improved. For the benchmark instance set, the largest relative reduction on the number of relocations is around 38% and 30%, which leads to a 9.6% and 11.3% reduction in the average relevant waiting time, respectively, for the bay of a 50% and a 67% fill rate. The benefit is more obvious for the instances with smaller and sparse bays, larger batches, and concentrated truck arrivals within one of the sub-time windows. Lastly, customers preferring a specific sub-time window of their appointed time windows has a great influence on the results.

2.8 Conclusions

In this paper, we have considered the stochastic container relocation problem with flexible service policies (termed as SCRP-FS), which focuses on retrieving and relocating import containers with uncertain truck arrival orders. The trucks arrive at the terminal randomly within their appointed time windows. The containers whose designated trucks arrive at the same sub-time window are allowed to be retrieved out-of-order. Customers (trucks)' preference is taken into consideration to describe the randomness of truck arrivals within the same time window. The problem is first formulated by a stochastic dynamic programming model to minimise the expected number of relocations, which is termed as the Sooo model. Then a Sooo extension model is developed considering a primary objective the same as the Sooo model and a secondary objective of minimising the total truck waiting times of each batch sequentially. The Sooo extension model not only considers the terminal operator's objective but also the trucks' objective. Built upon a state-of-the-art algorithm for solving the SCRP, tree search-based algorithms are developed to make optimal recommendations about the retrieval sequence of the next batch of containers and the relocation positions of the blocking containers. Moreover, two heuristic algorithms, SEM and SEML, are designed to seek high-quality solutions efficiently for practical-size problems. A discrete event-driven simulation model is developed to evaluate the performance of the algorithms (optimal and heuristic). Extensive computational experiments demonstrate the effectiveness of the models and the algorithms.

On the theoretical side, firstly, the SCRP-FS generalizes the conventional SCRP from two perspectives. On the one hand, the flexible service policy relaxes the traditional FCFS policy, which provides more opportunities for reducing the number of relocations and allows for reducing the trucks' waiting time as well. On the other hand, the assumption of uniformly distributed truck arrivals within the same time window is relaxed by a more general probabilistic model. The capability of capturing the customers' preference-based arrival behaviour, in particular, is a major advantage of the probabilistic model. Secondly,

the proposed methodology contributes to the literature of solving multiple objective multi-stage stochastic optimisation problems by embedding the optimisation of the secondary objectives within the multi-stage optimisation procedure for the primary objective. Such methodology may be applicable to other transportation optimisation problems such as berth allocation problems or train loading problems, in which decisions are made dynamically and multiple objectives are prioritized.

On the practical side, based on our findings, we provide some managerial insights to terminal operators and truck companies. Firstly, by slightly diverting the current FCFS service policy to the flexible service policy that implements out-of-order retrievals within half of the appointment time window, both the number of relocations and the average truck waiting time during the retrieval service can be significantly reduced; and the service equity between two sub-groups of trucks is maintained. Secondly, the flexible service policy is more beneficial in the following practical situations: the container terminal uses small bay or/and sparse stacking strategy; the containers to be retrieved in a bay are booked in large batches; the trucks arrive within either the earlier segment or the latter segment of their appointed time windows concentratedly. Thirdly, customer preference has a great influence on both the number of relocations and the truck waiting times during the retrieval service. Lastly, the developed SEML heuristic can generate good solutions very fast, which can be applied in practice to enable the real-time dynamic decision-making for the SCRP-FS.

This paper provides several directions for further research. Firstly, the proposed models and algorithms are reasonably general and flexible, which allows for further refining and improvement, e.g. terminal operators could choose different sizes of the sub-batches or multiple sub-batches. The optimisation framework will be similar and the structure of the decision tree does not need change. However, if the terminal operator decides to have more sub-batches, the size of the search tree will be larger due to the consideration of more possibilities of sub-batches. Hence, a more efficient search algorithm needs to be developed to obtain exact solutions. Nevertheless, our proposed heuristics are supposed to be still efficient because their time complexities are decreasing with the decrease of the batch size. Besides, the benefits of the flexible service policy need to be further evaluated because the terminal operator will have less control over the truck service sequence if they use more sub-batches. Secondly, this study could be extended to address more general SCRP problems where trucks do not necessarily arrive within their appointed time windows. In the real world, the arrival of a truck may be prior to or later than the appointed time window. An extended arrival time window that includes both the preceding and the succeeding time window relative to the appointed time window is more appropriate to predict trucks' arrival times. The probability of the deviation from the appointed time window could be gained from historical data and be considered as our proposed customer preference. Thirdly, based on our proposed lower bound, more efficient stacking policies considering the possibility of out-of-order retrievals in the future could be developed to stack import containers in an orderly configuration so that fewer relocations are needed during the retrieval process.

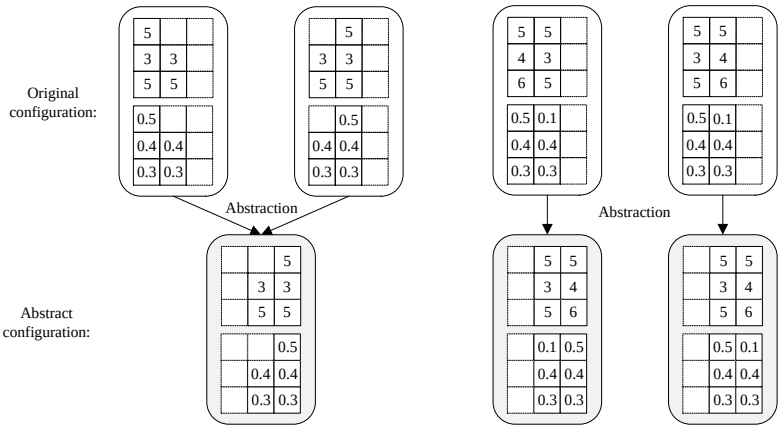
2.9 Appendix

Appendix A. Some illustrations of the exact algorithms

A.1. Illustration of the abstraction technique

Fig. 2. A.1(a) shows the application of the abstract technique on the node 10 and node 11 in Fig. 2. 4. By $Abstract(10)$ and $Abstract(11)$, node 10 and node 11 are projected to the same abstract configuration. This means if $f(10)$ is known, $f(11)$ can be directly returned to be $f(10)$ without further branching. Fig. 2.

A.1(b) illustrates two unequivalent abstract configurations due to the difference in their abstract preference configurations although they have the same abstract priority configuration.



(a) Equivalent configuration (b) Unequivalent configuration

Fig. 2. A.1. Illustration of the abstraction technique

A.2. A sample decision tree developed by the extended APBFS algorithm

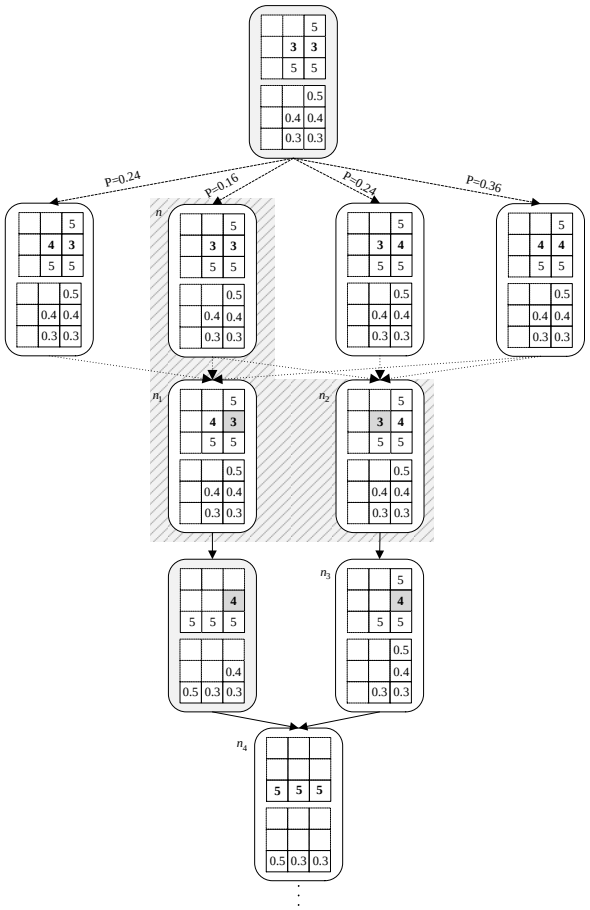


Fig. 2. A.2. A sample decision tree developed by the extended APBFS algorithm.

Fig. 2. A.2 presents a decision tree developed by the extended APBFS algorithm. The initial node in this decision tree is the abstract configuration of the last node in the decision tree of Fig. 2. 4. Let us focus on the nodes highlighted with upward diagonal background: n, n_1, and n_2, to illustrate the consideration of the secondary objective. n is a SD node with $\Phi_n^{APBFS} = \{n_1, n_2\}$, and $f(n)$=1. As $f(n_1) = f(n_2) = 1 = f(n)$, we calculate the waiting time indicator of n_1 and n_2 by Algorithm 2. We obtain that $w(n_2) = 1 < w(n_1) = 2$.

Therefore, the best offspring of node n is n_2 (step 4.7-4.8 in Algorithm 3). Note that in the APBFS algorithm, if n_1 is first added into Φ_n^{APBFS}, n_2 will not be able to be included into Φ_n^{APBFS} as $lb(n_2)=1$ is not less than $f(n_1)$, which means that we lose the opportunity to find the optimal solution with regard to the secondary objective.

Appendix B. Details of the heuristics

B.1. Calculating the BI and DI of the EM extension heuristic

Fig. 2. B.1 is used for illustration, which shows how the EM extension heuristic makes decisions on a simple example where the truck arrival sequence of the three containers in the first batch (u_7, u_{10}, u_5) has been revealed. The container in the shaded slot is the target container to be retrieved. The container in the upward diagonal slot is the blocking container to be relocated at the current step. The numbers under the priority matrix correspond to the *m*(*s*) of each candidate stack, and the value of *M* is also given.

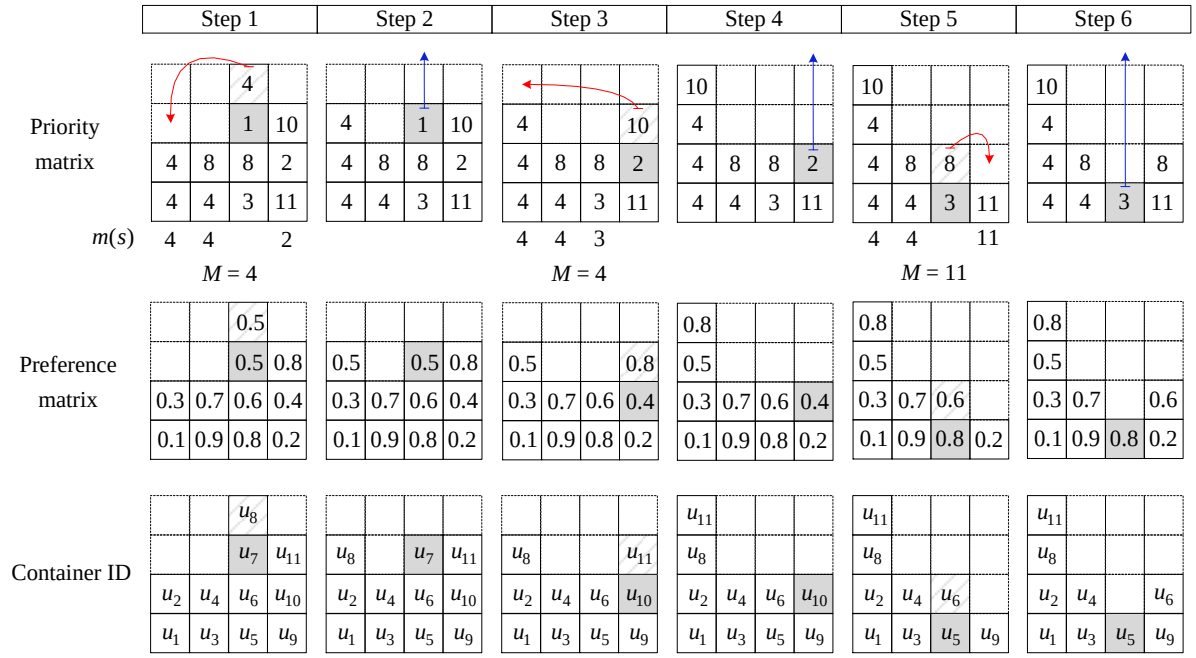


Fig. 2. B.1. Decisions by the EM extension heuristic on an example

(1) Method of computing BI

If $M=l_c$, EM extension selects the stack with the minimum $BI(s')$ to minimise the probability of *c* being blocking if *c* is relocated to stack s'. Given a configuration *B* with *M*, a stack *s* where *c* is located before being relocated, a stack s' that satisfies $m(s')=M$, $BI(s')$ is computed as follows. Step 1 in Fig. 2. B.1 is used to illustrate the computing method. Without special instruction, the 'stack 1' used in this sub-section refers to the stack 1 in the configuration under step 1 in Fig. 2. B.1.

Let $M_{s'}=\{c_1,...,c_N\}$, $N=|M_{s'}|$, be the set of containers labelled *M* and located in s'. We first compute the probability that *c* is not blocking if relocated to s', i.e., $1-BI(s')$. Let us consider the two cases in terms of the sub-batch of *c*: #1) *c* is in the former sub-batch; #2) *c* is in the latter sub-batch.

#1. *c* is in the former sub-batch.

Under case 1, we consider two mutually exclusive sub-cases (#1.1 and #1.2) in terms of the sub-batch of containers in $M_{s'}$.

#1.1. At least one container $c_i \in M_{s'}$ is in the former sub-batch.

There are totally $\sum_{k=1}^{N}\binom{N}{k}$ scenarios that satisfy #1.1. Let $Comb_k = \left\{comb_k^1, comb_k^2, ..., comb_k^{\binom{N}{k}}\right\}$ denote the set of all scenarios represented by the combinations of the N elements in $M_{s'}$ taken k, $k = \{1, ..., N\}$. The size of $Comb_k$ is $\binom{N}{k}$. Each element $comb_k^i \in Comb_k$, $i \in \left\{1, ..., \binom{N}{k}\right\}$, represents a scenario where the elements in $comb_k^i$ are in the former sub-batch. For example (see Fig. 2. B.1), in the configuration of step 1, $M_1 = \{u_1, u_2\}$, N=2, $Comb_1 = \left\{comb_1^1, comb_1^2\right\}$, wherein $comb_1^1 = \{u_1\}$, $comb_1^2 = \{u_2\}$, and $Comb_2 = \{comb_2^1\}$, wherein $comb_2^1 = \{u_1, u_2\}$. The probability of $Comb_k$ is equal to $\sum_{i=1}^{\binom{N}{k}} \prod_{c_j \in comb_k^i} p_{c_j} \prod_{c_j \notin comb_k^i} (1 - p_{c_j})$. Then the probability that c is not blocking in the scenario set $Comb_k$ is equal to $\sum_{i=1}^{\binom{N}{k}} \prod_{c_j \in comb_k^i} p_{c_j} \prod_{c_j \notin comb_k^i} (1 - p_{c_j})/(k+1) \cdot p_{c_i}$. Considering all combinations from k=1 to k=N, we have the probability that c is not blocking in case 1.1, which is equal to $\sum_{k=1}^{N}\sum_{i=1}^{\binom{N}{k}} \prod_{c_j \in comb_k^i} p_{c_j} \prod_{c_j \notin comb_k^i} (1 - p_{c_j})/(k+1) \cdot p_{c_i}$. Taking stack 1 for example, we have

$$\sum_{k=1}^{N}\sum_{i=1}^{\binom{N}{k}} \prod_{c_j \in comb_k^i} p_{c_j} \prod_{c_j \notin comb_k^i} (1 - p_{c_j})/(k+1) \cdot p_{c_i} = \sum_{k=1}^{2}\sum_{i=1}^{\binom{2}{k}} \prod_{c_j \in comb_k^i} p_{c_j} \prod_{c_j \notin comb_k^i} (1 - p_{c_j})/(k+1) \cdot p_{c_i}$$

$$= (0.3 \times 0.9/2 + 0.1 \times 0.7/2 + 0.3 \times 0.1/3) \times 0.5 = 0.09.$$

#1.2. All containers in $M_{s'}$ are in the latter sub-batch.

In this case, c is surely not blocking. Then the probability that c is not blocking in case 1.2 is equal to $\prod_{i=1}^{N}(1 - p_{c_i}) \cdot p_c$. Taking stack 1 for example, we have $\prod_{i=1}^{N}(1 - p_{c_i}) \cdot p_c = \prod_{i=1}^{2}(1 - p_{c_i}) \cdot p_c = 0.7 \times 0.9 \times 0.5 = 0.315$.

#2. c is in the latter sub-batch.

In this case, there exists only one scenario in which it is possible that c is not blocking, that is, all containers in $M_{s'}$ are in the latter sub-slot. Then the probability that c is not blocking in case 2 is equal to $\prod_{i=1}^{N}(1 - p_{c_i}) \cdot (1 - p_c)/(N+1)$. Taking stack 1 for example, we have $\prod_{i=1}^{N}(1 - p_{c_i}) \cdot (1 - p_c)/(N+1) = \prod_{i=1}^{2}(1 - p_{c_i}) \cdot (1 - p_c)/3 = 0.7 \times 0.9 \times 0.5/3 = 0.105$.

The above cases exhaust all the possible scenarios of the sub-batches of the containers labelled M. Therefore, by summing the above expressions, we have the probability that c is not blocking if relocated to s', i.e., $1 - BI(s')$, as expressed in Eq. (2.A.1):

$$1-BI(s')=\sum_{k=1}^{N}\sum_{i=1}^{\binom{N}{k}}\prod_{c_j\in comb_k^i}p_{c_j}\prod_{c_j\notin comb_k^i}(1-p_{c_j})/(k+1)\cdot p_{c_i}+\prod_{i=1}^{N}(1-p_{c_i})\cdot p_c+\prod_{i=1}^{N}(1-p_{c_i})\cdot(1-p_c)/(N+1)$$

$$=\sum_{k=1}^{N}\sum_{i=1}^{\binom{N}{k}}\prod_{c_j\in comb_k^i}p_{c_j}\prod_{c_j\notin comb_k^i}(1-p_{c_j})/(k+1)\cdot p_{c_i}+\prod_{i=1}^{N}(1-p_{c_i})\cdot\left(p_c+(1-p_c)/(N+1)\right) \quad (2.A.1)$$

Finally, we have the probability of *c* being blocking if relocated to s', i.e., $BI(s')$, as calculated by Eq. (2.A.2).

$$BI(s')=1-\sum_{k=1}^{N}\sum_{i=1}^{\binom{N}{k}}\prod_{c_j\in comb_k^i}p_{c_j}\prod_{c_j\notin comb_k^i}(1-p_{c_j})/(k+1)\cdot p_{c_i}-\prod_{i=1}^{N}(1-p_{c_i})\cdot\left(p_c+(1-p_c)/(N+1)\right) \quad (2.A.2)$$

For example (see Fig. 2. B.1), in the configuration of step 1, the BI of stack 1 is calculated as $BI(1)=1-(0.09+0.315+0.105)=0.49$. Also, we obtain *BI*(2)=0.7 by Eq. (2.A.2). As *BI*(1)<*BI*(2), stack 1 is selected as the relocating stack at step 1 by the EM extension heuristic.

(2) Method of computing DI

If $M<l_c$, which means that it is unavoidable that *c* will be relocated again in the future, EM extension selects the stack with the minimum $DI(s')$ to delay the next relocation of *c*. Given a configuration *B* with *M*, a stack *s* where *c* is located before being relocated, a stack s' that satisfies $m(s')=M$, $DI(s')$ is computed as follows. Step 3 in Fig. 2. B.1. is used to illustrate the computing method. Without special instruction, the 'configuration' used in this sub-section refers to the configuration at step 3 in Fig. 2. B.1.

Let $M_B=\{c_1,...,c_{L+1}\}$, $|M_B|=L+1$, be the set of all containers labelled *M* in configuration *B*, and $M_{s'}$ be the set of containers labelled *M* and located in s'. For example, in the illustrated configuration, $M_B=\{u_1,u_2,u_8,u_3\}$, *L*=3, $M_1=\{u_1,u_2,u_8\}$, and $M_2=\{u_3\}$. Given a candidate stack s', we first compute the probability of each container $c_i\in M_{s'}$ being the first one to be retrieved among the containers in M_B, denoted by $DI'(c_i)$. Since the retrieval of any container $c_i\in M_{s'}$ will cause the next relocation of *c* if *c* is relocated to s', by definition, we have $DI(s')=\sum_{c_i\in M_{s'}}DI'(c_i)$.

Now let us consider a container $c_i\in M_{s'}$ and compute $DI'(c_i)$. Suppose all the containers in $M_B\setminus c_i$ are located in a dummy stack and c_i is the container to be relocated to this stack. Then $DI'(c_i)$ is equal to the probability that c_i is not blocking if relocated to this dummy stack. Therefore, using the Eq. (2.A.1) of calculating $1-BI(s')$, $DI'(c_i)$ is computed by Eq. (2.A.3).

$$DI'(c_i)=\sum_{k=1}^{L}\sum_{n=1}^{\binom{L}{k}}\prod_{c_j\in cmb_{c_i,k}^n}p_{c_j}\cdot\prod_{c_j\notin cmb_{c_i,k}^n}(1-p_{c_j})/(k+1)\cdot p_{c_i}+\prod_{c_j\in M_B\setminus c_i}(1-p_{c_j})\cdot p_{c_i}+\prod_{c_j\in M_B\setminus c_i}(1-p_{c_j})\cdot(1-p_{c_i})/(L+1)$$

$$=\sum_{k=1}^{L}\sum_{n=1}^{\binom{L}{k}}\prod_{c_j\in cmb_{c_i,k}^n}p_{c_j}\cdot\prod_{c_j\notin cmb_{c_i,k}^n}(1-p_{c_j})/(k+1)\cdot p_{c_i}+\prod_{c_j\in M_B\setminus c_i}(1-p_{c_j})\cdot\left(p_{c_i}+(1-p_{c_i})/(L+1)\right) \quad (2.A.3)$$

wherein, $cmb_{c_i,k}^n\in Cmb_{c_i,k}$. $Cmb_{c_i,k}=\left\{cmb_{c_i,k}^1,cmb_{c_i,k}^2,...,cmb_{c_i,k}^{\binom{L}{k}}\right\}$ denotes the set of all scenarios represented

by the combinations of the *L* elements in $M_B \setminus c_i$ ($c_i \in M_{s'}$) taken *k*, $k=\{1,...,L\}$. The size of $Cmb_{c_i,k}$ is $\binom{L}{k}$.

Therefore, $DI(s')=\sum_{c_i \in M_{s'}} DI'(c_i)$ is computed by Eq. (2.A.4).

$$DI(s')=\sum_{c_i \in M_{s'}}\left[\sum_{k=1}^{L}\sum_{n=1}^{\binom{L}{k}}\prod_{c_j \in cmb^n_{c_i,k}} p_{c_j}\cdot\prod_{c_j \notin cmb^n_{c_i,k}}(1-p_{c_j})/(k+1)\cdot p_{c_i}+\prod_{c_j \in M_B \setminus c_i}(1-p_{c_j})\cdot\left(p_{c_i}+(1-p_{c_i})/(L+1)\right)\right] \quad (2.A.4)$$

Let us calculate $DI'(u_8)$ in the illustrated configuration. $Cmb_{u_8,1}=\{cmb^1_{u_8,1},cmb^2_{u_8,1},cmb^3_{u_8,1}\}$, wherein $cmb^1_{u_8,1}=\{u_1\}$, $cmb^2_{u_8,1}=\{u_2\}$, and $cmb^3_{u_8,1}=\{u_3\}$; $Cmb_{u_8,2}=\{cmb^1_{u_8,2},cmb^2_{u_8,2},cmb^3_{u_8,2}\}$, wherein $cmb^1_{u_8,2}=\{u_1,u_2\}$, $cmb^2_{u_8,2}=\{u_1,u_3\}$, and $cmb^3_{u_8,2}=\{u_2,u_3\}$; $Cmb_{u_8,3}=\{cmb^1_{u_8,3}\}$, wherein $cmb^1_{u_8,3}=\{u_1,u_2,u_3\}$. For $k=\{1,...,3\}$, we calculate the first term of Eq. (2.A.3):

For *k*=1,

$$\sum_{n=1}^{\binom{3}{1}}\prod_{c_j \in cmb^n_{u_8,1}} p_{c_j}\cdot\prod_{c_j \notin cmb^n_{u_8,1}}(1-p_{c_j})/(1+1)\cdot p_{u_8}=(0.1\times0.7\times0.1+0.3\times0.9\times0.1+0.9\times0.7\times0.9)/2\times0.5=0.15025$$

For *k*=2,

$$\sum_{n=1}^{\binom{3}{2}}\prod_{c_j \in cmb^n_{u_8,2}} p_{c_j}\cdot\prod_{c_j \notin cmb^n_{u_8,2}}(1-p_{c_j})/(2+1)\cdot p_{u_8}=(0.1\times0.3\times0.1+0.1\times0.9\times0.7+0.3\times0.9\times0.9)/3\times0.5=0.0515$$

For *k*=3, $\sum_{n=1}^{\binom{3}{3}}\prod_{c_j \in cmb^n_{u_8,3}} p_{c_j}\cdot\prod_{c_j \notin cmb^n_{u_8,3}}(1-p_{c_j})/(3+1)\cdot p_{u_8}=0.1\times0.3\times0.9/4\times0.5=0.003375$

Summing the above expressions, we have,

$$\sum_{k=1}^{3}\sum_{n=1}^{\binom{3}{k}}\prod_{c_j \in cmb^n_{u_8,k}} p_{c_j}\cdot\prod_{c_j \notin cmb^n_{u_8,k}}(1-p_{c_j})/(k+1)\cdot p_{u_8}=0.15025+0.0515+0.003375=0.205125$$

Then, we calculate the second term of Eq. (2.A.3):

$$\prod_{c_j \in M_B \setminus u_8}(1-p_{c_j})\cdot\left(p_{u_8}+(1-p_{u_8})/(L+1)\right)=(0.9\times0.7\times0.1)\times\left(0.5+0.5/4\right)=0.039375$$

By Eq. (2.A.4), we have $DI'(u_8)=0.205125+0.039375=0.2445$. In the same way, we obtain $DI'(u_2)=0.1385$, $DI'(u_1)=0.0485$, and $DI'(u_3)=0.5685$. Therefore, $DI(1)=DI'(u_8)+DI'(u_2)+DI'(u_1)$ =0.2445+0.1385+0.0485=0.4315, and $DI(2)=DI'(u_3)=0.5685$. As $DI(1)<DI(2)$, stack 1 is selected as the target relocating stack at step 3 by the EM extension heuristic.

B.2. An illustrative example for the SEM heuristic

We use Fig. 2. B.2. to illustrate the sequencing decision of the SEM heuristic. In the initial configuration X_0, there are five batches of containers. The truck arrival information of the first batch is revealed to be that u_7 is in the first sub-batch and u_5 and u_{10} are in the second sub-batch, as shown in bold in Step 1. Now we present the decisions to retrieve the first batch containers. The container in the shaded slot represents the target container to be retrieved. The container in the upward diagonal slot represents the blocking container to be relocated. At Step 1, $lmin=1$, $\Theta=\{u_7\}$, and thus there is no doubt that u_7 is selected as the target container. After retrieving u_7, $lmin=2$, $\Theta=\{u_5,u_{10}\}$, and $r(u_5)=r(u_{10})=1$. As

$r(u_5)=r(u_{10})$, the SEM selects u_{10} as the target container arbitrarily. After Step 4, $lmin=3$, $\Theta=\{u_5\}$, and thus u_5 is selected as the target container out of question.

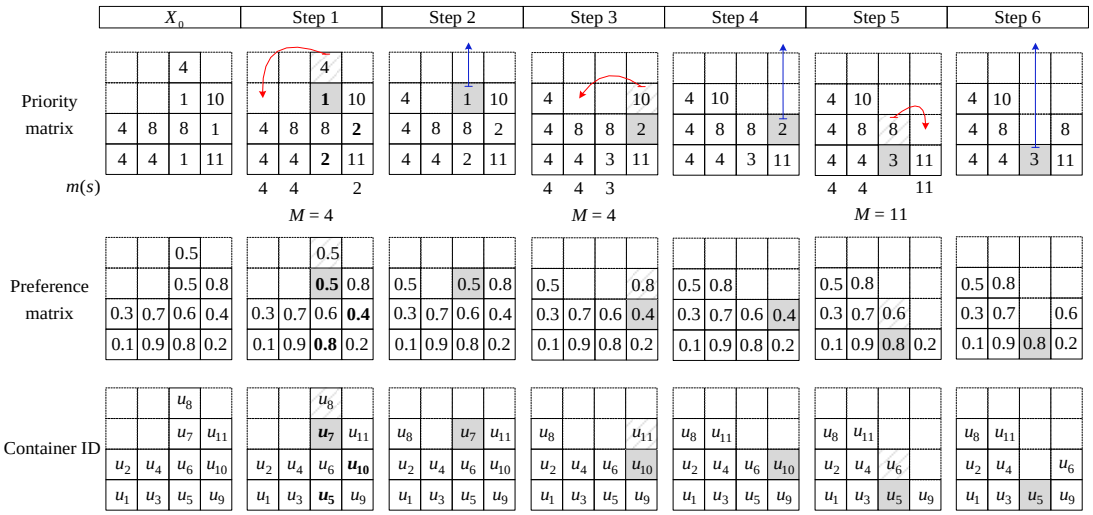


Fig. 2. B.2. Decisions by the SEM heuristic on an example

B.3. Calculating the BIS and DIS of the SEM heuristic

Fig. 2. B.2 is used for illustration.

(1) Method of computing BIS

Given a configuration *B* with *M*, a stack *s* where *c* is located before being relocated, a stack s' that satisfies $m(s')=M$, the method of computing $BIS(s')$ is introduced here. Let $M_{s'}=\{c_1,...,c_N\}$, $N=|M_{s'}|$, be the set of containers labelled *M* and located in s'. Container *c* being blocking if relocated to s' occurs only in the scenario where *c* is in the latter sub-batch and there is at least one container $c_i \in M_{s'}$ in the former sub-batch. Therefore, we have,

$$BIS(s')=(1-p_c)\left(1-\prod_{c_i\in M(s')}(1-p_{c_i})\right) \tag{2.A.5}$$

The term $\prod_{c_i\in M(s')}(1-p_{c_i})$ of Eq. (2.A.5) is the probability that all the containers in $M_{s'}$ are in the latter sub-batch, and thus $\left(1-\prod_{c_i\in M(s')}(1-p_{c_i})\right)$ is the probability that at least one of them is in the former sub-batch.

Take the configuration at step 1 (see Fig. 2. B.2) for example. $c=u_8$, *M*=4, and thus stack 1 and stack 2 are candidate stacks. By Eq. (2.A.5), $BIS(1)=(1-0.5)(1-0.7\times0.9)=0.185$, $BIS(2)=(1-0.5)(1-0.1)=0.45$. As *BIS*(1)<*BIS*(2), stack 1 is selected for relocating u_8.

(2) Method of computing DIS

In the sequencing rule introduced above, the one with the lowest number of blocking containers among the containers with the smallest label is selected as the target container, ties being broken arbitrarily. Therefore, a container c_i is surely being the first one to be retrieved in its batch only in the situation that satisfies the following two conditions: 1) c_i is in the former sub-batch; 2) c_i is with the lowest number of

blocking containers (i.e., r_{c_i}) among the containers in the former sub-batch. The second condition means that all the containers above c_i labelled *M* must be in the latter sub-batch and for each stack s' that satisfies $m(s') = M$, if there are containers labelled *M* among the top $r_{c_i}+1$ number of containers (if any) in stack s', these containers must be in the latter sub-batch.

Given a configuration *B* with *M*, a stack *s* where *c* is located before being relocated, a stack $s' \in S_M$, wherein S_M is the set of stack that satisfies $m(s') = M$, the method of computing $DIS(s')$ is introduced here. Let $M_{s'}$ be the set of containers labelled *M* located in s'. Given a candidate stack s', we first compute the probability of each container $c_i \in M_{s'}$ surely being the first one to be retrieved in its batch, denoted by $DIS'(c_i)$. By definition, we have $DIS(s') = \sum_{c_i \in M_{s'}} DIS'(c_i)$. Let $T(n,s)$ denote the set of top *n* number of containers labelled *M* in stack *s*. Then, we have $DIS'(c_i) = p_{c_i} \cdot \sum_{c_j \in T(r(c_i),s')} (1-p_{c_j}) \sum_{s'' \in S_M \setminus s'} \sum_{c_j \in T(r(c_i)+1,s'')} (1-p_{c_j})$. By summing the $DIS'(c_i)$ of all containers $c_i \in M_{s'}$, we get,

$$DIS(s') = \sum_{c_i \in M_{s'}} p_{c_i} \cdot \sum_{c_j \in T(r(c_i),s')} (1-p_{c_j}) \sum_{s'' \in S_M \setminus s'} \sum_{c_j \in T(r(c_i)+1,s'')} (1-p_{c_j}) \quad (2.A.6)$$

Taking the configuration at step 3 (see Fig. 2. B.2) for example, where $c = u_{11}$, *M*=4, $S_M = \{1,2\}$, let us compute $DIS'(u_2)$. $r(u_2) = 1$, $T(1,1) = \{u_8\}$, $T(2,2) = \{u_3\}$, and thus $\sum_{c_j \in T(r(c_i),s')} (1-p_{c_j}) = \sum_{c_j \in T(1,1)} (1-p_{c_j}) = (1-p_{u_8}) = 0.5$, $\sum_{s'' \in S_M \setminus s'} \sum_{c_j \in T(r(c_i)+1,s'')} (1-p_{c_j}) = \sum_{s'' \in \{2\}} \sum_{c_j \in T(2,s'')} (1-p_{c_j}) = (1-p_{u_3}) = 0.1$. Consequently, $DIS'(u_2) = 0.3 \times 0.5 \times 0.1 = 0.015$. Similarly, we get $DIS'(u_8) = 0.5$, $DIS'(u_1) = 0.0035$, and $DIS'(u_3) = 0.315$. By Eq. (2.A.6), we have $DIS(1) = DIS'(u_2) + DIS'(u_8) + DIS'(u_1) = 0.5 + 0.015 + 0.0035 = 0.5185$ and $DIS(2) = DIS'(u_3) = 0.315$. As $DIS(2) < DIS(1)$, stack 2 is selected for relocating u_{11}.

B.4. An illustrative example for the SEML heuristic

We use Fig. 2. B.3, which continues Step 2 of Fig. 2. B.2, to illustrate the sequencing rule of the SEML heuristic. For brevity, we only present the priority matrix. After step 2, $lmin = 2$, and $\Theta = \{u_5, u_{10}\}$. Because $r(u_5) = r(u_{10}) = 1$, both u_5 and u_{10} are potential target containers (*H*=2). Hence, we evaluate the contribution of the two feasible retrieval sequences respectively: $u_5 \to u_{10}$ and $u_{10} \to u_5$, through step 3 to step 6. The final configurations after implementing the two sequences are given at step 7. It is obvious that the two final configurations have the same lower bound. And because both of the sequences cause two realised relocations, their contributions are the same. Therefore, we can choose one arbitrarily from $u_5 \to u_{10}$ and $u_{10} \to u_5$ as the determined retrieval sequence for u_5 and u_{10}.

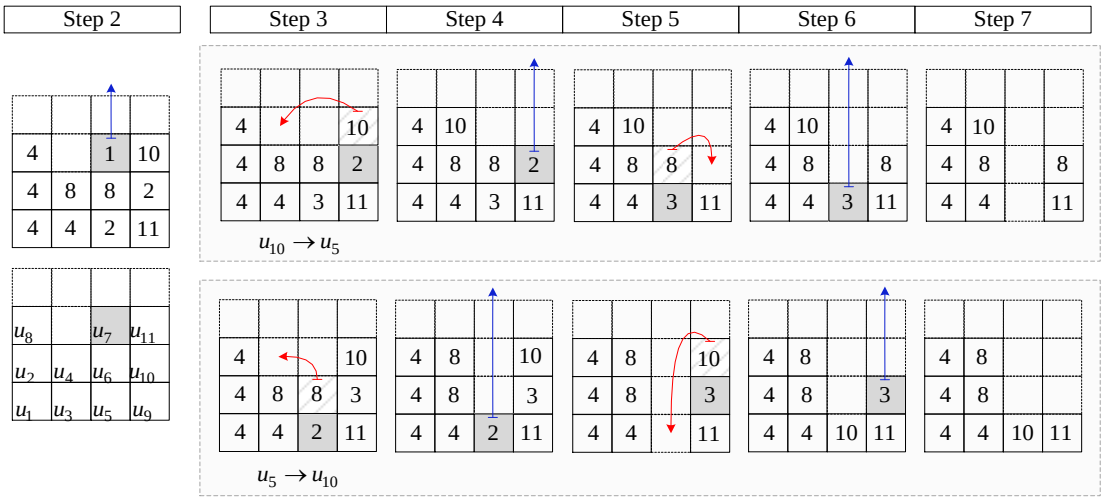


Fig. 2. B.3. Decisions by the SEML heuristic on an example

Appendix C. Performance of the proposed models and exact algorithms

Note that in the tables, the "Opt" column represents the optimal expected number of relocations obtained by the corresponding exact algorithms; the "Rel" column represents the estimated number of relocations obtained by simulation.

C.1. Results of the extended APBFS algorithm, the SEML heuristic, and the lower bound

Table 2. C.1 Performance of the extended APBFS algorithm, the SEML heuristics and lower bound for instances with small batches and 67% fill rate

				Extended APBFS						SEML		Gap	
T	*S*	*C*	lb	lb*	Opt	Solved	CPU(s)	Rel	AveWait (min)	Rel	AveWait (min)	Gap [Rel]	Gap [AveWait]
3	5	10	2.621	2.621	2.789	√	0.05	2.786	3.224	2.786	3.222	0.00%	-0.06%
	6	12	3.433	3.433	3.621	√	0.05	3.620	3.524	3.659	3.526	1.08%	0.06%
	7	14	3.708	3.708	3.756	√	0.06	3.759	3.506	3.786	3.506	0.72%	0.00%
	8	16	4.254	4.254	4.379	√	0.08	4.382	3.385	4.405	3.389	0.52%	0.12%
	9	18	4.592	4.592	4.815	√	0.11	4.816	3.408	4.846	3.408	0.62%	0.00%
	10	20	5.021	5.021	5.067	√	0.11	5.066	3.176	5.073	3.179	0.14%	0.09%
4	5	13	4.433	4.433	5.073	√	0.14	5.071	3.976	5.241	3.972	3.35%	-0.10%
	6	16	6.017	6.017	6.925	√	1.19	6.931	3.898	7.200	3.936	3.88%	0.97%
	7	19	6.042	6.042	6.927	√	1.16	6.929	3.465	7.026	3.475	1.40%	0.29%
	8	21	7.375	7.375	7.967	√	20.25	7.969	3.601	8.016	3.605	0.59%	0.11%
	9	24	8.775	8.775	9.259	√	11.80	9.257	3.619	9.353	3.619	1.04%	0.00%
	10	27	8.992	8.935	9.618	27	92.17	9.622	3.426	9.709	3.424	0.90%	-0.05%
5	5	17	7.358	7.058	8.802	26	24.26	8.802	4.145	9.187	4.172	4.37%	0.66%
	6	20	7.992	7.522	8.465	23	66.56	8.468	3.774	8.679	3.796	2.49%	0.59%
	7	23	9.475	9.091	10.339	22	340.50	10.340	3.934	10.569	3.945	2.21%	0.28%
	8	27	11.354	10.816	11.570	17	803.78	11.568	3.705	11.671	3.714	0.89%	0.24%
	9	30	12.879	12.388	13.436	10	728.19	13.436	3.669	13.555	3.673	0.89%	0.10%
	10	34	14.229	12.5	12.983	11	195.66	12.990	3.549	13.032	3.548	0.32%	-0.02%
6	5	20	9.496	8.958	11.183	15	613.80	11.181	4.085	12.006	4.148	7.38%	1.53%
	6	24	11.304	10.894	12.341	13	491.90	12.343	4.102	12.527	4.127	1.49%	0.60%
	7	28	13.258	13.75	15.375	2	253.16	15.357	3.864	15.359	3.918	0.01%	1.39%
	8	32	15.367	13.2	13.869	5	693.38	13.865	3.695	13.865	3.685	0.00%	-0.26%
	9	36	16.454	16.5	16.500	2	949.85	16.504	3.646	16.504	3.652	0.00%	0.18%

	10	40	19.375	15.25	15.250	1	232.22	15.259	3.857	15.259	3.880	0.00%	0.61%

Note: customer preference scenario: 50%

Table 2. C.2 Performance of the extended APBFS algorithm, the SEML heuristics and lower bound for instances with large batches and 50% fill rate

				Extended APBFS						SEML		Gap	
T	*S*	*C*	lb	lb*	Opt	Solved	CPU(s)	Rel	AveWait (min)	Rel	AveWait (min)	Gap [Rel]	Gap [AveWait]
3	5	8	1.263	1.263	1.278	√	0.45	1.281	7.206	1.283	7.215	0.16%	0.12%
	6	9	1.317	1.317	1.334	√	0.32	1.336	7.388	1.336	7.389	0.00%	0.01%
	7	11	2.317	2.317	2.340	√	0.47	2.339	7.075	2.339	7.078	0.00%	0.04%
	8	12	1.971	1.971	1.973	√	0.55	1.969	6.499	1.969	6.499	0.00%	0.00%
	9	14	2.617	2.617	2.621	√	0.72	2.614	6.764	2.615	6.764	0.04%	0.00%
	10	15	2.908	2.908	2.914	√	0.86	2.915	6.708	2.915	6.710	0.00%	0.03%
4	5	10	2.546	2.546	2.703	√	0.56	2.702	6.967	2.779	6.991	2.85%	0.34%
	6	12	3.183	3.183	3.315	√	1.13	3.312	7.122	3.357	7.140	1.36%	0.25%
	7	14	3.550	3.550	3.645	√	4.50	3.648	7.473	3.662	7.479	0.38%	0.08%
	8	16	4.058	4.058	4.094	√	6.46	4.090	7.502	4.102	7.507	0.29%	0.07%
	9	18	5.233	5.233	5.274	√	19.65	5.273	7.408	5.297	7.409	0.46%	0.01%
	10	20	5.746	5.746	5.799	√	16.80	5.803	6.863	5.844	6.869	0.71%	0.09%
5	5	13	3.952	3.952	4.470	√	32.02	4.466	8.154	4.633	8.187	3.74%	0.40%
	6	15	4.844	4.821	5.181	29	84.30	5.181	7.586	5.323	7.610	2.74%	0.31%
	7	18	5.704	5.594	5.809	28	75.49	5.809	8.007	5.843	7.988	0.58%	-0.24%
	8	20	6.813	6.690	6.971	27	102.75	6.974	7.315	7.055	7.327	1.16%	0.17%
	9	23	7.950	7.894	8.206	26	419.04	8.205	7.691	8.274	7.701	0.84%	0.12%
	10	25	8.671	8.65	8.875	25	183.02	8.870	7.441	8.933	7.446	0.71%	0.07%
6	5	15	5.717	5.714	6.599	28	197.67	6.602	8.074	6.857	8.134	3.86%	0.75%
	6	18	6.404	6.125	6.840	26	273.82	6.842	8.244	7.056	8.257	3.13%	0.16%
	7	21	8.140	7.545	7.975	14	271.02	7.976	8.125	8.074	8.159	1.23%	0.42%
	8	24	8.985	8.351	8.668	13	548.95	8.670	7.690	8.730	7.696	0.70%	0.08%
	9	27	9.913	9.366	9.758	14	589.12	9.763	7.122	9.878	7.173	1.18%	0.72%
	10	30	11.552	10.714	10.884	12	207.75	10.890	7.468	10.891	7.469	0.00%	0.02%

Note: customer preference scenario: 50%

Table 2. C.3 Performance of the extended APBFS algorithm, the SEML heuristics and lower bound for instances with large batches and 67% fill rate

				Extended APBFS						SEML		Gap	
T	*S*	*C*	lb	lb*	Opt	Solved	CPU(s)	Rel	AveWait (min)	Rel	AveWait (min)	Gap [Rel]	Gap [AveWait]
3	5	10	2.296	2.296	2.404	√	0.90	2.403	6.945	2.415	6.953	0.50%	0.12%
	6	12	3.113	3.113	3.239	√	1.21	3.245	7.468	3.277	7.477	0.99%	0.12%
	7	14	3.463	3.463	3.520	√	3.58	3.518	7.720	3.527	7.723	0.26%	0.04%
	8	16	4.100	4.100	4.176	√	3.36	4.179	7.281	4.186	7.282	0.17%	0.01%
	9	18	4.346	4.346	4.483	√	4.05	4.485	7.349	4.541	7.365	1.25%	0.22%
	10	20	4.717	4.717	4.755	√	3.40	4.755	6.937	4.760	6.940	0.11%	0.04%
4	5	13	4.127	4.127	4.635	√	27.60	4.631	8.342	4.829	8.368	4.28%	0.31%
	6	16	5.629	5.594	6.169	28	39.35	6.167	8.091	6.398	8.142	3.74%	0.62%
	7	19	5.754	5.728	6.318	28	175.06	6.307	7.310	6.397	7.320	1.43%	0.14%
	8	21	6.921	6.726	7.025	26	73.15	7.027	7.580	7.171	7.591	2.04%	0.15%
	9	24	8.229	7.859	8.201	24	320.01	8.208	7.655	8.314	7.663	1.30%	0.10%
	10	27	8.888	8.429	8.897	21	441.13	8.898	7.233	8.994	7.246	1.08%	0.18%
5	5	17	7.006	6.679	7.664	22	498.41	7.677	8.612	8.178	8.684	6.52%	0.84%
	6	20	7.677	6.863	7.398	16	151.72	7.402	7.596	7.703	7.644	4.07%	0.64%
	7	23	9.208	7.889	8.566	9	443.50	8.566	7.681	8.755	7.723	2.20%	0.54%
	8	27	10.838	10.271	10.696	6	1091.25	10.678	7.467	10.871	7.477	1.81%	0.13%
	9	30	12.546	10.219	10.914	4	858.35	10.912	7.368	10.955	7.381	0.39%	0.17%
	10	34	13.925	10.979	11.264	6	549.90	11.249	7.162	11.264	7.177	0.13%	0.21%
6	5	20	9.183	8.075	9.527	5	1567	9.529	8.042	10.463	8.162	9.80%	1.49%
	6	24	11.067	9.844	10.531	2	1019.19	10.506	7.841	10.590	7.871	0.80%	0.38%
	7	28	12.967	-	-	0	-	-	-	-	-	-	-
	8	32	14.829	11.75	11.781	1	694.38	11.745	8.051	11.740	8.050	-0.04%	-0.01%
	9	36	16.196	-	-	0	-	-	-	-	-	-	-
	10	40	18.935	-	-	0	-	-	-	-	-	-	-

Note: customer preference scenario: 50%

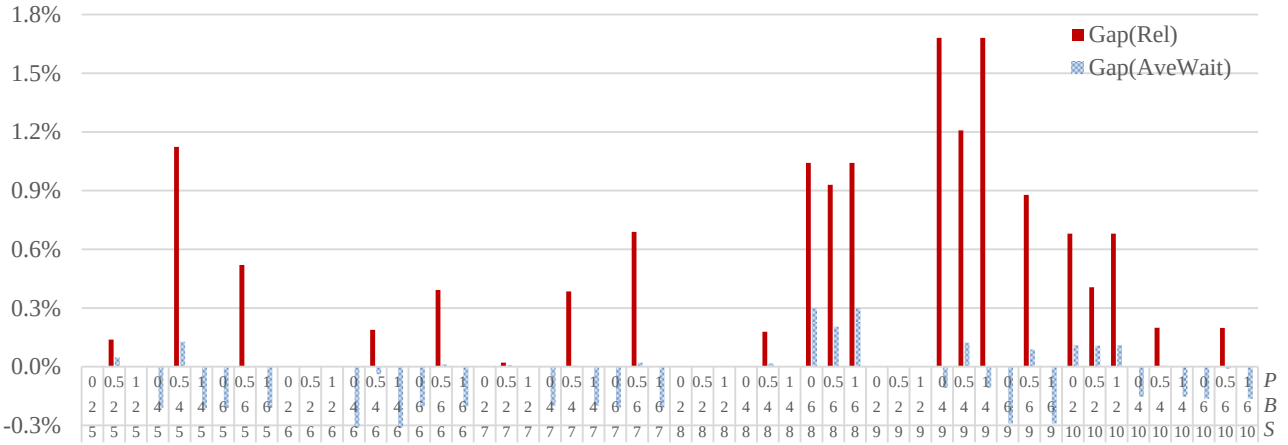
C.2. The calibrated results of Table 2. 2 for the comparison between the Sooo model and the Sooo extension model

Table 2. C.4. Comparison between the Sooo model and the Sooo extension model with small batches and 50% fill rate

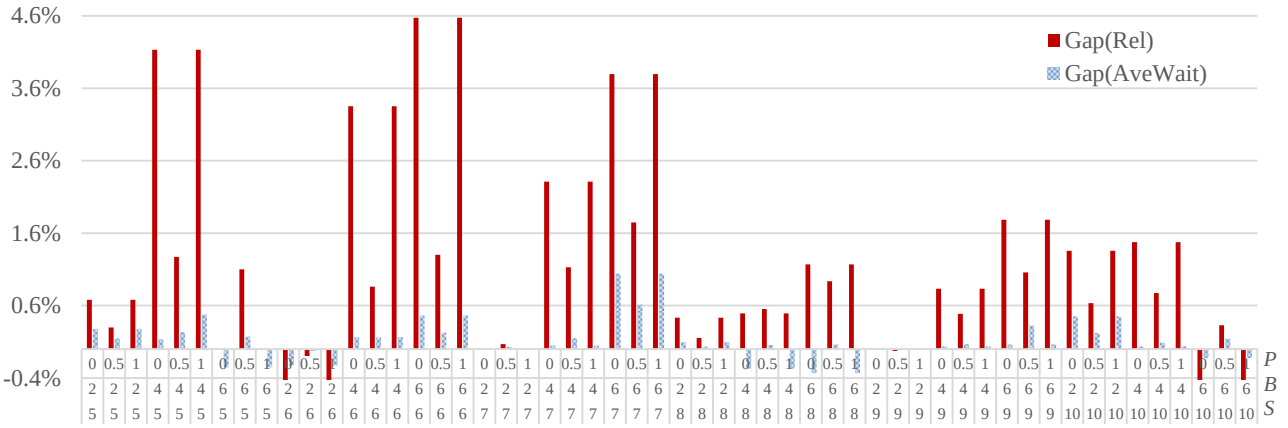
T	*S*	*C*	Sooo- APBFS				Sooo extension- extended APBFS				Gap [AveWait]
			Solved	CPU(s)	Rel	AveWait (min)	Solved	CPU(s)	Rel	AveWait (min)	
3	5	8	√	0.02	1.478	3.319	√	0.03	1.478	3.207	3.38%
	6	9	√	0.02	1.581	3.428	√	0.03	1.581	3.303	3.65%
	7	11	√	0.02	2.654	3.485	√	0.03	2.654	3.356	3.70%
	8	12	√	0.01	2.169	3.091	√	0.03	2.169	3.014	2.49%
	9	14	√	0.02	2.885	3.226	√	0.04	2.885	3.106	3.72%
	10	15	√	0.03	3.093	3.044	√	0.06	3.093	2.930	3.77%
4	5	10	√	0.02	2.855	3.534	√	0.03	2.855	3.411	3.49%
	6	12	√	0.03	3.466	3.532	√	0.05	3.466	3.351	5.12%
	7	14	√	0.04	3.941	3.485	√	0.05	3.941	3.324	4.61%
	8	16	√	0.16	4.559	3.556	√	0.20	4.559	3.390	4.67%
	9	18	√	0.29	5.523	3.691	√	0.34	5.523	3.474	5.87%
	10	20	√	0.72	6.015	3.334	√	0.80	6.015	3.181	4.59%
5	5	13	√	0.16	4.883	4.042	√	0.19	4.884	3.900	3.53%
	6	15	√	2.44	5.544	3.708	√	2.98	5.544	3.553	4.19%
	7	18	√	0.72	6.573	3.993	√	0.77	6.573	3.777	5.39%
	8	20	√	7.57	7.516	3.695	√	8.16	7.516	3.482	5.77%
	9	23	29	67.35	8.696	3.835	29	68.25	8.696	3.606	5.96%
	10	25	29	39.40	9.238	3.719	29	49.37	9.238	3.517	5.43%
6	5	15	√	4.56	6.999	4.034	√	5.06	6.999	3.886	3.66%
	6	18	√	5.48	7.728	4.162	√	6.18	7.728	3.959	4.89%
	7	21	23	*148.91*	*8.923*	*4.006*	22	160.87	8.923	3.787	5.46%
	8	24	22	150.07	9.881	3.998	22	164.07	9.882	3.748	6.24%
	9	27	18	100.24	10.538	3.661	18	104.47	10.538	3.419	6.60%
	10	30	19	*108.42*	*11.576*	*3.804*	18	108.52	11.576	3.580	5.90%

*Note: customer preference scenario: 50%

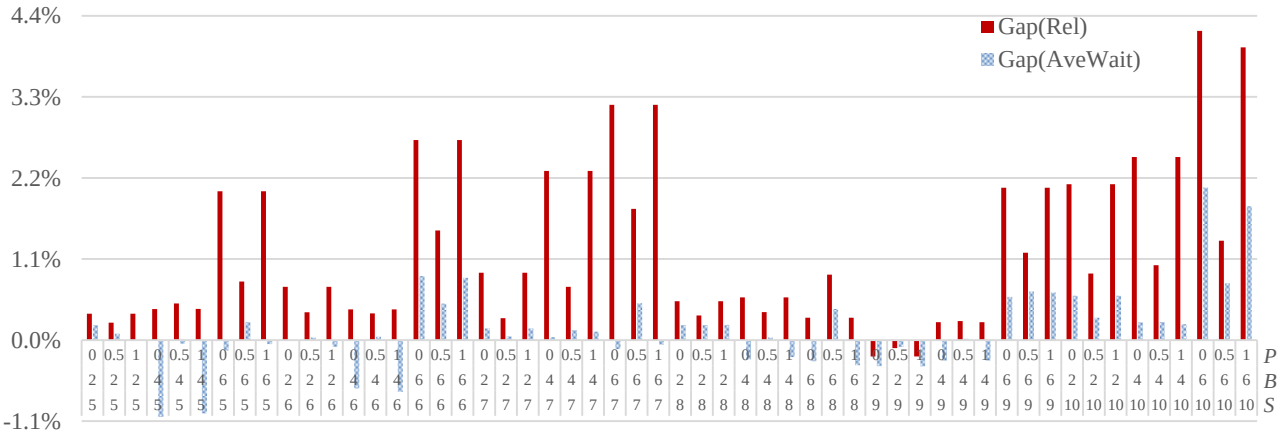
Appendix D. Comparisons between the SEM heuristic and the SEML heuristic



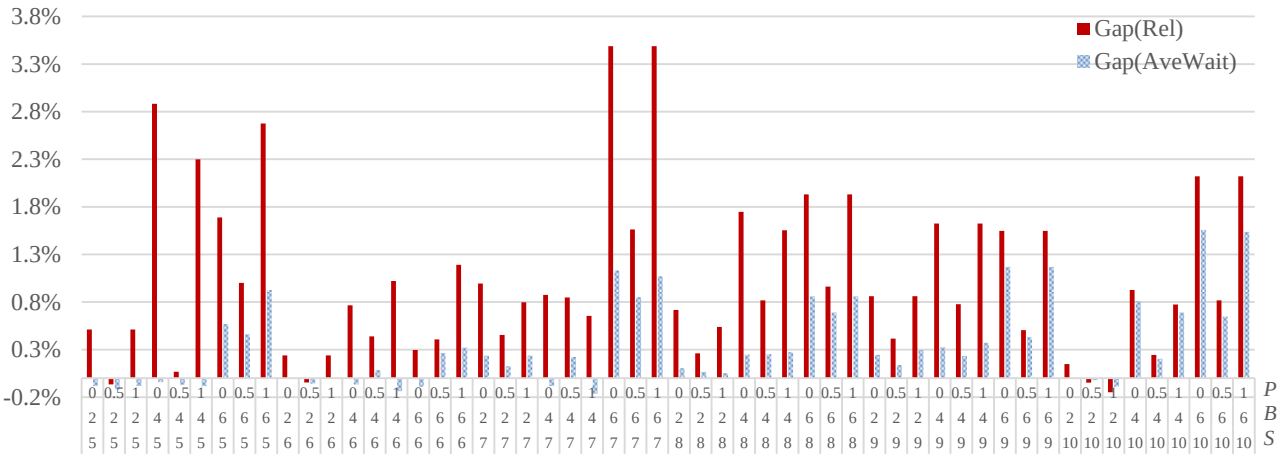
(a) Instances of three tiers



(b) Instances of four tiers



(c) Instances of five tiers



(d) Instances of six tiers

Fig. 2. D.1. Comparisons between the SEM heuristic and the SEML heuristic for all the instances with 67% fill rate

Appendix E. Additional results for the effectiveness of the flexible service policy

E.1. Absolute reduction on the total number of relocations

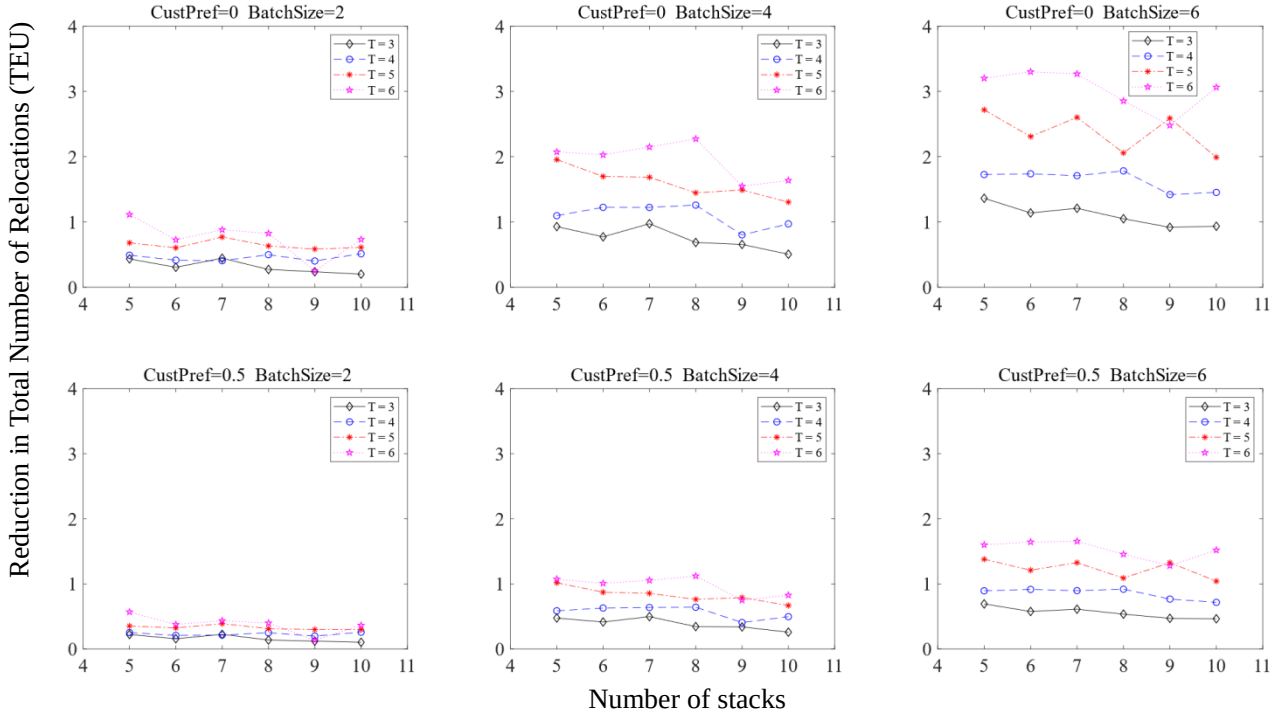


Fig. 2. E.1 (a) Reduction on the total number of relocations by the flexible service policy for instances of 50% fill rate

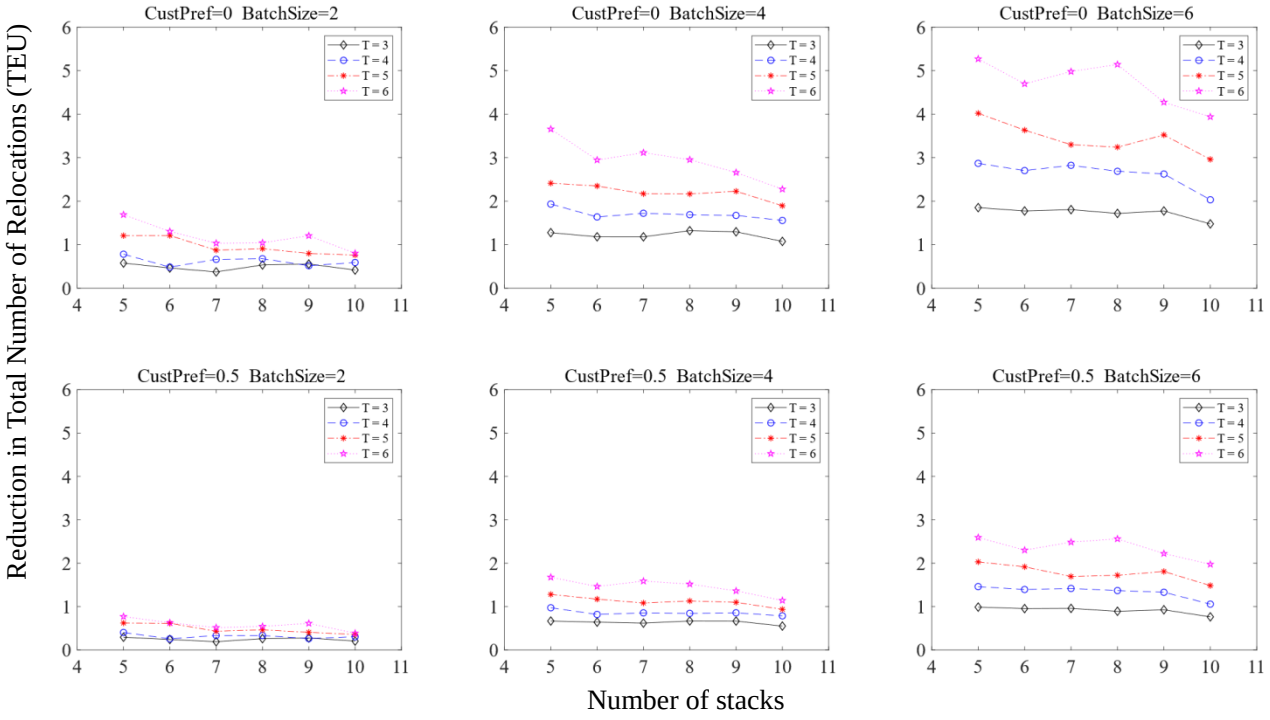


Fig. 2. E.1 (b) Reduction on the total number of relocations by the flexible service policy for instances of 67% fill rate

E.2. Absolute reduction on the average relevant truck waiting time

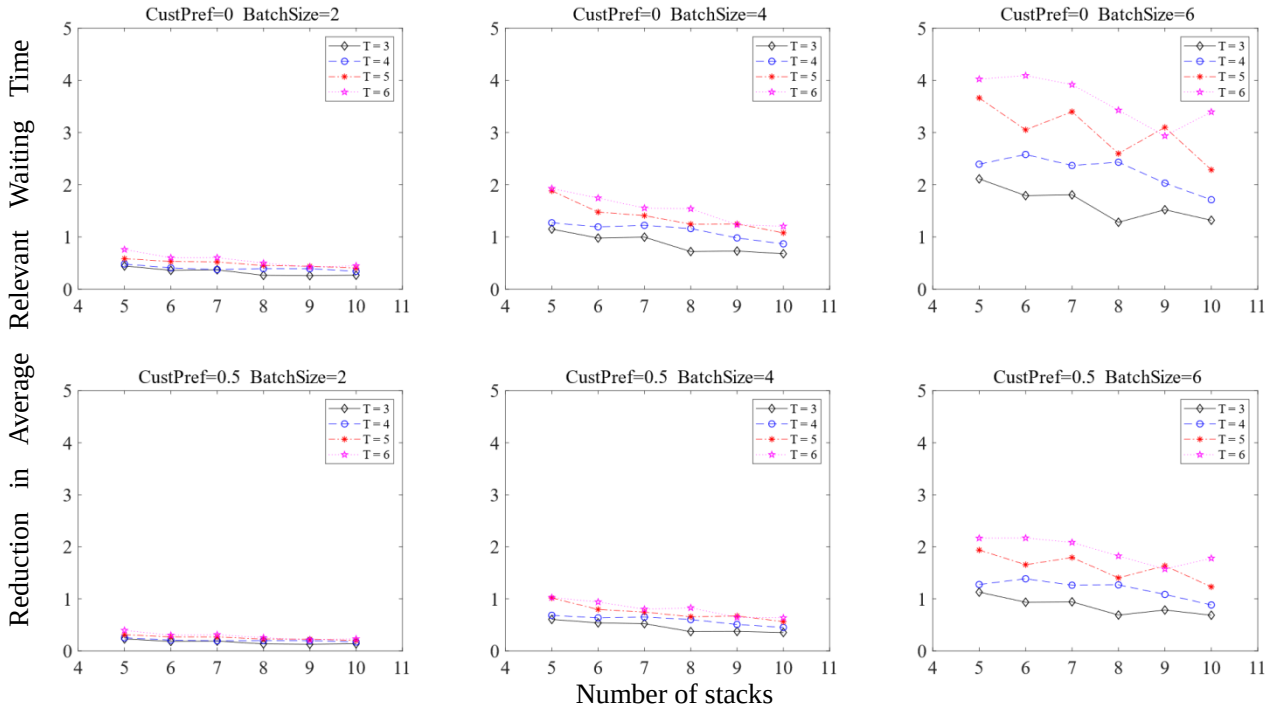


Fig. 2. E.2 (a) Reduction on the average relevant waiting time by the flexible service policy for instances of 50% fill rate

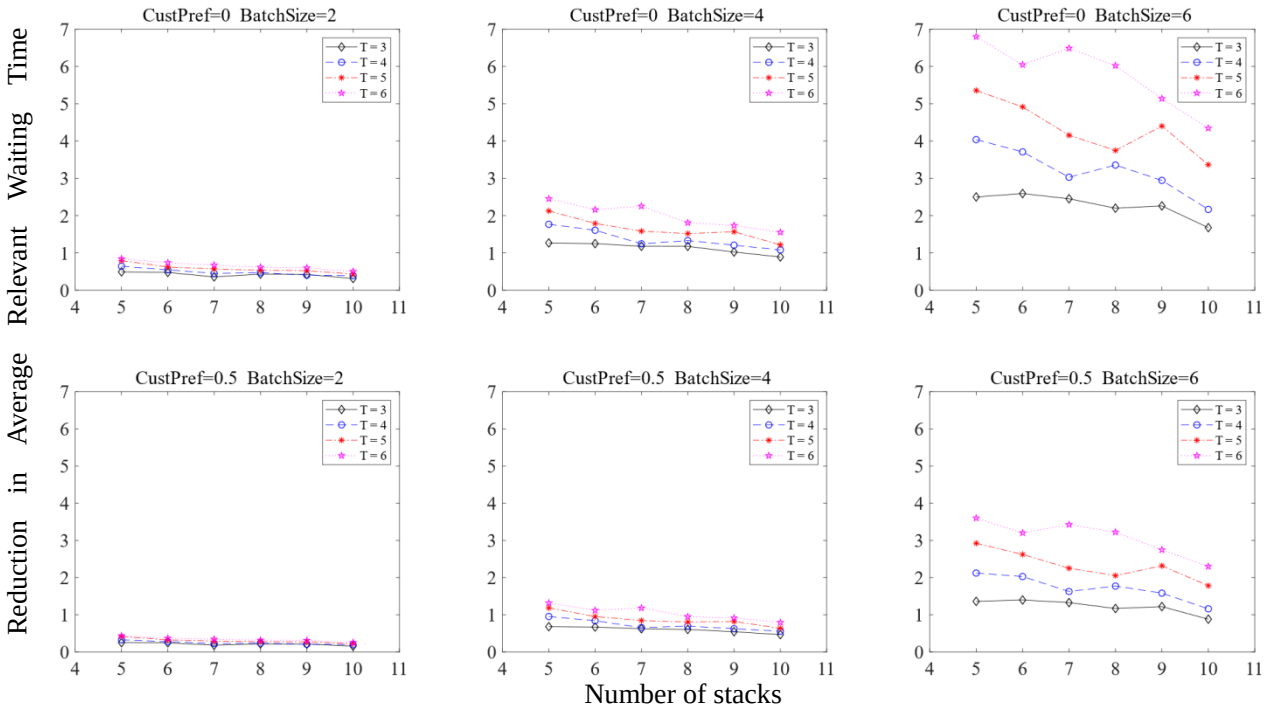


Fig. 2. E.2 (b) Reduction on the average relevant waiting time by the flexible service policy for instances of 67% fill rate

E.3. Relative reduction on the average relevant truck waiting time

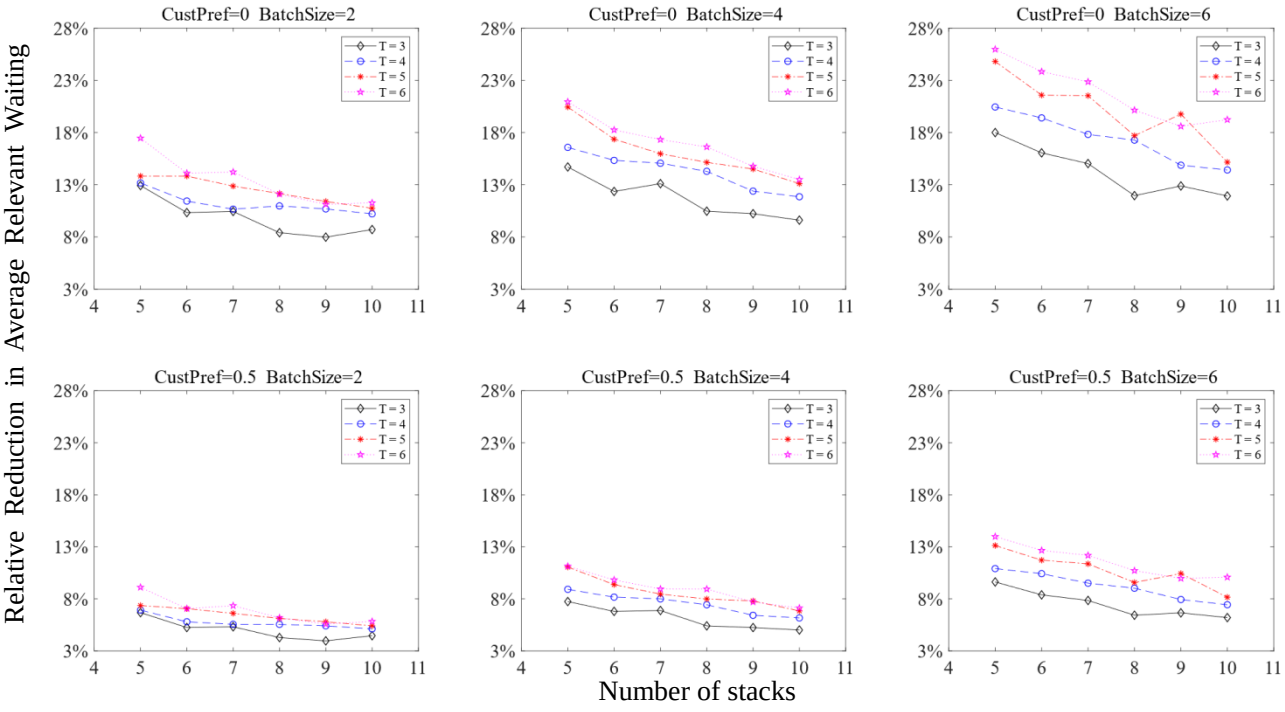


Fig. 2. E.3 (a) Effect of the flexible service policy on average relevant waiting time for instances of 50% fill rate

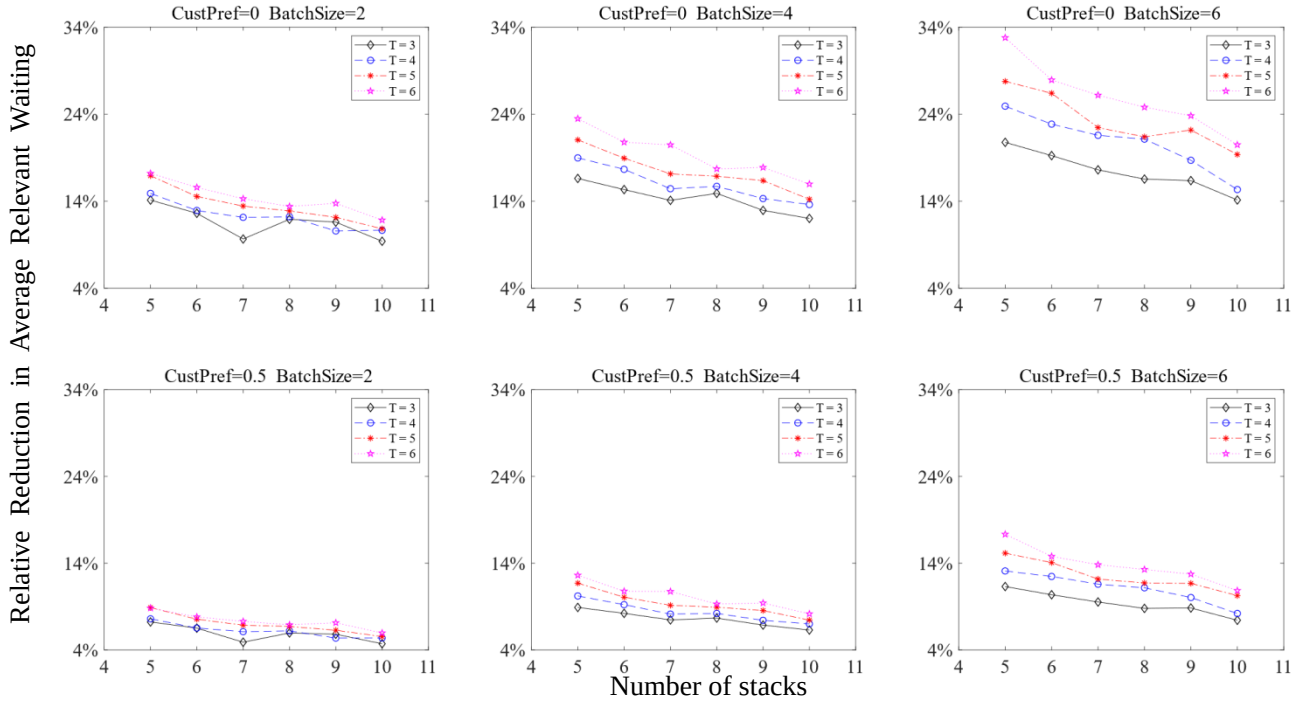


Fig. 2. E.3 (b) Effect of the flexible service policy on average relevant waiting time for instances of 67% fill rate

E.4. Average truck turn time

Table 2. E.1 Results of the average truck turn time for all instances under the flexible service policy

			Fill rate = 50%				*Fill rate* = 67%			
S	*B*	*P*	*T*=3	*T*=4	*T*=5	*T*=6	*T*=3	*T*=4	*T*=5	*T*=6
5	2	0	13.995	14.178	14.652	14.590	13.981	14.660	14.894	15.052
	2	0.5	21.709	21.915	22.396	22.393	21.731	22.467	22.764	22.975
	2	1	28.995	29.178	29.631	29.577	28.981	29.660	29.894	30.049
	4	0	17.676	17.402	18.330	18.276	17.370	18.551	18.984	18.983
	4	0.5	25.717	25.492	26.678	26.682	25.452	26.875	27.434	27.629
	4	1	32.676	32.402	33.315	33.276	32.370	33.525	33.956	33.967
	6	0	20.630	20.320	22.102	22.467	20.532	23.169	24.930	24.936
	6	0.5	29.109	28.927	31.329	31.828	29.179	32.577	34.865	35.674
	6	1	35.630	35.320	37.102	37.467	35.532	38.169	39.918	39.906
6	2	0	14.125	14.142	14.296	14.669	14.305	14.667	14.648	14.983
	2	0.5	21.804	21.859	22.055	22.454	22.038	22.437	22.459	22.851
	2	1	29.125	29.142	29.296	29.669	29.305	29.667	29.648	29.983
	4	0	17.944	17.594	18.031	18.821	17.892	18.495	18.653	19.230
	4	0.5	25.899	25.640	26.209	27.128	25.975	26.745	26.998	27.769
	4	1	32.944	32.594	33.031	33.821	32.892	33.495	33.653	34.221
	6	0	20.377	21.716	22.086	24.077	21.880	23.528	24.706	26.605
	6	0.5	28.733	30.411	30.961	33.468	30.576	32.725	34.531	36.959
	6	1	35.377	36.716	37.086	39.077	36.880	38.528	39.706	41.485
7	2	0	14.175	14.160	14.528	14.639	14.331	14.258	14.670	15.027
	2	0.5	21.852	21.829	22.277	22.295	22.003	21.976	22.455	22.861
	2	1	29.175	29.160	29.528	29.626	29.331	29.258	29.670	30.008
	4	0	17.613	17.892	18.410	18.411	18.174	17.818	18.652	19.758
	4	0.5	25.582	25.977	26.580	26.658	26.221	25.893	26.891	28.330
	4	1	32.613	32.892	33.410	33.408	33.174	32.818	33.637	34.751
	6	0	21.210	21.924	23.390	24.223	22.480	22.009	25.351	29.294
	6	0.5	29.559	30.513	32.495	33.544	31.099	30.922	34.755	39.858
	6	1	36.210	36.924	38.390	39.223	37.480	37.009	40.325	44.297
8	2	0	13.881	14.188	14.271	14.625	14.172	14.377	14.612	14.973
	2	0.5	21.514	21.891	21.999	22.247	21.889	22.115	22.374	22.779
	2	1	28.881	29.188	29.271	29.614	29.172	29.377	29.607	29.969
	4	0	17.154	17.964	17.976	18.730	17.706	18.142	18.487	19.407
	4	0.5	24.989	26.001	26.062	26.947	25.777	26.277	26.712	27.788
	4	1	32.154	32.964	32.976	33.730	32.706	33.142	33.478	34.399
	6	0	20.435	22.649	23.074	24.595	22.098	23.526	24.767	29.260
	6	0.5	28.524	31.295	31.759	33.710	30.625	32.603	33.968	39.559
	6	1	35.435	37.649	38.074	39.595	37.098	38.526	39.767	44.264
9	2	0	13.979	14.280	14.413	14.382	14.206	14.415	14.764	14.757
	2	0.5	21.610	21.981	22.111	21.918	21.917	22.125	22.521	22.542
	2	1	28.979	29.280	29.413	29.382	29.206	29.415	29.757	29.751
	4	0	17.411	17.937	18.368	18.120	17.885	18.235	19.012	18.968
	4	0.5	25.265	25.906	26.439	26.213	25.866	26.314	27.258	27.298
	4	1	32.411	32.937	33.368	33.120	32.885	33.235	34.006	33.959
	6	0	21.281	22.632	23.580	23.853	22.558	23.807	26.444	27.431
	6	0.5	29.524	31.075	32.547	32.706	31.083	32.673	36.034	37.316
	6	1	36.281	37.632	38.580	38.853	37.558	38.807	41.430	42.431
10	2	0	13.805	14.016	14.343	14.524	14.026	14.271	14.573	14.751
	2	0.5	21.436	21.686	22.020	22.074	21.692	21.963	22.287	22.502
	2	1	28.805	29.016	29.343	29.518	29.026	29.271	29.571	29.747

4	0	17.383	17.450	18.151	18.725	17.516	17.854	18.340	19.175
4	0.5	25.208	25.376	26.165	26.799	25.442	25.880	26.421	27.433
4	1	32.383	32.450	33.151	33.720	32.516	32.854	33.338	34.173
6	0	20.750	21.174	23.791	25.256	21.192	22.956	25.013	27.868
6	0.5	28.878	29.498	32.363	34.376	29.493	31.476	34.084	37.408
6	1	35.750	36.174	38.791	40.256	36.192	37.956	40.005	42.872

Appendix F. Results of three sets of customer preference scenarios

Table 2. F.1 Comparison between three sets of customer preference scenarios for small batches and 50% fill rate

T	*S*	*C*	'100%' preference scenario				'50%' preference scenario				Heterogeneous preference scenario			
			Solved	CPU(s)	Rel	AveWait	Solved	CPU(s)	Rel	AveWait	Solved	CPU(s)	Rel	AveWait
3	5	8	√	0.04	1.267	2.992	√	0.03	1.478	3.207	√	0.02	1.475	3.205
	6	9	√	0.03	1.433	3.126	√	0.03	1.581	3.303	√	0.03	1.593	3.306
	7	11	√	0.03	2.433	3.170	√	0.03	2.654	3.356	√	0.04	2.638	3.348
	8	12	√	0.03	2.033	2.883	√	0.03	2.169	3.014	√	0.04	2.158	3.017
	9	14	√	0.05	2.767	2.976	√	0.04	2.885	3.106	√	0.05	2.885	3.111
	10	15	√	0.06	3.000	2.800	√	0.06	3.093	2.930	√	0.07	3.088	2.939
4	5	10	√	0.04	2.633	3.180	√	0.03	2.855	3.411	√	0.04	2.862	3.418
	6	12	√	0.05	3.267	3.144	√	0.05	3.466	3.351	√	0.06	3.475	3.355
	7	14	√	0.07	3.733	3.148	√	0.05	3.941	3.324	√	0.08	3.944	3.340
	8	16	√	0.15	4.300	3.192	√	0.20	4.559	3.390	√	0.31	4.533	3.387
	9	18	√	0.37	5.333	3.274	√	0.34	5.523	3.474	√	0.76	5.531	3.477
	10	20	√	0.27	5.767	3.010	√	0.80	6.015	3.181	√	2.65	6.003	3.187
5	5	13	√	0.08	4.500	3.621	√	0.19	4.884	3.900	√	0.32	4.856	3.884
	6	15	√	0.20	5.200	3.307	√	2.98	5.544	3.553	√	34.94	5.552	3.565
	7	18	√	0.28	6.200	3.519	√	0.77	6.573	3.777	√	2.29	6.562	3.783
	8	20	√	1.43	7.200	3.267	√	8.16	7.516	3.482	√	34.67	7.501	3.484
	9	23	√	*5.00*	*8.414*	*3.388*	29/30	68.25	8.696	3.606	284/300	151.57	8.700	3.615
	10	25	√	*7.33*	*8.931*	*3.321*	29/30	49.37	9.238	3.517	285/300	158.48	9.235	3.529
6	5	15	√	1.25	6.400	3.560	√	5.06	6.999	3.886	√	15.69	7.008	3.897
	6	18	√	0.99	7.233	3.670	√	6.18	7.728	3.959	299/300	65.82	7.727	3.963
	7	21	27/30	*44.46*	*8.455*	*3.506*	22/30	160.87	8.923	3.787	208/300	*260.84*	*8.843*	*3.795*
	8	24	25/30	*23.52*	*9.333*	*3.500*	22/30	*69.554*	*9.758*	*3.725*	208/300	184.58	9.751	3.734
	9	27	24/30	*48.71*	*10.389*	*3.235*	18/30	104.47	10.538	3.419	173/300	194.41	10.530	3.428
	10	30	23/30	*38.75*	*11.333*	*3.404*	18/30	108.52	11.572	3.580	-	-	-	-

Acknowledgments

The authors thank three anonymous reviewers' constructive comments that have helped improve the presentation of the paper. We also thank Ku and Arthanari (2016a) and Galle et al. (2018b) for their published source data. This study is partially supported by the China Scholarship Council, the Royal Society (Grant No. IEC\NSFC\170100), the EU H2020 (Grant No. 777742, EC H2020-MSCA-RISE-2017), the National Natural Science Foundation of China (Grant No. 71671021), and Fundamental Research Funds for the Central Universities.

References

Bacci, T., Mattia, S., & Ventura, P. (2019). The bounded beam search algorithm for the block relocation problem. Computers & Operations Research, 103, 252-264.

Bakker, H., Dunke, F., & Nickel, S. (2020). A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. Omega, 96, 102080.

Birge, J.R., Louveaux, F. (2011). Introduction to stochastic programming (2nd ed.). Springer, New York.

Bonney, J. (2015). US ports move toward truck appointment model. JOC.com. Apr 27, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-new-york-and-new-jersey/us-ports-move-toward-truck-appointment-model_20150427.html. Accessed February 23, 2020.

Borjian, S., Manshadi, V., Barnhart, C., & Jaillet, P. (2013). Dynamic Stochastic Optimization of Reshuffles in Container Terminals. In Manufacturing and Service Operations Management (MSOM) conference.

Borjian, S., Galle, V., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015a). Container relocation problem: Approximation, asymptotic, and incomplete information. arXiv preprint 1505.04229.

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015b). Managing relocation and delay in container terminals with flexible service policies. arXiv preprint 1503.01535.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. European journal of operational research, 235(2), 412-430.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. European Journal of Operational Research, 219(1), 96-104.

Caserta, M., Schwarze, S., & Voß, S. (2011a). Container Rehandling at Maritime Container Terminals. In J. W. Böse (Ed.), *Handbook of Terminal Planning*, Operations research/computer science interfaces series 49 (pp. 247-269). Springer, New York.

Caserta, M., Voß, S., & Sniedovich, M. (2011b). Applying the corridor method to a blocks relocation problem. OR Spectrum, 33(4), 915-929.

Chen, G., Govindan, K., & Golias, M. M. (2013a). Reducing truck emissions at container terminals in a low carbon economy: Proposal of a queueing-based bi-objective model for optimizing truck arrival pattern. Transportation Research Part E: Logistics and Transportation Review, 55, 3-22.

Chen, G., Govindan, K., & Yang, Z. (2013b). Managing truck arrivals with time windows to alleviate gate congestion at container terminals. International Journal of Production Economics, 141(1), 179-188.

Davies, P. (2009). Container Terminal Reservation Systems: Paper presented at the 3rd Annual METRANS National Urban Freight Conference. Long Beach, USA.

de Melo da Silva, M., Erdoğan, G., Battarra, M., & Strusevich, V. (2018). The block retrieval problem. European Journal of Operational Research, 265(3), 931-950.

Dragović, B., Tzannatos, E., & Park, N. K. (2017). Simulation modelling in ports and container terminals: literature overview and analysis by research field, application area and tool. Flexible Services and Manufacturing Journal, 29(1), 4-34.

DP World. Retrievd from https://www.londongateway.com/port/book-a-vehicle. Accessed February 28, 2020.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2014). A domain-specific knowledge-based heuristic for the blocks relocation problem. Advanced Engineering Informatics, 28(4), 327-343.

fenixmarineservices.com. Retrieved from https://www.fenixmarineservices.com/terminal/#appointments. Accessed February 21, 2020.

Galle, V., Manshadi, V. H., Boroujeni, S. B., Barnhart, C., & Jaillet, P. (2018a). The stochastic container relocation problem. Transportation Science, 52(5), 1035-1058.

Galle, V., Barnhart, C., & Jaillet, P. (2018b). Yard Crane Scheduling for container storage, retrieval, and relocation. European Journal of Operational Research, 271(1), 288-316.

Giuliano, G., & O'Brien, T. (2007). Reducing port-related truck emissions: The terminal gate appointment system at the Ports of Los Angeles and Long Beach. Transportation Research Part D: Transport and Environment, 12(7), 460-473.

Gharehgozli, A. H., Roy, D., & De Koster, R. (2016). Sea container terminals: New technologies and OR models. Maritime Economics & Logistics, 18(2), 103-140.

Huynh, N., Walton, C. M., & Davis, J. (2004). Finding the number of yard cranes needed to achieve desired truck turn time at marine container terminals. Transportation research record, 1873(1), 99-108.

Huynh, N., & Zumerchik, J. (2010). Analysis of stacking priority rules to improve drayage operations using existing and emerging technologies. Transportation research record, 2162(1), 1-8.

Jovanovic, R., & Voß, S. (2014). A chain heuristic for the blocks relocation problem. Computers & Industrial Engineering, 75, 79-86.

Jin, B., Zhu, W., & Lim, A. (2015). Solving the container relocation problem by an improved greedy look-ahead heuristic. European Journal of Operational Research, 240(3), 837-847.

Kim, K. H., & Hong, G. P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research, 33(4), 940-954.

Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 124(1), 89-101.

Ku, D., & Arthanari, T. S. (2016a). Container relocation problem with time windows for container departure. European Journal of Operational Research, 252(3), 1031-1039.

Ku, D., & Arthanari, T. S. (2016b). On the abstraction method for the container relocation problem. Computers & Operations Research, 68, 110-122.

Lee, Y., & Lee, Y. J. (2010). A heuristic for retrieving containers from a yard. Computers & Operations Research, 37(6), 1139-1147.

Lee, C. Y., & Song, D. P. (2017). Ocean container transport in global supply chains: Overview and research opportunities. Transportation Research Part B: Methodological, 95, 442-474.

Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. European Journal of Operational Research, 239(2), 297-312.

Lin, D. Y., Lee, Y. J., & Lee, Y. (2015). The container retrieval problem with respect to relocation. Transportation Research Part C: Emerging Technologies, 52, 132-143.

Li, N., Chen, G., Ng, M., Talley, W. K., & Jin, Z. (2020). Optimized appointment scheduling for export container deliveries at marine terminals. Maritime Policy & Management, 47(4), 456-478.

López-Plata, I., Expósito-Izquierdo, C., Lalla-Ruiz, E., Melián-Batista, B., & Moreno-Vega, J. M. (2017). Minimizing the Waiting Times of block retrieval operations in stacking facilities. Computers & Industrial Engineering, 103, 70-84.

Mongelluzzo, B. (2016). Long Beach automated terminal expects fastest harbor truck turns, JOC.com. Feb 11, 2016. Retrieved from https://www.joc.com/port-news/us-ports/port-long-beach/long-beach-automated-terminal-expects-fastest-harbor-truck-turns_20160211.html. Accessed February 21, 2020.

Mongelluzzo, B. (2019). LA-LB truckers: We need true interoperable chassis pools, JOC.com, Oct 29 2019. Retrieved from https://www.joc.com/port-news/la-lb-truckers-we-need-true-interoperable-chassis-pools_20191029.html. Accessed February 29, 2020.

Mongelluzzo, B. (2020). Technology, mandatory truck slots push LA-LB turn times to near six-year low, JOC.com, Jan 13 2020. Retrieved from https://www.joc.com/port-news/terminal-operators/technology-mandatory-truck-slots-push-la-lb-turn-times-almost-six-year-low_20200113.html. Accessed February 22.

Patrick. Retrieved from http://www.patrick.com.au/documents/VBS-Charges-Melbourne-July-2018.pdf. Accessed February 28, 2020.

Petering, M. E., & Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. European Journal of Operational Research, 231(1), 120-130.

Pham, Q., Huynh, N., & Xie, Y. (2011). Estimating truck queuing time at marine terminal gates. Transportation research record, 2222(1), 43-53.

Port Botany. Retrieved from https://www.adventintermodal.com/customers/port-botany-new-south-wales-australia/. Accessed February 29, 2020.

Port Metro Vancouver. Retrieved from https://cleanairactionplan.org/documents/final-2017-clean-air-action-plan-update.pdf/. Accessed February 21, 2020.

Quispe, K. E. Y., Lintzmayer, C. N., & Xavier, E. C. (2018). An exact algorithm for the blocks relocation problem with new lower bounds. Computers & Operations Research, 99, 206-217.

Russell, S.J., & Norvig, P. (2016). Artificial intelligence: a Modern Approach (3rd ed.) (pp. 80). Pearson Education, Limited.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: a literature update. OR Spectrum, 30(1), 1-52.

Saanen, Y. A. (2011). Modeling techniques in planning of terminals: The quantitative approach. In J. W. Böse (Ed.), *Handbook of Terminal Planning*, Operations research/computer science interfaces series 49 (pp. 83–102). Springer, New York.

Talley, W. K., & Ng, M. (2016). Port multi-service congestion. Transportation Research Part E: Logistics and Transportation Review, 94, 66-70.

Tanaka, S., & Takii, K. (2016). A faster branch-and-bound algorithm for the block relocation problem. IEEE Transactions on Automation Science and Engineering, 13(1), 181-190.

Tang, L., Jiang, W., Liu, J., & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions, 47(7), 751-766.

Ting, C. J., & Wu, K. C. (2017). Optimizing container relocation operations at container yards with beam search. Transportation Research Part E: Logistics and Transportation Review, 103, 17-31.

Tong, X., Woo, Y. J., Jang, D. W., & Kim, K. H. (2015). Heuristic Rules Based on a Probabilistic Model and a Genetic Algorithm for Relocating Inbound Containers with Uncertain Pickup Times. International Journal of Industrial Engineering, 22, 93–101.

Ünlüyurt, T., & Aydın, C. (2012). Improved rehandling strategies for the container retrieval process. Journal of Advanced Transportation, 46(4), 378-393.

Wan, Y. W., Liu, J., & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack. Naval Research Logistics (NRL), 56(8), 699-713.

Yang, M., Allen, T. T., Fry, M. J., & Kelton, W. D. (2013). The call for equity: simulation optimization models to minimize the range of waiting times. IIE Transactions, 45(7), 781-795.

Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. European Journal of Operational Research, 245(2), 415-422.

Zehendner, E., & Feillet, D. (2014). A branch and price approach for the container relocation problem. International Journal of Production Research, 52(24), 7159-7176.

Zehendner, E., Feillet, D., & Jaillet, P. (2017). An algorithm with performance guarantee for the online container relocation problem. European Journal of Operational Research, 259(1), 48-62.

Zeng, Q., Feng, Y., & Yang, Z. (2019). Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering, 127, 366-382.

Zhang, C., Chen, W., Shi, L., & Zheng, L. (2010). A note on deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 205(2), 483-485.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. Transportation Research Part E: Logistics and Transportation Review, 46(3), 327-343.

Zhen, L., Jiang, X., Lee, L. H., & Chew, E. P. (2013). A review on yard management in container terminals. Industrial Engineering and Management Systems, 12(4), 289-304.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. IEEE Transactions on Automation Science and Engineering, 9(4), 710-722.

Zweers, B. G., Bhulai, S., & van der Mei, R. D. (2020). Optimizing pre-processing and relocation moves in the stochastic container relocation problem. European Journal of Operational Research, 283(3), 954-971.

Chapter 3

Service fairness and value of customer information for the stochastic container relocation problem under flexible service policy

Abstract: The Stochastic Container Relocation Problem (SCRP) is to find a sequence of moves to retrieve all containers from a bay by minimising the expected number of relocations, where each container has been booked to a time window in which a truck will arrive randomly to retrieve the container. Applying a flexible service policy, under which the trucks arriving within the same time window are allowed to be served out-of-order, the expected number of relocations can be significantly reduced compared with the first-come-first-serve service policy. However, the flexible policy may cause some earlier arriving trucks to be served later than some later arriving trucks, which raises the concern of service fairness. In addition, a question arises as to whether the information of trucks' arrival probability over the time window (which represents the customer preference information) would add value to the terminal operators. In this paper, we incorporate the concept of service fairness into the SCRP in two phases. In phase 1, we propose a multiple sub-windows-based flexible service policy (SCRP-MFS), under which the time window is divided into multiple sub-windows and the flexible service policy is only applied to each individual sub-window. In phase 2, the SCRP-MFS is formulated into a stochastic dynamic programming model with two lexicographically ordered objectives representing both relocation efficiency and service fairness, which is solved via a hierarchical iterative approach. Extensive computational experiments are conducted to evaluate the impacts of the number of sub-time windows and to examine the impacts and value of customer preference information in various scenarios. The results show interesting trade-offs between efficiency and fairness. As the number of sub-time windows increases, the service fairness is generally improving (but not guaranteed) while the expected number of relocations is increasing. Hence, the model is useful to balance the performance measures from both the terminal operators' side and the trucks' side by finding an appropriate number of sub-windows. It is also found that the customer preference information can be valuable in some circumstances, especially when each truck indicates a certain arrival sub-time window. This indicates that the model can provide terminal operators with insights into whether it is worth committing effort to capture such information. [1]

[1] Feng, Y., Song, D.P., Li, D., & Xie, Y. (2021). Service fairness and value of customer information for the stochastic container relocation problem under flexible service policy, Transportation Research Part E: Logistics and Transportation Review, under the second-round review.

Keywords: OR in port operations, stochastic container relocation problem, flexible service policy, service fairness, value of information

3.1 Introduction

Maritime container ports/terminals, where containers are transferred between different transport modes, are an indispensable part of the global container shipping network, and their handling productivities and efficiencies are essential to ensure efficient container transport logistics. There have been many studies on improving the productivity and operational efficiencies of container terminals, as reported in the survey paper by Stahlbock and Voß (2008) and Carlo et al. (2014). The disruptions caused by the COVID-19 pandemic and the Suez Canal blockage have exacerbated port congestion and global supply chain delays. This has led to further overloading of already insufficient yard and gate handling capacity, with further inefficiencies in releasing cargos for hinterland transportation (Notteboom et al., 2021) and a bigger variation of truck turnaround times. To help to address these issues, this research focuses on improving the performance of the import container retrieval operations at container terminals.

Container relocation (also known as reshuffling or rehandling) is a major source of inefficiency in the import container retrieval operations in most container terminals (Ku and Arthanari, 2016a). It is a common practice that import containers are stored in the storage yard after being discharged from ships, waiting to be picked up by external trucks or loaded onto trains for inland transport. Due to limited space in the storage yard, containers are piled up vertically in a set of stacks, which can be up to six tiers high (Caserta et al., 2012). Container relocation occurs when the target container is buried underneath other containers and these blocking containers have to be relocated to other stacks in order to retrieve the target container. The container relocation operations are costly to terminal operators and cause delays to external trucks when retrieving containers.

The Container Relocation Problem (CRP) has been extensively studied, which focuses on finding a sequence of moves to retrieve all containers from a yard bay with a minimum number of relocations, in response to certain container departure priorities. In reality, the departure priorities of import containers are subject to high uncertainties due to the unpredictable arrival times of the external trucks. Ports would know the arrival precedence of trucks if the trucks book specific time windows in the truck appointment system or the vehicle booking system, but the specific arrival order of the trucks booked in the same time window is still uncertain. The CRP that considers the randomness of truck arrivals in the same time window is termed as the Stochastic Container Relocation Problem (SCRP) (Galle et al., 2018) or the CRP with Time Windows (CRPTW) (Ku and Arthanari, 2016a) in the literature. In the majority of the existing studies on the CRP or SCRP, import containers are retrieved on the first come first served (FCFS) basis, that is, containers are retrieved in the arriving order of their designated trucks. A few studies (e.g., Borjian et al., 2015; Zeng et al. 2019; Feng et al., 2020) introduce the flexible service policies, under which the arriving external trucks are allowed to be served out-of-order. The flexible service policies provide the terminal operators new opportunities to determine the container retrieval sequence so that the number of relocations can be further reduced compared to the FCFS policy.

However, the flexible policy might be perceived as unfair to some of the trucks involved, because the out-of-order service may cause some early arriving trucks to be served later than some later arriving trucks. In addition, it may also lead to a larger variation of the truck turnaround times, which further differentiates the customers' experience at container terminals. As a result, the issue of service fairness to customers (i.e., trucks) may become a barrier to the implementation of the flexible service policy. The fairness concerns in

a general sense have been emphasized in the operations research and management science literature and across multiple industrial applications, e.g. urban transportation systems (Li et al., 2019; Wu et al., 2021), vehicle routing problems (Matl et al., 2018), healthcare appointment systems (Qi, 2017), resource allocation problems (Bertsimas et al. 2011, 2013), and vessel scheduling at ports (Zhang et al., 2017; Wang et al., 2017; Jia and Meng, 2021). However, no studies have explicitly treated service fairness as part of objective functions to be optimised in the CRP and SCRP contexts. To the best of our knowledge, only two papers (Borjian et al., 2015; Feng et al., 2020) have mentioned service equity (fairness) and evaluated the impact of the relocation-minimisation-oriented solutions on the service equity in the CRP and SCRP. The consideration of fairness issues in the container retrieval service can bring tangible benefits to port operators. Consistently meeting truck turnaround time would enhance the credibility of the appointment system, as well as improving service reliability and customer satisfaction. The National Retail Federation made a recommendation to the U.S. Secretary of Transportation that the range for truck turnaround time should be listed as one of the port performance metrics (Knatz, 2017). The reduction on the maximum truck turnaround time could enhance a port's performance. In this paper, the service fairness refers to avoiding excessively long turnaround time for any truck, in other words, keeping the expected maximum truck turnaround time as small as possible. In the SCRP context, one measure to mitigate the service unfairness among trucks is to divide the appointment window into smaller sub-time windows and only apply the flexible service policy for the trucks arriving within individual sub-time windows. This would narrow the scope of out-of-order services and achieve the trade-off between the relocation efficiency and the service fairness. Another measure is to embed the service fairness objective into the relocation minimisation procedure so that not only the number of relocations will be minimised but also the service fairness could be improved. This paper extends the model of the SCRP with the Flexible Service policies (SCRP-FS) in Feng et al. (2020) to the case with multiple sub-time windows, and it focuses on the balance between reducing the expected number of relocations and the concern of service fairness. This problem is termed as the *SCRP with Multiple sub-time windows-based Flexible Service policy* or *SCRP-MFS*.

Another feature of the SCRP-FS model is that it takes the customers' preference on different sub-time windows into consideration to characterise the randomness of truck arrivals. However, the influences of the customer preference information on the results of the SCRP-FS have not been sufficiently examined. More importantly, in reality, the information about customer preference can be uncertain or may not be available to terminal operators. Therefore, the question remains as to whether it is worthwhile to commit efforts to gather and utilise the customer preference information. In this paper, we will consider various scenarios and evaluate the impacts and values of the customer preference information on the number of relocations.

In summary, this paper aims to achieve the following objectives: (i) to generalize the SCRP-FS into the case with multiple sub-time windows and incorporate the measure for service fairness (i.e., the maximum turnaround time among trucks); (ii) to analyse the impacts of the number of sub-time windows on the system overall efficiency (represented by the total number of relocations and the average truck turnaround time) and the quality of service to individual trucks (represented by the maximum value and the coefficient of variation of truck turnaround times); (iii) to investigate how customer preference on different sub-time windows impacts the number of relocations (iv) and to evaluate the value of customer preference information in reducing the number of relocations in the SCRP-MFS.

This research sheds new light on some important issues in the SCRP that applies flexible service policies by: (i) demonstrating how the overall efficiency of container retrieval and the quality of service to individual trucks are traded off under different levels of service flexibilities, which can help the terminal operators to determine the appropriate number of sub-time windows to implement flexible service policy;

(ii) having a more comprehensive understanding of whether the problem is sensitive to customer preference, to ensure the robustness of implementing flexible service policy; (iii) and gaining an insight into the value of customer preference information, which could inform decision making on whether to commit resources to gather such information in order to improve the container retrieval performance.

The remainder of the paper is organized as follows. Section 3.2 reviews the relevant literature regarding the (S)CRP and discusses the service fairness and value of information. Section 3.3 formulates the SCRP-MFS by stochastic dynamic programming. A heuristic algorithm is presented in Section 3.4 to solve the problem. The results of computational experiments are provided in Section 3.5. In Section 3.6, we summarize the findings, provide managerial insights and envisage future research directions.

3.2 Literature review

We organize the literature into three parts. Section 3.2.1 reviews the studies on Container Relocation Problem (CRP) and stochastic CRP. Section 3.2.2 focuses on the service policies and service fairness associated with CRP and SCRP. Section 3.2.3 discusses the value of information to container handling operations in the yard.

3.2.1 Container relocation problems

The CRP may be formally defined as follows: given a set of containers stored in a yard bay and the retrieval priorities among the containers, the objective is to empty the bay with the minimum number of relocations (Ku and Arthanari, 2016b). The goal of the CRP is usually to determine the stacking positions for the relocated containers. In the standard CRP, the problem setting is static and restricted in the sense that it assumes that i) no containers are added to the storage stacks during the container retrieval process and ii) a container is relocatable only when it blocks a target container. The static and restricted CRP has been the focus of attention of many researchers. In this paper, we also confine the problem under consideration to this version. With the relaxations of each of the above assumptions, there are two variants of the CRP, i.e., the dynamic CRP and the unrestricted CRP. The interested readers are referred to the works by Wan et al. (2009) and Hakan Akyüz and Lee (2014) for the dynamic CRP and the works by Jin et al. (2015) and Tanaka and Mizuno (2018) for the unrestricted CRP to get in-depth knowledge.

More complicated variants of the CRP come in the form of different settings of the container retrieval priorities. Regarding the retrieval priority, two types of assumptions have been traditionally made (Jang et al., 2013): 1) unique or group retrieval priorities; and 2) deterministic or uncertain retrieval priorities. Table 3. 1 classifies the literature on the CRP that is most relevant to this paper by these two types of assumptions.

Under the first type of assumptions, it is often assumed that each container has a unique retrieval priority and is regarded as a single group by itself. Otherwise, it may be assumed that grouped containers have the same priorities. In the latter case, the retrieval sequence for the containers with the same retrieval priority is to be determined. Practical examples and academic research of the latter case are as follows. For export containers, their loading precedence is specified by the precedence relationship among clusters of empty slots in a vessel. A cluster of slots is characterised by the attributes of containers, for example, container weight class, destination port and type. When a container with specified attributes is requested for loading into a slot, any container with the same set of attributes can be loaded into that slot (Kang et al., 2006). In the work of Kim and Hong (2006), the retrieval precedence among groups of containers is given, and the authors solve the CRP to minimise the total number of relocations by optimising both the retrieval sequence

and the relocation positions. As another example, for import containers, when multiple containers are destined for the same consignee or are to be transported by the trucks from the same shipping company, their pickup sequence is not important and thus can be in any order (Jang et al., 2013). de Melo da Silva et al. (2018) propose the Block Retrieval Problem (BRP) and the Bi-objective Block Retrieval Problem (2BRTP), assuming that the containers in the same group are to be retrieved by the same customer. The BRP aims to minimise the number of relocations for the initial target group by optimising the retrieval sequence and the relocation positions. The 2BRTP includes a secondary objective to minimise the expected number of relocations for the next group.

The second type of assumptions regards whether the retrieval priorities of different groups of containers are deterministic or uncertain. In reality, the retrieval order of import containers usually depends on the truck arrival order, hence, it is subject to high uncertainties due to the unpredictability of truck arrival. While most of the studies on the CRP assume containers are sequenced with a certain prescribed order, several researchers have addressed the problem considering uncertain retrieval priorities. The uncertain CRPs can be further categorised into two sub-categories (Feng et al., 2020): the online setting and the probabilistic setting. In the online setting, the container retrieval sequence is revealed over time, and the knowledge of future retrievals is limited to a look-ahead horizon. The goal is to design efficient online heuristics to determine the relocation positions for the blocking containers using information updated real-time (e.g., Zehendner et al., 2017; Zhao and Goodchild, 2010). In the probabilistic setting, the containers' retrieval priorities are modelled by a probability distribution and the research aim is to minimise the expectation of associated performance measures, such as the expected total number of relocations (Tong et al., 2015; Ku and Arthanari, 2016a; Galle et al., 2018; Feng et al., 2020) and the weighted sum of the expected number of relocations and total retrieval delays (Borjian et al., 2013). For a more detailed review of the uncertain CRPs, we refer the readers to Feng et al. (2020). Here, we would like to note that the uncertain CRP in which groups of containers are ordered by the appointed time windows and the trucks arrive at the appointed time windows randomly is referred to as the Stochastic Container Relocation Problem (SCRP) in the literature (Galle et al., 2018; Feng et al., 2020). The problem considered in this paper belongs to the SCRP.

Methodologically, the CRPs have been usually modelled using (mixed) integer programming models (e.g., Wan et al., 2009; Petering and Hussein, 2013; Tang et al., 2015; Zehendner et al., 2015) or (stochastic) dynamic programming models (e.g., Kim and Hong, 2006; Ku and Arthanari, 2016a; Galle et al., 2018; Feng et al., 2020). For exact solution algorithms, search-based algorithms are mainly used to seek optimal solutions: (iterative deepening) A* algorithms (Zhu et al., 2012; Quispe et al., 2018), branch and bound (Kim and Hong, 2006; Expósito-Izquierdo et al., 2015; Tanaka and Takii, 2015), branch and price (Zehendner and Feillet, 2014), and branch and cut (Bacci et al., 2020). As the CRP has proven to be NP-hard (Caserta et al., 2012), only small-scale instances are able to be solved exactly within reasonable times. As a result, researchers often turn to heuristic approaches to overcome the computational complexities, such as index-based heuristics (Caserta et al., 2012; Hakan Akyüz and Lee, 2014), beam search (Bacci et al., 2019; Ting and Wu, 2017), metaheuristics (Maglić et al., 2020), and other greedy heuristics (Jovanovic and Voß, 2014; Jin et al., 2015) (c.f. Caserta et al. (2020) and the references therein). Recently, a new trend of the solution approach has been developed by using machine learning techniques (Zhang et al., 2020) and reinforcement learning (Jiang et al., 2021).

Finally, it is worth mentioning that there are several related problem types regarding container relocation, such as storage space allocation that deals with the initial storage of containers into the yard (e.g., Zhou et al., 2020) and container marshalling operation that re-arranges the container stacking positions before

containers leave the yard (e.g., Parreño-Torres et al., 2020). In addition, in seaport rail terminals, prestaging operation is performed to pre-move the containers from the storage yard to local storage areas near the train before the train arrival in order to meet the departure deadline of trains (Xie and Song, 2018).

3.2.2 Service policies and service fairness

A service policy for CRP is to determine how the arriving trucks are served at the container yard. The service policy plays an important role in improving the efficiency of the import container retrieval process. The most commonly used service policy is the first-come-first-serve (FCFS) rule for trucks arriving at the yard. However, the FCFS policy does not lead to the most efficient container retrievals. A few studies have proposed flexible service policies or out-of-order retrievals to CRP, which has been proved to be more effective than FCFS policy. When a group of containers have the same retrieval priority, i.e. they are exchangeable in terms of retrieval sequence during the retrieval process (Kim and Hong 2006; de Melo da Silva et al. 2018), the out-of-order service policy is implicitly accepted by customers, and no concern of service fairness is raised. On the other hand, when containers are not exchangeable, a flexible service policy can cause extra waiting times for some customers and may raise the issue of service unfairness. The service unfairness herein refers to the difference in the waiting times among trucks or customers when out-of-order service is implemented. For example, under the truck appointment system, a truck books a time window and the containers to be retrieved, therefore, each container is bound with a specific truck. Namely, even though multiple containers are booked in the same time window, these containers are not exchangeable as they are requested by different trucks. In this case, retrieving the containers in an order different from the truck arrival order will influence the waiting time of individual customers (i.e., trucks). It may be perceived to be unfair if a truck experiences out-of-order retrievals and gets served later than expected. The problem under consideration in this paper falls into this category. Studies in this regard try to maintain fairness among trucks by controlling the retrieval flexibilities within a certain range. A couple of papers (e.g., Zhao and Goodchild, 2010; Zeng et al., 2019) restrict out-of-order retrievals within a group of containers that are booked in the same time window, but they focus only on minimising the number of relocations. In the study of Zhao and Goodchild (2010), the impact of dictating the pickup sequence on the waiting times of external trucks is not analysed. In contrast, although Zeng et al. (2019) also focus on minimising the number of relocations, they evaluate the average truck waiting times by simulation and conclude that when the number of containers booked in each time window exceeds a certain value, adjusting the pickup sequence will cause excessive delay for external trucks. This result may be caused by the assumption that the trucks receive the retrieval service immediately after they arrive at the terminal by ignoring the phenomenon of queuing. In busy terminals, the queue of the trucks may create a beneficial context in which serving a later arrival truck before an earlier arrival truck may not add a lot of additional waiting times for the earlier truck since all trucks have been ready in the queue before the retrieval service starts. Recently, Azab and Morita (2021) introduce a new problem — the Block Relocation Problem with Appointment Scheduling (BRPAS) — aiming at improving import container relocation operations by coordinating with appointment scheduling. It assumes that the terminal operator can reschedule the container pickup times requested in the appointment system and determine the final appointments and the pickup order for each container. An appointment shift allowance is introduced to control the gap between the requested pickup times and the allocated appointment times. The FCFS policy is applied to the containers in the same subgroup to reduce truck waiting times.

When dealing with the CRP that applies flexible service policies, very few studies have incorporated customer-centric performance metrics into the objective functions. Borjian et al. (2013) set a maximum

service delay for each container and construct the objective function to minimise the weighted sum of the expected number of relocations and total retrieval delays. In their study, the uncertain information about the arrival times of all trucks is revealed at once, which does not represent the dynamism of uncertainties in real terminal operations. Borjian et al. (2015) control the level of flexibility by setting a maximum number of out-of-order retrievals before each truck. They conclude that the flexible retrieval planning can decrease the total number of relocations and average waiting time of external trucks while maintaining service equity for each truck in the long term. However, they study the CRP in a determinism setting and assume that the exact arrival time of the external trucks is known, which is unrealistic. Feng et al. (2020) investigate the SCRP that applies the flexible service policy (termed as the SCRP-FS), where the primary objective is to minimise the expected total number of relocations and the secondary objective is to minimise the total truck waiting times of each batch sequentially. A batch refers to the trucks that have booked in the same time window. A batch of trucks arrive at its booked time window randomly and their exact arrival order is revealed batch by batch. The flexible service policy allows the trucks that arrive at the same sub-time window to be served out-of-order. However, the service fairness is neither optimised nor balanced against the number of relocations in Feng et al. (2020).

3.2.3 Value of information

Relevant information could reduce the degree of uncertainty and improve yard operation performance (Zuidwijk and Veenstra, 2015). Utilising information on truck arrival times collected from the truck appointment system, a few studies examine ways of reducing the number of relocations during the container retrieval process. Zhao and Goodchild (2010) develop two heuristics to address the CRP with time windows where containers are grouped and ordered by the trucks' arrival time windows and new information on the exact truck arrival order is updated one truck at a time; this model is referred to as the online model by Galle et al. (2018). Their findings show that significant reductions in relocations can be achieved from just knowing in which groups a truck will arrive. Ku and Arthanari (2016a) also address the CRP in the context of the online model, but they formulate the problem as a stochastic dynamic programming model and solve it optimally by a decision tree scheme. Different from using the online model, Galle et al. (2018) consider the batch model in which the exact truck arrival order is updated a batch (group) at a time and formulate it as a multi-stage optimisation problem. They prove the value of taking into account the within batch information in reducing the expected number of relocations both theoretically and numerically.

Some studies utilise the arrival information of trucks or vessels to improve container stacking efficiency and reduce the number of relocations. van Asperen et al. (2013) evaluate how information of truck announcement time impacts the performance of online container stacking operations; their findings show that an average announcement time of 0.5-24 hours can significantly improve the stacking efficiency. Gharehgozli and Zaerpour (2018) propose a shared stacking policy that utilises the information on the arrival time windows of barges to determine the storage locations of outbound barge containers; compared to the practical staking policy, this policy proves to reduce up to 30% of the total retrieval time. In addition, the information related to truck arrivals has also been utilised in container (p)re-marshalling operations to rearrange the container stacking positions in order to improve future container retrieval efficiency. Covic (2017) introduce an online rule-based solution method for container re-marshalling by taking the truck arrival information into consideration. The results show that imprecise arrival information of trucks, not deviating above a certain threshold, can significantly reduce truck waiting time. Kim and Yi (2021) develop heuristic algorithms to locate and pre-marshal import containers using various information sources related

to truck arrivals (dwell time distribution of containers, truck dispatching notice, truck appointment, and real-time position of trucks). The results show that when all these sources of information are utilised, the truck system time and the number of relocations during a container retrieval can be reduced by 47% and 98%, respectively.

Different from the aforementioned studies that mainly use information about truck arrivals, this paper is to assess the value of customer preference information in reducing the number of relocations in the SCRP that applies flexible service policies. The customer preference information is measured as the probabilities of a truck arriving in each sub-time window within its booked time window. The motivation is based on the fact that the specific probability distribution of the arrival order of trucks in an appointment time window is hard to predict. The existing studies on the SCRP (Ku and Arthanari, 2016a; Galle et al., 2018) make a simplified assumption that any possible arrival order of the trucks booked in an appointment time window has the same probability. Such an assumption may not capture the customer (i.e., truck) behaviours adequately as the customers may have their preferred arrival segments within the booked time window. There have been studies about shifting trucks' arrival times from their preferred arrival times to reduce truck congestion in port areas (e.g., Chen et al., 2013a, b; Phan and Kim, 2015) and to reduce relocations (Azab and Morita, 2021). In a recent study on the SCRP-FS, Feng et al. (2020) consider unequal probabilities of the truck arrival order that vary with the customer's preference for each sub-time window of the booked time window. Such customer preference information can be utilised to reduce the number of relocations but the extent of the improvement has not been quantified.

3.2.4 Research gap

Table 3. 1 summarizes the comparisons of this paper with the most closely related literature from four key aspects. Only two papers discuss the issue of service fairness (equity) among trucks in the CRP. Borjian et al. (2015) consider the service fairness in a deterministic context and evaluate it by the number of out-of-order retrievals performed before a particular truck is served. Feng et al. (2020) studies the CRP in a probabilistic setting and evaluate the impacts of out-of-order retrievals on individual trucks in the SCRP-FS by the maximum truck turnaround time, but their model is limited to a special case in which each appointment time window is divided into two sub-time windows. More importantly, neither of the studies optimise the service fairness, nor has it explicitly addressed the trade-off decision between the expected number of relocations and the service fairness. This paper attempts to fill these gaps. In this paper, we generalize the SCRP-FS to the case with multiple sub-time windows, termed as SCRP-MFS, and incorporate the service fairness into objective functions. Through such generalization, we can control the level of service flexibility more accurately, making the mathematical model more relevant to reality. Applying multiple sub-time windows is beneficial to reduce the maximum truck turnaround time and the coefficient of variation of the turnaround times among different trucks (see our experiments' result in Section 3.5.1), which can both reflect the service fairness.

In addition, the smaller sub-time window can capture the customer preference information more accurately. Feng et al. (2020) consider the situation of having two sub-time windows under which the probability of a truck arriving at a sub-time window is characterised by customer preference. But trucks arriving at multiple sub-time windows remains an unresearched question in the CRP discipline. Besides, the value added by the customer information in reducing the number of relocations has not been assessed, and it remains unclear as to whether it is worthwhile to commit resources to gather and utilise such information. This research also aims to fill these two gaps, by proposing a general probabilistic model of truck arrivals at multiple sub-time windows, and evaluating the impacts and the value of customer preference information

on the number of relocations.

Table 3. 1 Classification of the most relevant literature on the CRP.

Literature	Unique or group priority	Certainty of priorities	Service policy	Performance metrics
Borjian et al. (2013)	Group	Deterministic; Uncertain	Flexible	WS of ENR and RD
Borjian et al. (2015)	Unique	Deterministic	Flexible	WS of NR and RD; ORT*
de Melo da Silva et al. (2018)	Group	Uncertain	Flexible	NR; ENR
Galle et al. (2018)	Unique	Uncertain	FCFS	ENR
Kim and Hong (2006)	Unique; Group	Deterministic	FCFS; Flexible	NR
Ku and Arthanari (2016a)	Unique	Uncertain	FCFS	ENR
Zeng et al. (2019)	Unique	Deterministic	Flexible	NR
Zhao and Goodchild (2010)	Unique	Deterministic; Uncertain	FCFS; Flexible	NR; CHTD*
Feng et al. (2020)	Unique	Uncertain	Flexible with 2-sub-windows	ENR; total TWT; maximum TTT*
This paper	Unique	Uncertain	Flexible with multi-sub-windows	ENR; maximum TTT; average and CoV of TTT*

NR: number of relocations; ENR: expected number of relocations; TWT: truck waiting times; TTT: truck turnaround times: RD: retrieval delays; CHTD: crane horizontal travel distance; WS: weighted sum; CoV: coefficient of variation; ORT: out-of-order retrievals performed before a truck is served.

Note: performance metrics without "*" represent optimised objectives, and those with "*" represent evaluated metrics.

3.3 The SCRP-MFS

In this section, we first describe the SCRP-MFS and then introduce the probabilistic model of truck arrivals. Next, we present the problem formulation. Last, we analyse the model and develop the handling approach.

3.3.1 Problem description

The definitions and notations used in the SCRP-MFS are given in Section 3.3.1.1, followed by the problem assumptions presented in Section 3.3.1.2. A list of the essential modelling notations is provided in Appendix A.1.

3.3.1.1 Definitions and notations

A bay consists of *S* stacks, *T* tiers, and *C* containers. In order to avoid infeasible relocations, the storage capacity of the bay is restricted to be $(S-1)T+1$ containers. Each container is booked to a time window and the corresponding truck (i.e., one container corresponds to one truck) will arrive at the terminal within the appointed time window. Furthermore, containers and trucks are grouped into a set of batches and sub-batches:

(1) A ***batch*** of containers/trucks is defined as a set of containers/trucks booked to the same time window. Let B_k denote the set of containers in batch *k* and C_k denote the number of containers in batch *k*, $k \in \{1,...,K\}$. By definition $\sum_{k=1}^{K} C_k = C$.

(2) The arrival precedence relationship among batches of trucks is known, but the exact arrival order of trucks within each batch is uncertain, and it is revealed as the retrieval proceeds.

(3) Each appointment time window is divided into W ($W \in \mathbb{Z}^+$) sub-time windows with identical time

lengths. A ***sub-batch*** of trucks is a set of trucks that have arrived in the same sub-time window $w \in \{1,...,W\}$. Accordingly, a ***sub-batch*** of containers is a set of containers whose trucks have arrived in the same sub-time window $w \in \{1,...,W\}$.

(4) Flexible service policy is applied to the containers/trucks in the same sub-batch, that is, the containers/trucks in the same sub-batch can be retrieved/served out-of-order.

Each container has three attributes: priority label, unique ID, and customer preference:

(1) The ***priority label***, denoted by l_i, $i \in \{1,...,C\}$, is used to trace the retrieval priorities among containers. The priority label is updated during the container retrieval process. Initially, containers in each batch *k* are labelled by a *batch priority*, denoted by L_k, which represents the arrival precedence among batches of trucks (see Fig. 3. 1(a)). We let $L_k = 1 + \sum_{j=1}^{k-1} C_j$, and thus given L_k, there is a unique $k \in \{1,...,K\}$. Then, once the arrival order of trucks in batch *k* is revealed, the priority labels of the containers in this batch are updated to the *sub-batch priority* that represents the arrival precedence among sub-batches of trucks (see Fig. 3. 1(c)). If the truck for a container in batch *k* is revealed to arrive in sub-time window *w*, its label changes to $L_k + \sum_{w'=1}^{w-1} n_{kw'}$, where $n_{kw'}$ denote the number of trucks that have arrived in sub-time window w'. Finally, once the retrieval sequence of a container in batch *k* is determined, its label is updated to the determined retrieval sequence that is within $[L_k, L_k + C_k - 1]$.

(2) The ***unique ID***, denoted by u_i, $i \in \{1,...,C\}$, is used to differentiate individual containers/trucks (see Fig. 3. 1(b)).

(3) The ***customer preference***, denoted by $p_{i,w}^W$, $i \in \{1,...,C\}$, $w \in \{1,...,W\}$, represents the probability of truck u_i arriving at sub-time window *w* of its appointed time window (see Fig. 3. 1(d)). Each appointment time window is divided into *W* sub-time windows. The values of $p_{i,w}^W$ can be derived from the truck appointment system that requires trucks to provide their preferences when booking a window; alternatively, the values of $p_{i,w}^W$ can be estimated from historical data as the proportion of truck u_i arriving at sub-time window *w*.

3.3.1.2 Assumptions

The following assumptions are made in the SCRP-MFS.

A1: Relocations are performed only within a single bay.

A2: A container is relocatable only when it is blocking the target container.

A3: No new containers arrive at the bay during the container retrieval process.

A4: (information revealing time) For each batch, the full arrival order of the trucks in this batch is revealed at once after all containers in its prior batch have been retrieved.

A5: (service beginning time) The retrieval service of a batch begins at the end of the appointed time window associated with the batch.

A6: (probabilities of truck arrivals) (1) Within each batch, the probability of a truck arriving at a sub-time window depends on customer preference, and (2) within each sub-batch, the arrival order of the trucks follows an uniform distribution.

A1 to A3 are generic to the standard CRP. A4 and A5 ensure that when the retrieval service of a batch begins, the arrival order of the trucks in this batch has been revealed. A6 is about the probabilistic model of truck arrivals, which will be introduced in detail in section 3.3.2.

3.3.2 General probabilistic model of truck arrivals

The arrival order of trucks in a batch is stochastic. We characterise the truck arrival order in the same way as Feng et al. (2020), who model it by customer preference. The customer preference refers to a truck's arrival probabilities for each sub-time window of the booked time window. In this paper, we propose a general probabilistic model of truck arrivals, which extends the two sub-time windows model in Feng et al. (2020) to the case of multiple sub-time windows.

Each appointment time window is divided into *W* sub-time windows with identical time lengths. Let ζ_k^W, $k \in \{1,...,K\}$, refer to a random scenario of which trucks in batch *k* arrive at each of the *W* sub-time windows of the appointed time window, and let $p(\zeta_k^W)$ refer to its probability. $\zeta_k^W = \left\{\xi_{k,w}^W \mid w \in \{1,...,W\}\right\}$, where $\xi_{k,w}^W$ is a random set of the trucks in batch *k* that arrive in sub-time window *w*. $\xi_{k,w}^W$, $w \in \{1,...,W\}$, are mutually exclusive and collectively exhaustive, that is, $\bigcup_{w\in\{1,...,W\}} \xi_{k,w}^W = B_k$ and $\bigcap_{w\in\{1,...,W\}} \xi_{k,w}^W = \varnothing$. The random variables in $\xi_{k,w}^W$ take values in B_k, and $\left\{u_i \in \xi_{k,w}^W\right\}$ represents the event that truck u_i arrives in the sub-time window *w*. By definition, $p_{i,w}^W = P(u_i \in \xi_{k,w}^W)$, where $u_i \in B_k$ and $w \in \{1,...,W\}$. Let $\zeta^W = \bigcup_{k\in\{1,...,K\}} \zeta_k^W$ denote the truck arrival scenario for all the batches. There are a total of W^{C_k} possible scenarios of ζ_k^W for batch *k* and a total of $\prod_{k=1}^{K} W^{C_k}$ scenarios for ζ^W.

The probabilities of ζ_k^W can be calculated by $p_{i,w}^W$. An example is given in Fig. 3. 1 to illustrate the containers' attributes and explain the probabilistic model. In this example, *W* = 3. Fig. 3. 1(a), (b) and (d) constitute the initial bay configuration. Fig. 3. 1(c) reveals the sub-batch priority of the first batch (in bold). Fig. 3. 1(d) presents the customer preference of each container/truck for each sub-time window, i.e., $p_{i,w}^W$. Now we explain the general probabilistic model of truck arrivals by taking the example of ζ_1^3. There are totally nine scenarios of ζ_1^3, which are respectively $\left\{\xi_{1,1}^3 = \{u_4,u_7\}, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \varnothing\right\}$, $\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \{u_4,u_7\}, \xi_{1,3}^3 = \varnothing\right\}$, $\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \{u_4,u_7\}\right\}$, $\left\{\xi_{1,1}^3 = \{u_4\}, \xi_{1,2}^3 = \{u_7\}, \xi_{1,3}^3 = \varnothing\right\}$, $\left\{\xi_{1,1}^3 = \{u_4\}, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \{u_7\}\right\}$, $\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \{u_4\}, \xi_{1,3}^3 = \{u_7\}\right\}$, $\left\{\xi_{1,1}^3 = \{u_7\}, \xi_{1,2}^3 = \{u_4\}, \xi_{1,3}^3 = \varnothing\right\}$, $\left\{\xi_{1,1}^3 = \{u_7\}, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \{u_4\}\right\}$, $\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \{u_7\}, \xi_{1,3}^3 = \{u_4\}\right\}$. The probabilities of these scenarios are computed as:

$$p\left(\left\{\xi_{1,1}^3 = \{u_4,u_7\}, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \varnothing\right\}\right) = 0.8\times0.4 = 0.32 \quad ;$$

$$p\left(\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \{u_4,u_7\}, \xi_{1,3}^3 = \varnothing\right\}\right) = 0.2\times0.3 = 0.06 \quad ; \quad p\left(\left\{\xi_{1,1}^3 = \varnothing, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \{u_4,u_7\}\right\}\right) = 0\times0.3 = 0 \quad ;$$

$$p\left(\left\{\xi_{1,1}^3 = \{u_4\}, \xi_{1,2}^3 = \{u_7\}, \xi_{1,3}^3 = \varnothing\right\}\right) = 0.8\times0.3 = 0.24 \quad ; \quad p\left(\left\{\xi_{1,1}^3 = \{u_4\}, \xi_{1,2}^3 = \varnothing, \xi_{1,3}^3 = \{u_7\}\right\}\right) = 0.8\times0.3 = 0.24 \quad ;$$

$p\left(\left\{\xi_{1,1}^3=\varnothing,\xi_{1,2}^3=\{u_4\},\xi_{1,3}^3=\{u_7\}\right\}\right)=0.2\times0.3=0.06$ ； $p\left(\left\{\xi_{1,1}^3=\{u_7\},\xi_{1,2}^3=\{u_4\},\xi_{1,3}^3=\varnothing\right\}\right)=0.4\times0.2=0.08$ ；

$p\left(\left\{\xi_{1,1}^3=\{u_7\},\xi_{1,2}^3=\varnothing,\xi_{1,3}^3=\{u_4\}\right\}\right)=0.4\times0=0$ ； $p\left(\left\{\xi_{1,1}^3=\varnothing,\xi_{1,2}^3=\{u_7\},\xi_{1,3}^3=\{u_4\}\right\}\right)=0.3\times0=0$. According to the real truck arrival order, ζ_1^3 will be revealed to be one of these nine scenarios. For example, if ζ_1^3 is revealed to be $\left\{\xi_{1,1}^3=\{u_4\},\xi_{1,2}^3=\{u_7\},\xi_{1,3}^3=\varnothing\right\}$, $\left\{\xi_{1,1}^3=\{u_4\},\xi_{1,2}^3=\varnothing,\xi_{1,3}^3=\{u_7\}\right\}$, or $\left\{\xi_{1,1}^3=\varnothing,\xi_{1,2}^3=\{u_4\},\xi_{1,3}^3=\{u_7\}\right\}$, the sub-batch priority for the first batch will be updated to be that shown in Fig. 3. 1(c), which indicates that truck u_4 and u_7 arrive in different sub-time windows and u_4 arrives before u_7.

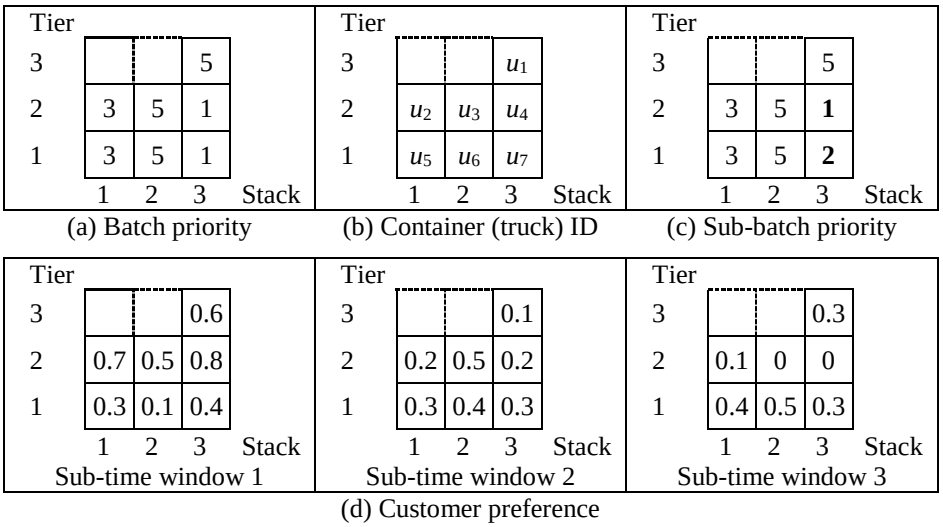


Fig. 3. 1 An illustration of containers' attributes

3.3.3 Problem formulation

The SCRP-MFS proposed in this research is a multi-stage sequential decision-making problem with dynamic information revealing. The truck arrival information is revealed batch by batch (i.e., stage). At each stage, decisions are made based on the revealed information. Mathematically, it can be formulated as a stochastic dynamic programming model (SDP) similar to the Sooo model in Feng et al. (2020). However, the two models differ in several ways. While both models consider the randomness of truck arrivals within an appointed time window, SCRP-MFS models the arrival in multiple sub-time windows rather than in two sub-time windows as considered in the Sooo model. Correspondingly, the decision variable in SCRP-MFS is the retrieval sequence for the containers in multiple sub-batches, as opposed to the retrieval sequence in two sub-batches in the Sooo model. Last, the objective functions are different. This study introduces a secondary objective function measuring service fairness, which has been rarely considered in the CRP literature. In the following, we first introduce the objective functions of the SCRP-MFS. Then, we present the formulation regarding each objective.

3.3.3.1 Objective functions

The main subjects involved in the SCRP-MFS are terminal operators and trucks. We formulate the SCRP-MFS as a multi-objective optimisation problem aiming to improve the relocation efficiency from the terminal operators' perspective and meanwhile to mitigate the unfairness among trucks from the perspective of the worst-off trucks. We adopt two lexicographically ordered objectives. The primary objective is to minimise the expected total number of relocations, which reflects the relocation efficiency.

The secondary objective is to minimise the maximum truck turnaround time, which reflects service fairness and is called min-max fairness in this paper. The min-max index is a well-studied objective function used to measure fairness in transportation systems (e.g., Li et al., 2019; Wu et al., 2021), the healthcare industry (e.g., Qi 2017) and other domains of the operational research (see the references in Yang et al., 2013). It has also been used in berth allocation problems in container terminals to minimise the maximum lateness of any vessel where each vessel has a due departure time (Teye and Bell, 2016). The adoption of lexicographically ordered objectives indicates that we still take the relocation efficiency as our principal aim and the fairness concern is considered under the promise that the number of relocations is minimised. This is because that the number of relocations has a direct effect on the truck turnaround time. The minimisation of the number of relocations is beneficial for reducing the truck turnaround time, but multiple solutions with the same minimal number of relocations can lead to different maximum truck turnaround times. In the following, we introduce the formulations regarding the primary objective and the secondary objective respectively.

3.3.3.2 Primary objective for relocation efficiency

The stage, state, and action of the SDP model are defined as follows.

Stage: the sequence number of the batch to be retrieved. A stage refers to a batch. The example in Fig. 3. 1 is regarded as the first stage since the first batch of containers is to be retrieved.

State: the state of each stage is the state of the bay that consists of the stacking positions of the remaining containers and their attributes. The input state of the *k*th stage is the state of the bay after the (*k*-1)th batch has been retrieved and before ζ_k^W is revealed. For example, Fig. 3. 1(a), (b) and (d) constitute the input state of the first stage.

Action: a feasible action is defined as a sequence of moves to retrieve a batch of containers, which consists of two types of actions: (i) the retrieval/service sequences of the containers/trucks in each sub-batch, called *sequencing*, and (ii) the stacking positions of the relocated containers, called *relocating*.

The notations in the SDP model are defined as follows. These notations are also included in Appendix A.1.

W: the number of sub-time windows in an appointment time window (a decision variable).

K: the total number of batches in the initial bay, which also represents the total number of stages.

k: the stage number, $k \in \{1,...,K\}$, and stage *k* refers to the *k*th batch to be retrieved.

ζ_k^W : the scenario of the sub-batches of stage *k*, $k \in \{1,...,K\}$ (a random variable).

S_k : the input state of stage *k*, $k \in \{1,...,K\}$.

$p(\zeta_k^W)$: The probability of ζ_k^W, which is calculated by the general probabilistic model of truck arrivals introduced in section 3.3.2.

$\mathbf{a}_k(S_k,\zeta_k^W)$: The actions (a decision variable) taken for retrieving the *k*th batches of containers given S_k and ζ_k^W. $\mathbf{a}_k(S_k,\zeta_k^W)=\{\mathbf{a}_k^S(S_k,\zeta_k^W),\mathbf{a}_k^R(S_k,\zeta_k^W)\}$, wherein $\mathbf{a}_k^S(S_k,\zeta_k^W)$ represents the retrieval sequence for the containers in each sub-batch at stage *k* given S_k and ζ_k^W, and $\mathbf{a}_k^R(S_k,\zeta_k^W)$ represents the relocation positions that respect $\mathbf{a}_k^S(S_k,\zeta_k^W)$. For notational convenience, we suppress the dependence on (S_k,ζ_k^W) from $\mathbf{a}_k(S_k,\zeta_k^W)$ and use $\mathbf{a}_k$ instead.

$r_k(\mathbf{a}_k \mid S_k,\zeta_k^W)$: The number of relocations required during action $\mathbf{a}_k$ on the bay of state S_k given ζ_k^W.

$t_k(S_k,\zeta_k^W,\mathbf{a}_k)$: The state transition function that maps S_k, ζ_k^W, and $\mathbf{a}_k$ into the next state S_{k+1}. By $t_k(S_k,\zeta_k^W,\mathbf{a}_k)$, the *k*th batch of containers revealed by ζ_k^W are retrieved from state S_k according to action

$\mathbf{a}_k$, after which S_{k+1} is obtained.

$f_k^W(S_k)$: The expected minimum total number of relocations to retrieve the remaining *K*-*k*+1 batches of containers from state S_k when each appointment time window is divided into *W* sub-time windows.

The primary objective is formulated as a recursive function as follows:

$$f_1^W(S_1) = E\left[\min_{\mathbf{a}_1}\left\{r_1(\mathbf{a}_1 \mid S_1, \zeta_1^W) + f_2^W(S_2)\right\}\right] = \sum_{\zeta_1^W} p(\zeta_1^W) \min_{\mathbf{a}_1}\left[r_1(\mathbf{a}_1 \mid S_1, \zeta_1^W) + f_2^W(S_2)\right],$$

where $S_2 = t_1(S_1, \zeta_1^W, \mathbf{a}_1)$ (3.1)

Generally, the problem for any stage *k* can be formulated as follows:

$$f_k^W(S_k) = \sum_{\zeta_k^W} p(\zeta_k^W) \min_{\mathbf{a}_k}\left[r_k(\mathbf{a}_k \mid S_k, \zeta_k^W) + f_{k+1}^W(S_{k+1})\right],\ k \in \{1,...,K\},$$

where $S_{k+1} = t_k(S_k, \zeta_k^W, \mathbf{a}_k)$, for $k \in \{1,...,K\}$, and $f_{K+1}^W(S_{K+1}) = 0$ (3.2)

The objective (3.1) is to minimise the expected total number of relocations to retrieve all the containers. Equation (3.2) means that at the beginning of stage *k*, first, the truck arrival scenario for this stage, that is, ζ_k^W , is revealed, and then the optimal actions $\mathbf{a}_k(S_k, \zeta_k^W)$ to retrieve the containers at this stage are sought by considering all the scenarios of future stages.

3.3.3.3 Secondary objective for service fairness

The secondary objective is to minimise the maximum turnaround time among all of the trucks. The *turnaround time* of truck *i* is defined as the elapse of time between its arrival time $\tilde{a}_i$ and its departure time from the yard $\tilde{d}_i$, that is, the retrieval service completion time. Let $g_{k,i}^{\zeta_k^W} = (\tilde{d}_i - \tilde{a}_i)$ denote the turnaround time of truck *i* in batch *k* ($i \in B_k$) under ζ_k^W ($\zeta_k^W \subset \zeta^W$). Then, the secondary objective function can be expressed as

$$\text{Min} \sum_{\zeta^W} p(\zeta^W) \max_{k \in \{1,...,K\}, i \in B_k} g_{k,i}^{\zeta_k^W} \quad (3.3)$$

Note that the secondary objective in (3.3) is optimised under the promise that the primary objective in (3.1) has achieved optimality, which means that the two objectives are optimised sequentially. Given the decisions $\mathbf{a}_k(S_k, \zeta_k^W)$ that are optimal regarding the primary objective, we now derive the explicit expression of $g_{k,i}^{\zeta_k^W}$. The following notations are introduced to extract the relevant information implied in the decision variables $\mathbf{a}_k(S_k, \zeta_k^W) = \{\mathbf{a}_k^S(S_k, \zeta_k^W), \mathbf{a}_k^R(S_k, \zeta_k^W)\}$.

$o_i^{\zeta_k^W}$: the service order of truck *i*, $i \in B_k$, under ζ_k^W . $o_i^{\zeta_k^W}$ is implied in the service sequence decision $\mathbf{a}_k^S(S_k, \zeta_k^W)$.

$r_i^{\zeta_k^W}$: the number of relocations needed when serving truck *i*, $i \in B_k$, under ζ_k^W . $r_i^{\zeta_k^W}$ is implied in the relocation decision $\mathbf{a}_k^R(S_k, \zeta_k^W)$.

t^{ret} : the handling time per retrieval move.

t^{rel} : the handling time per relocation move.

Let e_k denote the end of the appointed time window of batch *k*. Let s_k denote the service starting time of batch *k* and c_k denote the completion time of retrieving the last container in batch *k*. Given the decisions $\mathbf{a}_k(S_k, \zeta_k^W)$ of batch *k*, c_k and s_k can be obtained. By A5, all the trucks in a batch have already waited at the yard stack when the service of this batch begins, and thus there is no idle time between the services of any two trucks in the batch. Therefore, c_k is calculated by

$$c_k = s_k + \sum_{i \in B_k} (t^{rel} \cdot r_i^{\zeta_k^W} + t^{ret}) \quad (3.4)$$

Given e_k and c_{k-1}, according to A5, s_k is calculated by

$$s_k = \max\{e_k, c_{k-1}\}, \quad k \in \{2,...,K\}; \quad s_1 = e_1. \tag{3.5}$$

Equation (3.5) means that, if the service completion time of batch k-1 is later than the end of the appointed time window of batch k, the service starting time of batch k is c_{k-1}. Otherwise, the service starting time of batch k coincides with the end of its appointed time window, that is, e_k.

Given the above expressions, $g_{k,i}^{\zeta_k^W}$ is calculated by

$$g_{k,i}^{\zeta_k^W} = \tilde{d}_i - \tilde{a}_i = \left(s_k + \sum_{j \in B_k, o_j^{\zeta_k^W} < o_i^{\zeta_k^W}} (t^{rel} \cdot r_j^{\zeta_k^W} + t^{ret}) + t^{rel} \cdot r_i^{\zeta_k^W} + t^{ret} \right) - \tilde{a}_i, \tag{3.6}$$

3.3.4 Model analysis and handling approach

Let $\gamma^{(1)}$ and $\gamma^{(2)}$ denote the primary objective and the secondary objective, respectively. Then, the SCRP-MFS can be formulated as follows:

$$\min \gamma^{(1)} = f_1^W(S_1), \quad f_1^W(S_1) \text{ is defined in Eq. (3.1)}$$

$$\min \gamma^{(2)} = \sum_{\zeta^W} p(\zeta^W) \max_{k \in \{1,...,K\}, i \in B_k} g_{k,i}^{\zeta_k^W}, \quad g_{k,i}^{\zeta_k^W} \text{ is defined in Eqs. (3.4)-(3.6)} \tag{3.7}$$

There are two levels of decisions in the model. At the higher level, the number of sub-time windows W directly impacts the level of flexibility in optimising the container retrieval sequence and thus the objective values. At the lower level, the container retrieval sequence $\mathbf{a}_k^S(S_k, \zeta_k^W)$ and the container relocation positions $\mathbf{a}_k^R(S_k, \zeta_k^W)$ are two-fold decision variables. We handle the model by a hierarchical iterative approach, under the framework presented in Fig. 3. 2. At the outer hierarchy, we determine the number of sub-time windows, while at the inner hierarchy, we make decisions on the container retrieval sequence and the container relocation positions. The two hierarchies are incorporated in an iterative process. At each iteration, we update the value of W, that is, the number of sub-time windows. Accordingly, we solve the model where W is treated as a parameter. To solve this model, the focus is to solve the SDP with the primary objective. Then, among the multiple solutions that optimise the primary objective, the one with the minimal secondary objective value can be selected as the optimal solution for the SCRP-MFS with a specific W. The optimal solutions of the SDP model can be obtained by optimising the recursive equation (2) backward from stage K to stage 1. However, solving the SDP model is very time-consuming for practical scale problems due to the curse of dimensionality (Feng et al., 2020). The case with multiple sub-time windows can be more time consuming, as it creates an increased number of possible sub-batches and corresponding scenarios. To accomplish extensive experiments in a reasonable time, we use a heuristic algorithm to solve the model. When the model is solved, we update the retrieval sequence and the relocation positions and evaluate the relevant performance measures by simulation.

The idea behind the scheme of updating W is to increase the value of W if the maximum truck turnaround time is smaller than that in the previous iteration and to terminate the iteration process either if the maximum truck turnaround time does not improve for Q_1 consecutive iterations or if it reaches the iteration

limit Q_2. In doing so, the decisions of the SCRP-MFS are forced to lead to a greater number of relocations and a shorter maximum truck turnaround time as the iteration process continues. The rationale is explained here. There is a trade-off between relocation efficiency and service fairness when optimising the SCRP-MFS. There is no ultimate solution with the lowest number of relocations and the smallest maximum truck turnaround time. If W is set to be a small value, the solution will turn out to have a smaller number of relocations but a longer maximum truck turnaround time. On the contrary, if W is set to be a higher value, the solution will show the opposite feature. However, A larger value of W does not necessarily guarantee a shorter maximum truck turnaround time. This is because the turnaround time of a truck is also influenced by the number of relocations that are needed to retrieve the target container. Under a larger W, since the flexibility in optimising the container retrieval sequence is very small, a greater number of relocations may be needed to retrieve a container, and thus it may cause longer turnaround times for some trucks. When such a point appears, there is no much need to continue increasing the value of W since the benefit for reducing the maximum truck turnaround time may be tiny. Initially, W is set to be a small value W_0 (e.g., 2) to allow a great extent of flexibility for optimising the container retrieval sequence. Then, by scanning W from small to large values, we are able to find a certain point that best balances the two objectives. However, it is the decision maker's choice to finally determine the value of W, which depends on whether the relocation efficiency or the service fairness is emphasized more by the terminal operators.

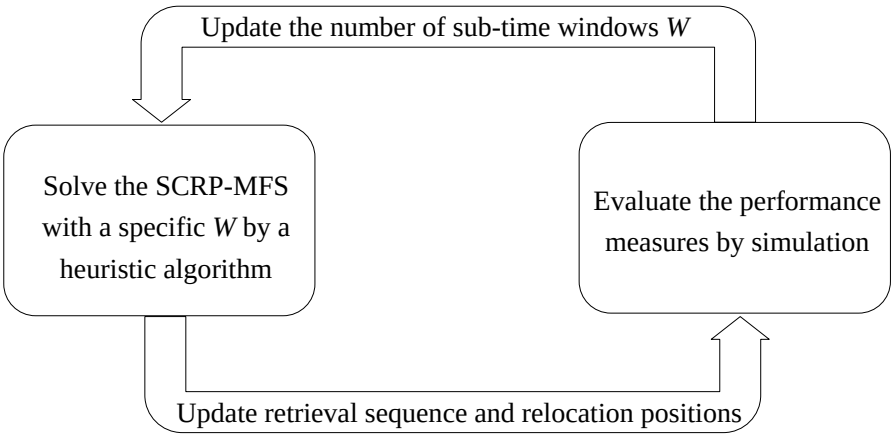


Fig. 3. 2 Framework of the model handling approach

3.4 Heuristic algorithm

It has been computationally proved that the SEM (Sequencing based Expected Minmax) heuristic is able to obtain near-optimal solutions of the SCRP-FS (Feng et al., 2020). In this section, we extend the SEM heuristic so that it can be applied to solve the SCRP-MFS. One of the main extensions we made to the SEM heuristic is on generalizing the BIS index and DIS index, which are two key criteria used for selecting the relocation positions. The details of the two indexes are referred to in section 3.4.2.2. In the SEM heuristic, the calculation methods of these two indexes are only applicable to the situation of two sub-batches. When considering multiple sub-batches, a general calculation method is needed. In addition, we embed a procedure to reduce the maximum truck turnaround time.

In the following, we first introduce the outline of the extended SEM heuristic algorithm. Then, we introduce the heuristic rules used in the algorithm. Last, we describe the computing methods of the BIS index and DIS index.

3.4.1 Algorithm outline

The extended SEM shares the same outline as the SEM. There are two decisions to be made: sequencing and relocating, which are made by two heuristic rules. First, by using the sequencing rule, the retrieval sequence is determined one container at a time. Then, according to the relocating rule, the relocation positions of the resulted blocking containers are determined.

The following notations are defined for describing the heuristic and used throughout this section. These notations are also included in Appendix A.2.

t_i: the *i*th target container to be retrieved, $i \in \{1,...,C\}$.

$n(u)$: the number of relocations needed for retrieving container *u*.

X: the bay configuration (state). Let X_0 represent the initial configuration.

lmin: the smallest priority label of containers in X.

Θ: the set of containers labelled *lmin* in X.

Φ_{kw}: the trucks in batch *k* that have arrived at sub-time window *w*.

The overall steps of the extended SEM heuristic are as follows:

Step 0. *Initialization*. Let $\mathrm{X} = \mathrm{X}_0$. Set *k* = 1, i.e., the index of the first batch and *i* = 1, i.e., the index of the first target container.

Step 1. *Reveal the truck arrival information*. If *k* > *K*, stop – all containers have been retrieved; otherwise, reveal the truck arrival information of batch *k*, that is, Φ_{kw}, $w \in \{1,...,W\}$, and go to Step 2.

Step 2. *Update the bay configuration*. Update X according to the revealed truck arrival information: for *w* from 2 to *W*, add $\sum_{w'=1}^{w-1} |\Phi_{kw'}|$ to the priority labels of each container in the sub-batch *w* of batch *k*.

Step 3. *Determine the target container*. Identify *lmin* and construct Θ. If $|\Theta| = 1$, let the only container in Θ be t_i; otherwise, determine t_i using the ***Sequencing Rule*** and update X accordingly.

Step 4. *Relocate the blocking containers*. Calculate $n(t_i)$. If $n(t_i) = 0$, go to step 5; otherwise, determine the relocation positions for each of the $n(t_i)$ blocking containers from top to bottom using the ***Relocating Rule*** and relocate these blocking containers accordingly, and as a result, X is updated.

Step 5. *Retrieve the target container*. Retrieve t_i from X. If $i = \sum_{j=1}^{k} C_j$, that is, all containers in batch *k* have been retrieved, then set *k* = *k* + 1 and go to step 1; otherwise, set *i* = *i* + 1, go to step 3.

In the above steps, the main extension we made is the way of updating the bay configuration in Step 2. When considering multiple sub-batches, the priority labels of the containers in each sub-batch *w* > 1 need to be updated according to the number of trucks in all of its former sub-batches.

3.4.2 Heuristic rules

The sequencing rule and the relocation rule used in the extended SEM algorithm are introduced in the following two sub-sections.

3.4.2.1 Sequencing rule

The main idea of the sequencing rule is to minimise the number of relocations needed in the next

retrieval, and thus the container with the least number of blocking containers is selected from the candidate containers as the target container. If more than one container has the same least number of blocking containers, the one with the earliest truck arrival time is selected as the target container in order to reduce the maximum truck turnaround time. The sequencing rule is presented below.

Step 1. *Identify the number of blocking containers of each candidate target container*. Given X, *lmin*, and Θ, compute the $n(u)$ of each container $u \in \Theta$.

Step 2. *Determine the target container*. Sort $\{n(u): u \in \Theta\}$ in non-decreasing order of $n(u)$. Choose the one with the smallest $n(u)$ in Θ as the target container t_i. If multiple ones are having the smallest $n(u)$, the one with the earliest truck arrival time is selected as the target container.

Step 3. *Update the bay configuration*. Update X by increasing the priority labels of the containers in $\Theta \setminus t_i$ by one.

The example in Fig. 3. 3 is used for illustration. In the example, W = 3, the length of a time window equals 30 minutes, and the system starting time equals zero. The priority matrix shows how the priority labels of the containers update as the retrieval proceeds. The preference matrix shows the customer preference of each container for each sub-time window. The container ID corresponds to the container in each slot. The truck arrival time matrix reveals the truck arrival time (in minutes) for each container. "×" represents that the truck arrival time for the corresponding container has not been revealed at the current step. The containers in bold represent that the truck arrival information of these containers has just been revealed at the current step. The containers in the shaded slot represent the target container to be retrieved. The containers in the striped slot represent the blocking containers to be relocated at the current step. At step 0 in Fig. 3. 3, the truck arrival information for the first batch is revealed to be that trucks u_6 and u_{10} have arrived at the same sub-time window that is earlier than that of truck u_{13}. The set of candidate target containers is $\Theta = \{u_6, u_{10}\}$ since both u_6 and u_{10} have the smallest priority label (i.e., 1). As u_6 and u_{10} have an equal number of blocking containers (i.e., one blocking container), the target container should be the one with the earlier truck arrival time. In our example, the truck arrival time of u_6 which equals 7 is earlier than that of u_{10} which equals 9. Therefore, u_6 is selected as the target container, and accordingly, the priority label of u_{10} is increased by one, leading to an updated priority label (i.e., 2) at step 2.

3.4.2.2 Relocating rule

We extend the two important indexes (i.e., BIS and DIS) in the SEM heuristic to the situations of multiple sub-batches. The blocking index with sequencing (BIS) of a stack *s*, denoted by *BIS*(*s*), is defined as the probability of a container being blocking if relocated to *s*. The delay index with sequencing (DIS) of a stack *s*, denoted by *DIS*(*s*), is defined as the probability of the containers with the highest priority in stack *s* being the first one to be retrieved within its batch. These two indexes are used for selecting the best relocation stack when ties on the candidate relocation stacks occur. They are particularly important for the problem with larger batches as the ties will occur frequently and need to be broken by some criteria. Before introducing the computing methods of the two indexes (see section 3.4.3), we first present the framework of the relocating rule in the following.

The following notations are defined for describing the relocating rule and used throughout this section. These notations are also included in Appendix A.1.

c: the blocking container that is to be relocated;

$\hat{s}$: the stack where the blocking container *c* is located before relocating;

m(*s*): the smallest priority label of a container in stack *s*, $s \in \{1,...,S\}$. For an empty stack, we let *m*(*s*)

equal C+1;

$h(s)$: the number of containers in stack s, $s \in \{1,...,S\}$;

S_C: the set of candidate stacks;

s^*: the selected relocation stack.

The heuristic rule that determines the relocation position for the blocking container c from stack $\hat{s}$ is described below.

[Condition 1] There is a non-full stack $s \neq \hat{s}$ that satisfies $m(s) > l_c$.

Let $M = \min_{s \in \{1,...,S\} \setminus \hat{s}} \{m(s) : h(s) < T, m(s) > l_c\}$. Select the stack that satisfies $m(s) = M$. Break ties by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

[Condition 2] For all non-full stacks $s \neq \hat{s}$, $m(s) \leq l_c$.

Let $M = \max_{s \in \{1,...,S\} \setminus \hat{s}} \{m(s) : h(s) < T\}$. The set of candidate stacks is represented by $S_C = \{s | s \in \{1,...,S\} \setminus \hat{s}, h(s) < T, m(s) = M\}$. If $|S_C| = 1$, select the only stack that satisfies $m(s) = M$. Otherwise, ties are broken in the following way. In the case of $M = l_c$, the stack in S_C with the minimum BIS is selected, that is, $s^* = \arg\min \{BIS(s) | s \in S_C\}$; In the case of $M < l_c$, the stack in S_C with the minimum DIS is selected, that is, $s^* = \arg\min \{DIS(s) | s \in S_C\}$. Further ties are broken by choosing from the highest ones, finally selecting the leftmost one if any ties remain.

The basic idea of the above relocating rule is to avoid or delay the blocking container being relocated again in the future. Two conditions arise, depending on whether the blocking container c needs to be relocated again in the future. In condition 1, since $m(s) > l_c$ (recall that l_c denotes the priority label of container c), c will not need to be relocated again in the future. In this condition, the stack with the minimum $m(s)$ is selected as the relocating stack in order to save the stacks with greater $m(s)$ for storing other blocking containers with greater priority labels. In condition 2, c will inevitably need to be relocated again. The stack with the maximum $m(s)$ is chosen to delay the next relocation of c as much as possible. In case of ties, the way how the ties are broken depending on the value of M. If $M = l_c$ (which means that c will be relocated again in the future with a probability), the stack with the minimum $BIS(s)$ is chosen to minimise the probability of c being relocated again. If $M < l_c$ (which means that c will surely be relocated again in the future), select the stack with the minimum $DIS(s)$ to delay the next relocation of c.

3.4.3 Two key indexes

In the following subsections 3.4.3.1 and 3.4.3.2, the methods of computing the two key indexes (BIS and DIS) used in the relocation rule are introduced.

3.4.3.1 Method of computing BIS

The BIS index is needed to break the first kind of tie in [Condition 2] where more than one candidate stacks satisfying $m(s) = M$ and $M = l_c$. Let M_s be the set of containers located in a candidate stack s and labelled M. Container c will block if relocated to s only in the situation where at least one container in M_s is in the sub-batches before the sub-batch of c. $BIS(s)$ is calculated by considering all scenarios of the sub-batches of c except for the scenario of the first sub-batch. The reason why there is no need to consider

the scenario of c in the first sub-batch is that if c is relocated to s in this scenario, according to the sequencing rule, c will be the first to be retrieved in stack s and thus will not block. The pseudocode of calculating $BIS(s)$ is given in Algorithm 1. In Algorithm 1, $p(w,u_i)$ denotes the probability that container u_i is in the sub-batches not before sub-batch w; $p(w)$ denotes the probability that all containers in M_s are in the sub-batches not before sub-batch w, and thus $(1- p(w))$ is the probability that at least one container in M_s is in the sub-batches before sub-batch w.

Algorithm 1: BIS

Input: c ← the blocking container to be relocated
M ← the maximum of the smallest priority label in each candidate stack
s ← the stack whose *BIS* is to be calculated
M_s ← the set of containers in stack s and with label M
W ← the number of sub-batches
$BIS(s) = 0$
for $w = 2$ to W **do**
  $p(w) = 1$
  for each container $u_i \in M_s$ **do**
    $p(w,u_i) = 0$
    for $k = w$ to W **do**
      $p(w,u_i) = p(w,u_i) + p_{ik}$
    end
    $p(w) = p(w) \cdot p(w,u_i)$
  end
  $BIS(s) = BIS(s) + p_{cw} (1- p(w))$
end
Return $BIS(s)$

Taking the step 1 in Fig. 3. 3 for example. At step 1, the target container to be retrieved is u_6 and the blocking container to be relocated is u_7. We need to determine the relocating stack for u_7. According to the state of step 1, $M = 4$ and the set of candidate stacks $S_C = \{1, 2\}$. By Algorithm 1, $BIS(1) = 0.2 \times (1-0.2) + 0.5 \times (1-0.1) = 0.61$; $BIS(2) = 0.2 \times (1-0.9 \times 0.8) + 0.5 \times (1-0.7 \times 0.6) = 0.346$. As $BIS(2) < BIS(1)$, stack 2 is selected as the relocation stack for u_7.

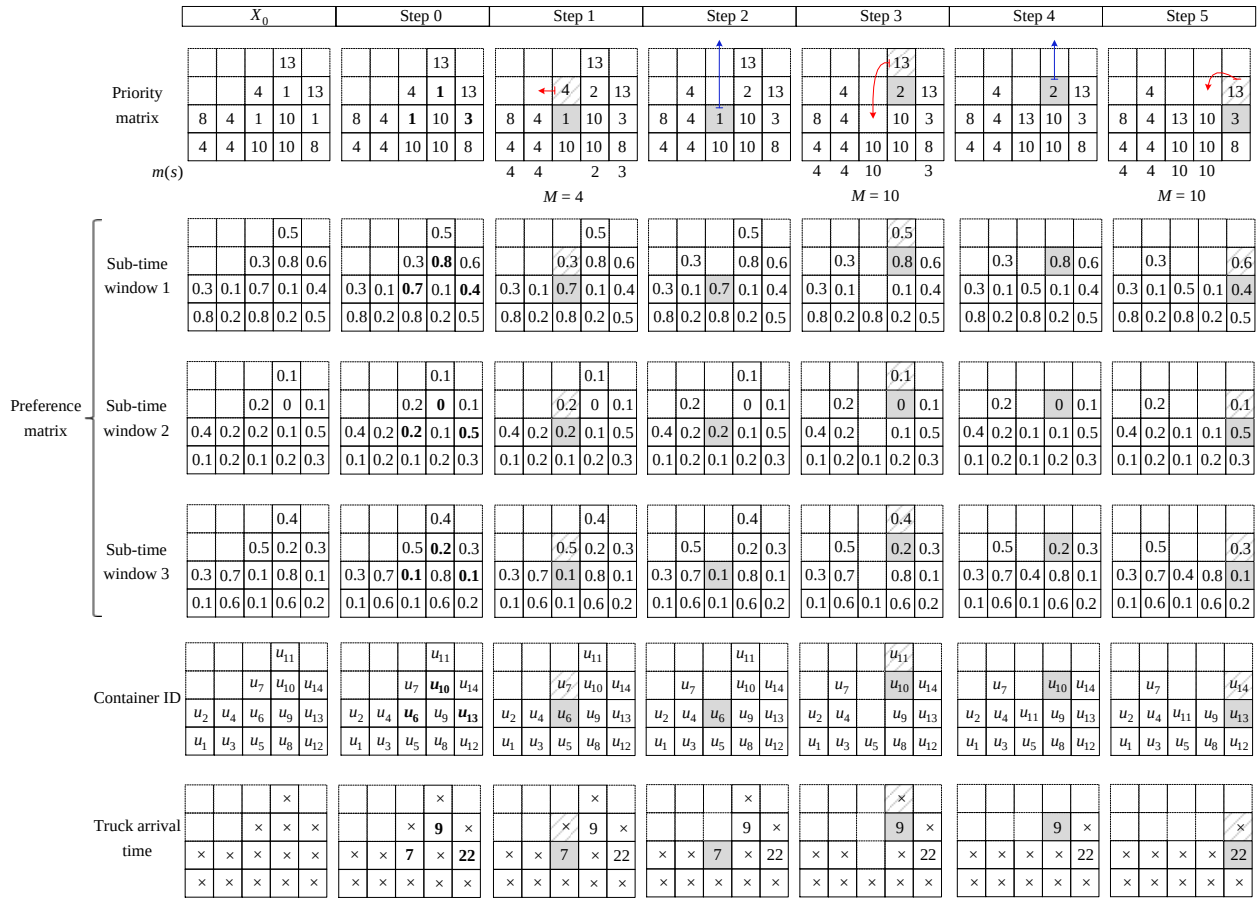


Fig. 3. 3 Decisions by the extended SEM heuristic on an example

3.4.3.2 Method of computing DIS

The DIS index is needed when a tie occurs in [Condition 2] where more than one candidate stacks s satisfying $m(s) = M$ and $M < l_c$. Since there may be more than one container with the same highest priority in a stack, $DIS(s)$ is calculated by $DIS(s) = \sum_{u_i \in M_s} DIS(s, u_i)$, where M_s denotes the set of containers located in stack s and with the highest priority, that is, the smallest label M. $DIS(s, u_i)$ denotes the probability of container u_i that has the smallest label in stack s being the first one to be retrieved within its batch. $DIS(s, u_i)$ is calculated by considering all the scenarios of the sub-batch of container u_i. In the scenario where u_i is in sub-batch w, u_i is surely being the first one to be retrieved within its batch when the following three conditions are satisfied simultaneously:

(i) the containers above u_i and with label M are in the sub-batches behind w;

(ii) the containers below u_i and with label M are in the sub-batches not before w;

(iii) for each of the other candidate stacks $s' \in S_C / s$, for each container u_j in stack s': if $n(u_j) \le n(u_i)$ (recall that $n(u_i)$ denote the number of blocking containers above u_i), u_j are in the sub-batches behind w; otherwise, u_j are in the sub-batches not before w.

The pseudocode of calculating $DIS(s)$ is shown in Algorithm 2. In Algorithm 2, lines 11-19, lines 20-28, and lines 29-47 are to calculate the probability of the above condition (i), (ii) and (iii), respectively. In line 48, $p_{iw} \cdot p(w, u_i)$ represents the probability that container u_i is in sub-batch w and is the first container to be retrieved within its batch. $DIS(s, u_i)$ is obtained by considering all the scenarios of the sub-batch of u_i (line 48). $DIS(s)$ is obtained by considering all the containers labelled M in stack s (line 50).

Algorithm 2: DIS

```
Input: B ← the state of the current bay
       M ← the maximum of the smallest priority label in each candidate stack
       W ← the number of sub-batches
       s ← the stack whose DIS is to be calculated; S_C ← the set of candidate stacks
       M_s ← the set of containers in stack s and with label M
DIS(s)= 0
for each container u_i ∈ M_s do
    DIS(s, u_i) = 0
    for w = 1 to W do
        p(w,u_i) = 1
        for each container u_j above u_i do
            if the label of u_j equals W then
                sum = 0
                for k = w + 1 to W do
                    sum = sum + p_jk
                end
                p(w,u_i) = p(w,u_i) ·sum
            end
        end
        for each container u_j below u_i do
            if the label of u_j equals W then
                sum = 0
                for k = w to W do
                    sum = sum + p_jk
                end
                p(w,u_i) = p(w,u_i) ·sum
            end
        end
        n(u_i) ← the number of blocking containers above u_i
        for each stack s' ∈ S_C / s do
            for each container u_j in stack s' do
                if the label of u_j equals W then
                    n(u_i) ← the number of blocking containers above u_i
                    sum = 0
                    if n(u_j) <= n(u_i) then
                        for k = w + 1 to W do
                            sum = sum + p_jk
                        end
                    else
                        for k = w to W do
                            sum = sum + p_jk
                        end
                    end
                    p(w,u_i) = p(w,u_i) ·sum
                end
            end
        end
        DIS(s, u_i) = DIS(s, u_i) + p_iw ·p(w,u_i)
    end
    DIS(s) = DIS(s) + DIS(s, u_i)
end
Return DIS(s)
```

Taking the step 5 in Fig. 3. 3 for example. At step 5, the blocking container to be relocated is u_{14}. M = 10 and the set of candidate stacks S_C = {3, 4}. By Algorithm 2, $DIS(3, u_5)$ = $0.8\times0.9\times0.8+0.1\times0.8\times0.6+0.1\times0\times0$ = 0.624; $DIS(4, u_9)$ = $0.1\times1\times1+0.1\times0.8\times0.2+0.8\times0.6\times0.1$ = 0.164; $DIS(4, u_8)$ = $0.2\times0.9\times0.2+0.2\times0.8\times0.1+0.6\times0\times0$ = 0.052. Therefore, $DIS(3) = DIS(3, u_5)$ = 0.624, and $DIS(4) = DIS(4, u_9) + DIS(4, u_8)$ = 0.216. As $DIS(4) < DIS(3)$, stack 4 is selected as the relocation stack for u_{14}.

3.5. Computational experiments

We perform three sets of experiments to achieve the research objectives raised in Section 3.1. Firstly, we evaluate the impact of the number of sub-time windows. Secondly, we investigate the impact of customer preference. Thirdly, we assess the value of customer preference information. The algorithm is coded in MATLAB 2018a and the experiments are performed on a desktop with Intel® Core ™ i5-6500 processor, 3.20 GHz CPU and 8 GB of RAM. The computational times of all the experiments are within several milliseconds.

Table 3. 2 lists the sets of parameters used in the experiments. The number of tiers (T) in a bay varies from 3 to 6 and the number of stacks (S) in a bay varies from 5 to 10, which covers the dimension of the bay in most container terminals. In total, we have 24 problem classes that are characterised by T and S. The utilisation rate (u) of the bay is set to be 67%. Given T, S and u, the number of containers in the bay is calculated by C = round(u*T*S), where round(x) rounds x to its closer integer. To provide a meaningful interpretation for the results, we consider a larger batch size, that is, there are on average six containers per batch. For each problem class, we have 30 instances that vary in the containers' stacking positions and the number of containers in each batch. These instances inherit from the corresponding instances in Feng et al. (2020), which were adapted from the CRPTW instances in Ku and Arthanari (2016a). Besides, the length of the appointment time window and the handling times per relocation move and per retrieval move also follow the settings in Feng et al. (2020).

Table 3. 2 Parameters setting for the experiments.

Parameter	Range of scenarios	Fixed parameters
Dimension of the bay (T×S)	[3, 6] ×[5, 10]	
Utilisation rate (u)		67%
Customer preference scenario (CPS)	{homogeneous, heterogeneous, exact}	
Average batch size		6
Length of an appointment time window		30 minutes
Time per relocation move		2 minutes
Time per retrieval move		4 minutes

Regarding the customer preference, we consider three Customer Preference Scenarios (**CPS**s) to characterise whether the customer preference information is available and whether the arrival sub-time windows are certain, which is detailed in Table 3. 3. In Scenario 1 of homogeneous CPS, the preference for each sub-time window w, $w \in \{1,...,W\}$, is evenly distributed, that is, each truck has an equal probability (i.e., 1/W) to arrive at each sub-time window. This scenario is equivalent to no customer preference information and is used as the baseline. In Scenario 2 of heterogeneous CPS, the probabilities of a truck arriving at different sub-time windows follow the distribution of p_w. We assume that p_w is generated from the uniform distribution $U(0,1)$. This scenario represents the situation where each truck has different

preferences for different sub-time windows. In the above two scenarios, the trucks will arrive at each sub-time window with a probability. In Scenario 3 of exact CPS, each truck is uniformly assigned to one of the sub-time windows which it will arrive at with 100% probability. In this scenario, we know which sub-time window each truck will arrive at exactly.

Table 3. 3 Customer Preference Scenarios (CPSs).

	Probabilities					Preference information availability	Arrival sub-time windows
Sub-time windows / Scenarios	1	…	*w*	…	*W*		
1. Homogeneous	1/*W*	1/*W*	1/*W*	1/*W*	1/*W*	Unavailable	Uncertain
2. Heterogeneous	p_1	…	p_w	…	p_W	Available	Uncertain
3. Exact	0	0	1	0	0	Available	Certain

In order to estimate the objective values, we need to sample the customer preference and the truck arrival times. The samples are generated through a Monte Carlo simulation. The number of samples required to obtain a relative error γ is calculated by

$$n(\gamma) = \delta^2 \left((1+\gamma) Z_{1-\alpha/2} / \gamma\mu\right)^2 \tag{3.8}$$

where δ^2 and μ respectively represent the variance and mean of the objective values and $Z_{1-\alpha/2}$ is the 1 - α/2 percentile of the normal distribution (Law and Kelton, 2000). In our experiments, we want to estimate the total number of relocations, the average truck turnaround time, and the maximum truck turnaround time, with a relative error of 5% (γ = 5%) and a confidence level of 90% (α = 10%) respectively for each indicator. We conduct two-stage preliminary experiments to calculate, first, the number of the samples of the truck arrival times required, denoted by $n_1(\gamma)$, and then, that of the customer preference, denoted by $n_2(\gamma)$, for each problem class under the three CPSs, respectively. Note that for Scenario 1 of homogeneous CPS, we do not need to calculate $n_2(\gamma)$ as there is only one possibility of the customer preference (that is, the probability for each sub-time window is equal) and thus $n_2(\gamma)$ = 1. At the first stage, given a random instance and a random sample of customer preference, we calculate δ^2 and μ for each performance indicator based on ten random samples of truck arrival times. Then, using Eq. (3.8), $n_1(\gamma)$ is obtained by taking the greatest value among the number of samples required for each performance indicator. At the second stage, given an instance, we calculate δ^2 and μ based on ten random samples of customer preference (the result of each sample of customer preference is the average over $n_1(\gamma)$ random samples of truck arrival times). Then, using Eq. (3.8), we obtain $n_2(\gamma)$ by taking the greatest value among the number of samples required for each performance indicator. Depending on parameters *T*, *S*, *W* and the CPS, the values of $n(\gamma)$ vary significantly between zero and several hundred. For example, for the problem class with *T* =3, *S* = 5 and *W* =2 under Scenario 2 of heterogeneous CPS, $n_1(\gamma)$ = 555 and $n_2(\gamma)$ = 30; while for the same problem class under the Scenario 3 of exact CPS, $n_1(\gamma)$ = 12 and $n_2(\gamma)$ = 537. If the resulted $n(\gamma)$ is less than five, we force $n(\gamma)$ to be five, which means that we make at least five repetitions respectively on the customer preference and truck arrival times for each instance.

The parameters of the developed hierarchical iterative approach for handling the model are selected to be: Q_1 = 2, Q_2 = 5, and W_0 = 2, by trials based on the experiment parameters.

3.5.1 Impact of the number of sub-time windows

In this section, we evaluate the impact of the number of sub-time windows on system overall efficiency and service fairness. Apart from the total number of relocations and the maximum truck turnaround time

that are optimised, two more evaluation indicators are proposed to evaluate the performance and listed as follows:

Average truck turnaround time (AveT): AveT is the mean of the total turnaround times of all the trucks, which indicates the turnaround time for each truck on average. AveT has a positive correlation with the total number of relocations and can also represent system overall efficiency.

Coefficient of variation of the truck turnaround time (CVT): Coefficient of variation (CV), also known as relative standard deviation, is defined as the ratio of the standard deviation to the mean, which is a standardized measure of the dispersion of a probability distribution or frequency distribution. As the means of the turnaround times under different numbers of sub-time windows are different, we use the CV to show the extent of variability of the turnaround time in relation to its mean. CVT has a positive correlation with the maximum truck turnaround time, which can also represent service fairness.

The experiments in this section are based on Scenario 1 of homogeneous CPS. Multiple problem classes are constructed to execute the experiments; these classes are characterised by different combinations of the number of tiers (T), the number of stacks (S), and the number of containers (C) (see Table 3. 4 and Table 3. 5). The results show that there is no ultimate solution to achieve the lowest number of relocations and the highest service fairness. The SCRP-MFS where each appointment time window is divided into two sub-time windows (i.e., $W = 2$) is regarded as the benchmark. For each problem class, the results obtained under different numbers of sub-time windows (in the range of [3, 6]) are compared with the benchmark. In Table 3. 4, columns "Rel" and "AveT" respectively give the total number of relocations and the average truck turnaround time for a problem class in the benchmark, which are obtained by taking the average of 30 instances, each one containing $n_1(\gamma)$ samples. Under the scenarios of "$W = 3$", "$W = 4$", "$W = 5$" and "$W = 6$", column "Rel%" reports the relative difference between the total number of relocations of the considered scenario and the benchmark, indicating the percentage increase of the total number of relocations when compared with the benchmark; column "AveT%" reports the relative difference between the average truck turnaround time of the considered scenario and the benchmark, revealing the percentage increase in the average turnaround time in comparison with the benchmark.

Table 3. 4 Comparisons of the total number of relocations and average truck turnaround times under different numbers of sub-time windows with the benchmark.

Problem class*[1]			$W = 2$[*2]		$W = 3$		$W = 4$		$W = 5$		$W = 6$	
T	S	C	Rel	AveT	Rel%	AveT%	Rel%	AveT%	Rel%	AveT%	Rel%	AveT%
3	5	10	2.27	29.16	14.0%	1.5%	21.1%	2.4%	25.5%	2.8%	27.6%	3.1%
	6	12	3.09	30.62	9.6%	1.5%	14.3%	2.1%	18.9%	3.0%	20.2%	3.0%
	7	14	3.35	31.15	9.0%	1.5%	13.7%	2.2%	16.2%	2.5%	18.1%	2.9%
	8	16	4.04	30.64	7.5%	0.9%	10.2%	1.8%	12.6%	2.2%	14.1%	2.4%
	9	18	4.41	31.00	6.6%	1.7%	11.0%	2.6%	11.2%	**2.3%**	12.2%	2.7%
	10	20	4.74	29.48	5.6%	1.1%	8.3%	1.6%	9.4%	1.9%	10.1%	2.0%
4	5	13	4.33	32.65	9.7%	2.0%	17.7%	3.3%	18.9%	4.0%	20.6%	4.1%
	6	16	6.28	32.78	6.8%	2.0%	10.5%	3.0%	12.1%	3.6%	14.4%	4.1%
	7	19	6.59	30.98	6.3%	1.7%	10.8%	2.5%	12.1%	3.0%	13.8%	3.4%
	8	21	7.02	32.66	5.7%	1.7%	9.6%	2.7%	**9.5%**	3.0%	11.7%	3.2%
	9	24	8.80	32.72	5.3%	1.9%	**5.0%**	**1.7%**	7.6%	2.8%	7.9%	**2.7%**
	10	27	9.30	31.47	4.4%	1.1%	6.6%	2.1%	7.4%	2.5%	7.9%	2.7%
5	5	17	8.57	34.92	7.4%	2.4%	10.4%	4.0%	12.5%	4.5%	14.7%	5.3%

	6	20	9.02	34.61	6.4%	2.5%	9.9%	3.5%	12.2%	4.3%	12.2%	4.5%
	7	23	10.89	34.80	4.0%	1.7%	7.5%	3.3%	7.9%	3.4%	10.4%	4.9%
	8	27	12.61	34.00	4.8%	2.1%	6.3%	2.7%	7.7%	3.7%	8.8%	3.9%
	9	30	14.59	36.19	3.5%	1.6%	6.2%	3.3%	6.2%	3.3%	**6.0%**	3.7%
	10	34	15.32	34.15	3.8%	1.9%	4.3%	2.3%	5.8%	3.0%	6.5%	3.3%
6	5	20	12.61	35.79	6.9%	3.9%	10.6%	5.5%	11.7%	6.3%	12.0%	**6.2%**
	6	24	13.77	37.01	5.4%	2.5%	8.1%	4.6%	9.3%	5.3%	11.3%	6.1%
	7	28	16.69	40.19	3.2%	2.2%	5.2%	3.4%	7.0%	4.5%	7.0%	4.5%
	8	32	18.01	39.68	3.9%	2.3%	5.6%	3.7%	8.2%	5.1%	8.7%	5.3%
	9	36	19.18	37.41	3.7%	2.6%	5.4%	3.7%	6.2%	3.9%	7.2%	4.5%
	10	40	22.33	37.55	0.7%	1.0%	3.4%	2.9%	4.6%	3.7%	4.6%	**3.5%**

*1: The CPS is "homogeneous"; *2: "$W = 2$" is the benchmark.

In Table 3. 5, columns "MaxT" and "CVT" respectively report the maximum truck turnaround time and the CV of the truck turnaround time for a problem class in the benchmark, which is obtained by taking the average of 30 instances each of which contains $n_1(\gamma)$ samples. Columns "MaxT%" and "CVT%" respectively represent the percentage reduction in the maximum turnaround time and the CV of the turnaround time when compared with the benchmark.

Table 3. 5 Comparisons of the maximum truck turnaround times and the coefficients of variation under different numbers of sub-time windows with the benchmark.

Problem class*1			$W = 2$*2		$W = 3$		$W = 4$		$W = 5$		$W = 6$	
T	*S*	*C*	MaxT	CVT	MaxT%	CVT%	MaxT%	CVT%	MaxT%	CVT%	MaxT%	CVT%
3	5	10	39.57	0.25	-2.7%	-7.3%	-3.8%	-10.1%	-4.1%	-10.8%	-4.5%	-11.1%
	6	12	41.55	0.24	-2.5%	-6.1%	-3.8%	-9.4%	-4.1%	-10.7%	-4.5%	-10.5%
	7	14	42.92	0.25	-2.7%	-4.1%	-3.7%	-6.0%	-4.2%	-6.6%	-4.5%	-7.8%
	8	16	42.83	0.24	-2.8%	-4.7%	-3.8%	-7.2%	-4.5%	-8.6%	-5.0%	**-8.1%**
	9	18	44.03	0.25	-3.1%	-4.3%	-3.8%	-7.1%	-4.7%	-8.0%	-5.1%	**-7.6%**
	10	20	42.27	0.24	-3.1%	-4.9%	-4.2%	-7.6%	-5.2%	-9.1%	-5.5%	**-9.0%**
4	5	13	44.73	0.25	-3.0%	-5.7%	-3.5%	-8.4%	-4.2%	-9.8%	-4.8%	-9.8%
	6	16	46.74	0.26	-3.0%	-8.4%	-3.9%	-10.8%	-4.4%	-11.8%	-4.6%	**-11.3%**
	7	19	44.59	0.26	-2.4%	-3.2%	-3.8%	-7.3%	-4.6%	-8.2%	-5.0%	-8.7%
	8	21	47.24	0.25	-3.3%	-8.0%	-4.3%	-9.8%	-4.4%	**-7.3%**	-5.1%	-10.2%
	9	24	48.25	0.26	-2.9%	-5.4%	-4.9%	-9.4%	**-4.8%**	-9.5%	-5.8%	-10.0%
	10	27	46.13	0.25	-2.7%	-5.3%	-3.3%	-6.9%	-4.2%	-7.9%	-4.9%	-8.8%
5	5	17	49.78	0.27	-2.7%	-7.5%	-3.3%	-8.0%	-3.8%	-10.0%	-4.0%	-10.6%
	6	20	49.80	0.27	-2.2%	-5.1%	-3.2%	-7.6%	-3.5%	-8.0%	-3.5%	**-7.5%**
	7	23	50.43	0.27	-2.7%	-6.4%	-2.9%	-7.5%	-4.3%	-9.2%	**-3.7%**	**-8.7%**
	8	27	51.01	0.27	-2.6%	-3.6%	-4.5%	-9.4%	**-4.3%**	**-8.0%**	-5.2%	-10.3%
	9	30	53.98	0.29	-3.2%	-5.3%	-3.7%	-5.4%	-4.6%	-6.6%	**-4.5%**	**-5.1%**
	10	34	51.91	0.30	-2.3%	-2.0%	-4.3%	-6.1%	-4.7%	**-6.0%**	**-4.4%**	**-5.9%**
6	5	20	51.26	0.26	-1.3%	-5.1%	-1.7%	-9.4%	-2.2%	-10.9%	-2.8%	**-10.3%**
	6	24	54.21	0.27	-2.0%	-5.9%	-2.2%	-8.4%	-3.2%	-10.1%	**-2.7%**	-10.4%
	7	28	59.28	0.31	-2.5%	-5.2%	-3.5%	-7.4%	**-3.1%**	-8.9%	-3.6%	-9.0%

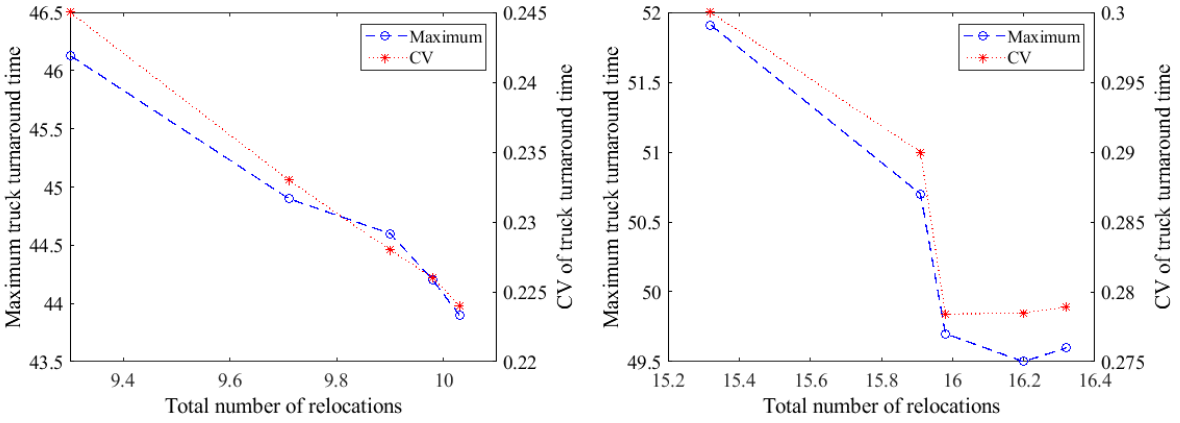
8	32	59.67	0.31	-3.1%	-3.9%	-3.4%	-6.6%	-3.5%	**-6.0%**	-3.9%	-7.3%
9	36	58.27	0.30	-3.2%	-5.6%	-4.1%	-7.3%	-4.7%	-9.9%	-4.7%	**-8.9%**
10	40	58.15	0.33	-2.8%	-2.9%	-3.6%	**-2.8%**	**-3.4%**	-3.5%	**-3.5%**	-4.8%

*1: The CPS is "homogeneous"; *2: "$W = 2$" is the benchmark.

From Table 3. 4 and Table 3. 5, we can see that the number of sub-time windows has a significant impact on the results of the SCRP-MFS. Firstly, the total number of relocations and the average truck turnaround times increase as the number of sub-time windows increases, which is as expected. The reason is that when each appointment time window is divided into more sub-time windows, the flexibility to optimise the service sequence shrinks. When out-of-order retrieval is implemented within a smaller sub-time window, there are fewer tucks in each sub-time window, and thus the opportunity for optimising the retrieval sequence decreases. It is noticed that for a few problem classes (highlighted in bold in Table 3. 4), the total number of relocations and the average truck turnaround times become smaller as the number of sub-time windows increases, which is counterintuitive. There are two possible reasons that can account for that. The first reason could be sampling bias. The second reason may be the use of the heuristic algorithm that generates near-optimal solutions.

Secondly, by increasing the number of sub-time windows, generally, both the maximum truck turnaround time and the CV of the turnaround time decrease, but for some problem classes, they show an increasing trend after a certain number of sub-time windows. The fundamental reason for the decreasing trend is the same as in Table 3. 4. Because the scope of out-of-order services is narrowed, the differences between the arrival times of trucks in a sub-batch get smaller, and thus the extra waiting times incurred to the early arriving trucks when later arriving trucks are served before them are shortened. On the other hand, the increasing trend is not surprising, which is attributed to the increasing number of relocations when the flexibility of the out-of-order retrieval decreases. In our experiment, when the number of sub-time windows is six, the flexible service policy is almost reduced to the FCFS policy as on average each sub-batch has only one truck. In this case, since there is little scope for out-of-order service, a truck may experience many relocations and thus have a longer turnaround time; and as some trucks may experience excessively long turnaround times, the variance of turnaround time may also increase.

To have a visual understanding of the impacts of the number of sub-time windows, we take two representative problem classes and display the trade-off between the total number of relocations, the maximum truck turnaround time, and the CV of truck turnaround time in Fig. 3. 4. Each point in Fig. 3. 4 corresponds to the results under a specific number of sub-time windows (from two to six). As shown in Fig. 3. 4(a), both the maximum value and the CV of the truck turnaround time exhibit a decreasing trend as the total number of relocations increases (with the number of sub-time windows increasing from two to six). While in Fig. 3. 4(b), as the number of sub-time windows increases, these two performance measures first decrease when the number of sub-time windows is within four and then tend to increase when the number of sub-time windows is over four.



(a) $T = 4$, $S = 10$ (b) $T = 5$, $S = 10$

Fig. 3. 4 Trade-off between total number of relocations, maximum truck turnaround time, and CV of truck turnaround time

Moreover, it is observed that when the number of sub-time windows is over four, increasing the number of sub-time windows would not have a great influence on the results of the SCRP-MFS. The reason is that the average numbers of trucks per sub-batch are very small, and as a result, their flexibilities of out-of-order retrievals are similar. Note that in our experiments, there are on average six trucks per batch. For the scenarios of $W = 5$ and 6, there is on average 1.2 and 1 truck per sub-batch, respectively. As shown in Fig. 3. 4, when W is within four, the relevant measures exhibit a significant changing trend with the increase of W. As W continues to increase, the changing trend tends to be gentle.

Furthermore, we can observe that the changing trend of the values of "Rel%" and "CVT%" in that, generally, the smaller the S and T, the greater (absolute) values of "Rel%" and "CVT%". Regarding "AveT%" and "MaxT%", as T increases, the absolute values of "MaxT%" show a slightly decreasing trend while the values of "AveT%" show an opposite trend. The values of "MaxT%" are not sensitive to S while the values of "AveT%" decrease as S increases. These observations indicate that, generally, the number of sub-time windows has a more significant influence on the bay with smaller dimensions. This is because smaller-scale bays are more sensitive to the flexibility of the out-of-order retrieval. To be specific, in a narrower bay (i.e., smaller S), there is less opportunity to find a good stack for the relocated container that will not cause future relocations; and in a lower bay (i.e., smaller T), as the containers are stacked in lower stacks, a larger proportion of relocations can be avoided by out-of-order retrievals. In both cases, it relies more on the optimisation of the retrieval sequence to reduce the number of relocations.

Above all, the results in the above two tables indicate that applying a larger number of sub-time windows is beneficial to reducing the maximum value and the CV of the truck turnaround time but at the expense of a high number of relocations and average truck turnaround time. For terminals that are more customer-centric or use a larger bay layout, a relatively larger number of sub-time windows could be chosen; while on the other hand, if the number of relocations tends to be the primary performance metric of the terminals or a smaller bay layout is used, applying a relatively smaller number of sub-time windows appears to be a good choice. It should be noted that the service fairness is not necessarily increasing as the number of sub-time windows increases. Generally, applying three or four sub-time windows seems reasonable, which can have a good balance between the relocation efficiency and the service fairness.

3.5.2 Impact of the customer preference scenario

In this section, we investigate whether the SCRP-MFS is sensitive to the CPSs by comparing the total

number of relocations under different CPSs. As the direct effect of the flexible service policy is reducing the number of relocations, here, we only compare the metric of relocations.

Table 3. 6 compares the total number of relocations of the SCRP-MFS with W = 2 under different CPSs. The results for W = 3 - 6 are given in Appendix B. The total number of relocations under the heterogeneous CPS and the exact CPS for each problem class are obtained by taking the average of 30 instances, each instance containing $n_1(\gamma)$ samples of customer preference, each customer preference sample containing $n_2(\gamma)$ samples of truck arrival times. The middle three columns report the relative difference of the total number of relocations between each pair of CPSs, which is calculated based on the results under the former CPS. For example, for the column "Ho vs. He", Gap = (heterogeneous [Rel] – homogeneous [Rel]) / homogeneous [Rel]×100%. The last three columns report the p-value of the paired t-test for each pair of CPSs. At a 5% significance level, the results that differ significantly between the two CPSs are highlighted in Table 3. 6.

Table 3. 6 Comparisons of the total number of relocations between different customer preference scenarios (W = 2).

Problem class			Gap			p-Value		
T	*S*	*C*	Ho vs. He	Ho vs. Ex	He vs. Ex	Ho vs. He	Ho vs. Ex	He vs. Ex
3	5	10	0.06%	-1.31%	-1.36%	0.92	**0.03***	0.06
	6	12	-0.48%	-1.62%	-1.15%	0.47	**0.03***	0.20
	7	14	-0.83%	0.57%	1.42%	**0.05***	0.45	0.11
	8	16	0.88%	-0.18%	-1.05%	0.25	0.89	0.46
	9	18	-0.46%	-0.19%	0.27%	0.53	0.77	0.73
	10	20	-1.40%	0.58%	2.00%	**0.02**	0.21	**0.01***
4	5	13	0.32%	-2.28%	-2.59%	0.75	**0.00***	**0.01***
	6	16	-1.30%	-2.09%	-0.81%	**0.04***	**0.00***	0.27
	7	19	-0.20%	-0.23%	-0.04%	0.77	0.81	0.97
	8	21	-0.52%	-0.07%	0.46%	0.22	0.89	0.41
	9	24	-0.88%	-0.79%	0.10%	0.09	0.21	0.89
	10	27	-0.37%	-0.37%	0.00%	0.39	0.65	1.00
5	5	17	-1.73%	-3.40%	-1.71%	**0.01***	**0.00***	0.08
	6	20	0.68%	-0.64%	-1.30%	0.28	0.25	**0.04***
	7	23	0.10%	-1.07%	-1.17%	0.88	0.08	0.16
	8	27	-0.65%	-0.12%	0.53%	**0.05***	0.81	0.29
	9	30	-0.34%	-0.62%	-0.28%	0.37	0.37	0.66
	10	34	0.31%	0.03%	-0.27%	0.33	0.93	0.46
6	5	20	-0.16%	-2.92%	-2.76%	0.81	**0.00***	**0.00***
	6	24	-0.96%	-2.31%	-1.36%	0.16	**0.05***	0.15
	7	28	-0.59%	-0.63%	-0.04%	0.20	0.37	0.95
	8	32	-0.51%	-0.62%	-0.11%	0.31	0.23	0.82
	9	36	-0.93%	0.54%	1.48%	**0.02***	0.53	0.06
	10	40	-0.16%	0.60%	0.76%	0.61	0.30	0.23

Notes: "Ho vs. He" represents homogeneous vs. heterogeneous; "Ho vs. Ex" represents homogeneous vs. exact; "He vs. Ex" represents heterogeneous vs. exact. * Significance level =5%.

From Table 3. 6 and the tables in Appendix B, some interesting observations can be identified. Firstly, the gap values in columns "Ho vs. Ex" and "He vs. Ex" show that, generally, the total number of

relocations under the exact CPS is lower than that under the homogeneous and heterogeneous CPSs. This is because that the influence of the truck arrival uncertainties on the service sequence can be avoided under the exact CPS. Under this CPS, since each truck will arrive at a certain sub-time window, regardless of the truck arrival times, under the flexible service policy, a sub-batch of trucks will be served in a deterministic order. Secondly, particularly, the results of the homogeneous CPS (Ho) and the exact CPS (Ex) significantly differ at a 5% level for the problem classes with a narrower bay (S = 5, 6). This significance can be explained by the frequent occurrence of [Condition 2] in the heuristic (section 3.4.2.2) for a bay that has fewer stacks. For a bay with a smaller S, there is less opportunity to find a good stack for the relocated container that will not cause future relocations, and thus [Condition 2] will occur frequently. Therefore, the BIS and DIS indexes need to be used frequently to determine the relocation stacks. The calculation of these two indexes relies on customer preference information. The exact CPS can lead to a more accurate value of the two indexes and thus a better decision on the relocation stacks.

Since the difference values for all the problem classes in Table 3. 6 and the results in Appendix B are less than 4.1% and in most of the cases the differences are not significant, it can be claimed that the SCRP-MFS is not sensitive to the CPSs under consideration. This finding complements the results of the corresponding experiments in Feng et al. (2020). Their study shows that the number of relocations is much lower under the CPSs of "100%" and "0%" when compared with those under the homogeneous CPS. The "100%" CPS and "0%" CPS represent that all trucks will arrive at the first sub-time window and the second sub-time window, respectively, both of which provide the largest opportunity for optimising the retrieval sequence. It should be noticed that these two sensitive CPSs are two extreme cases that are assumed just for evaluating the maximum effectiveness of the flexible service policy but probably will rarely occur in reality, and thus they cannot represent the general situation.

3.5.3 Impact of information availability

In this section, we assess the value of the customer preference information. The information about the customer preference may not always be available. Truckers may not be able to provide their exact arrival probabilities for each sub-time window. To acquire the customer preference, terminals need to keep track of the historical arrival data of each truck. When the customer preference information is unavailable, the terminal operators will assume that the preference for each sub-time window is the same, which is equivalent to the homogeneous CPS. However, in reality, the trucks' arrival behaviour may cohere with the heterogeneous CPS or the exact CPS. Therefore, the decisions made under the assumption of homogeneous CPS may perform badly under the heterogeneous CPS and the exact CPS. In order to understand whether it is necessary to gather and utilise the customer preference information, we assess the value of such information by comparing the results of the SCRP-MFS that utilises the preference information and that does not utilise the information. The results of the SCRP-MFS without customer information are obtained as follows. First, we make decisions under the assumption of the homogeneous CPS, and then we apply the decisions to the simulated situation where the trucks arrive under the heterogeneous CPS or exact CPS to obtain the total number of relocations.

Table 3. 7 and Table 3. 8 respectively show the impacts of the availability of the heterogeneous information (Scenario 2) and the exact information (Scenario 3) on the SCRP-MFS. Column "Gap" reports the relative difference of the total number of relocations obtained by utilising the customer preference information (Inf [Rel]) and not utilising the information (NoInf [Rel]), and it is calculated as Gap = (NoInf [Rel] – Inf [Rel]) / NoInf [Rel]$\times$100%. Column "p-value" reports the p-value of the paired t-test of using and not using the information (significance level = 5%).

Table 3. 7 Comparisons of the total number of relocations between utilising the "exact" information and not utilising such information.

Problem class			$W = 2$		$W = 3$		$W = 4$		$W = 5$		$W = 6$	
T	S	C	Gap	p-value	Gap	p-value	Gap	p-value	Gap	p-value	Gap	p-value
3	5	10	1.1%	0.00	1.4%	0.00	1.1%	0.00	1.2%	0.00	1.2%	0.00
	6	12	1.3%	0.00	1.9%	0.00	1.6%	0.00	2.0%	0.00	2.1%	0.00
	7	14	0.6%	0.00	0.9%	0.00	1.0%	0.00	1.0%	0.00	0.9%	0.00
	8	16	0.2%	**0.26**	0.8%	0.01	0.6%	0.02	0.9%	0.01	1.0%	0.01
	9	18	0.7%	0.05	0.6%	0.05	0.4%	0.01	0.8%	0.02	0.6%	0.01
	10	20	0.2%	0.03	0.3%	0.02	0.4%	0.03	0.5%	0.01	0.3%	0.04
4	5	13	2.3%	0.00	1.9%	0.00	2.0%	0.00	2.1%	0.00	2.1%	0.00
	6	16	1.2%	0.00	0.9%	0.00	1.5%	0.00	1.6%	0.00	1.7%	0.00
	7	19	0.7%	0.00	1.3%	0.00	1.2%	0.00	1.0%	0.00	1.1%	0.00
	8	21	0.8%	0.00	1.0%	0.00	0.9%	0.00	1.0%	0.00	1.0%	0.00
	9	24	0.5%	0.01	0.6%	0.00	0.5%	0.00	0.7%	0.00	0.5%	0.00
	10	27	0.4%	0.01	0.6%	0.02	0.6%	0.00	0.6%	0.00	0.5%	0.00
5	5	17	3.0%	0.00	2.8%	0.00	2.6%	0.00	3.2%	0.00	2.4%	0.00
	6	20	1.3%	0.00	1.5%	0.00	1.5%	0.00	1.7%	0.00	1.5%	0.00
	7	23	1.6%	0.00	1.8%	0.00	1.8%	0.00	2.2%	0.00	2.0%	0.00
	8	27	1.0%	0.00	0.9%	0.00	1.1%	0.00	1.3%	0.00	1.5%	0.00
	9	30	0.7%	0.00	0.5%	0.00	0.8%	0.00	0.6%	0.01	0.6%	0.00
	10	34	0.3%	0.01	0.4%	0.00	0.4%	0.00	0.5%	0.00	0.6%	0.00
6	5	20	2.4%	0.00	2.2%	0.00	2.5%	0.00	2.4%	0.00	2.1%	0.00
	6	24	2.2%	0.00	2.0%	0.00	1.7%	0.00	1.6%	0.00	1.7%	0.00
	7	28	0.7%	0.00	1.0%	0.00	0.9%	0.00	1.2%	0.00	1.1%	0.00
	8	32	0.6%	0.01	0.9%	0.00	0.9%	0.00	0.9%	0.00	0.7%	0.01
	9	36	0.5%	0.01	0.9%	0.00	0.9%	0.00	1.1%	0.00	1.1%	0.00
	10	40	0.4%	**0.11**	0.7%	0.01	0.7%	0.01	0.8%	0.02	0.6%	0.04

Note: The bold numbers represent not significantly differing at a 5% level.

In Table 3. 7, the p-values show that, in most cases (118 out of 120 problem classes), the results of utilising the exact information and not utilising such information significantly differ at a 5% level. However, their relative differences (see Gap) are small, which vary between 0.2% and 3.2%. In addition, regarding whether or not to utilise the heterogeneous information, Table 3. 8 shows that about one-fifth of problem classes (24 out of 120) do not show a significant difference (p-values >0.05). A common feature in these non-significant problem classes is that their bays generally have more stacks (e.g., S = 9, 10). Another observation from the two tables is that in both scenarios, the bays with fewer stacks have greater gap values, and in the exact scenario, the gap values show an increasing trend as the number of tiers increase. The reason is that in the problem classes with fewer stacks and higher tiers, the BIS and DIS indexes are used more frequently to determine the relocation stacks. The calculation of the two indexes relies on the preference information, but the assumed equal preference may lead to bad decisions on the relocation stacks and thus result in more relocations. Moreover, it is observed that the gap values in Table 3. 8 are smaller than Table 3. 7 and are no more than 1.3%. One explanation for this phenomenon could be that the differences in the values of the two indexes between the exact CPS and the homogeneous CPS are larger than that between the heterogeneous CPS and the homogeneous CPS, given that the information in the exact CPS is more accurate.

Table 3. 8 Comparisons of the total number of relocations between utilising the "heterogeneous" information and not utilising such information.

Problem class			$W = 2$		$W = 3$		$W = 4$		$W = 5$		$W = 6$	
T	S	C	Gap	p-value	Gap	p-value	Gap	p-value	Gap	p-value	Gap	p-value
3	5	10	0.5%	0.00	0.4%	0.00	0.3%	0.00	0.3%	0.00	0.2%	0.00
	6	12	0.7%	0.00	0.7%	0.00	0.6%	0.00	0.5%	0.00	0.6%	0.00
	7	14	0.3%	0.00	0.3%	0.00	0.2%	0.00	0.2%	0.00	0.2%	0.00
	8	16	0.4%	0.02	0.3%	**0.07**	0.2%	**0.17**	0.3%	0.02	0.3%	**0.08**
	9	18	0.3%	**0.06**	0.2%	**0.09**	0.2%	0.03	0.2%	**0.06**	0.0%	**0.54**
	10	20	0.2%	0.01	0.2%	**0.06**	0.2%	0.01	0.2%	**0.06**	0.1%	0.04
4	5	13	0.7%	0.00	0.5%	0.00	0.4%	0.00	0.4%	0.00	0.3%	0.04
	6	16	0.7%	0.00	0.6%	0.00	0.5%	0.00	0.3%	0.00	0.3%	0.00
	7	19	0.5%	0.00	0.5%	0.00	0.4%	0.01	0.3%	0.03	0.3%	0.00
	8	21	0.5%	0.00	0.5%	0.00	0.2%	0.00	0.2%	0.02	0.3%	0.00
	9	24	0.3%	0.03	0.3%	0.01	0.3%	0.00	0.2%	0.00	0.1%	0.03
	10	27	0.1%	**0.31**	0.1%	**0.25**	0.2%	0.04	0.2%	0.02	0.2%	0.01
5	5	17	1.3%	0.00	0.6%	0.00	0.7%	0.00	0.5%	0.00	0.4%	0.00
	6	20	0.7%	0.00	0.6%	0.00	0.5%	0.00	0.4%	0.00	0.3%	0.02
	7	23	1.0%	0.00	0.7%	0.00	0.8%	0.00	0.5%	0.00	0.5%	0.00
	8	27	0.6%	0.00	0.5%	0.00	0.4%	0.00	0.5%	0.00	0.4%	0.00
	9	30	0.4%	0.00	0.3%	0.01	0.5%	0.00	0.3%	0.04	0.2%	**0.08**
	10	34	0.1%	0.01	0.1%	0.02	0.1%	**0.37**	0.1%	0.01	0.1%	**0.26**
6	5	20	0.9%	0.00	0.8%	0.00	0.7%	0.01	0.4%	0.00	0.3%	0.03
	6	24	1.1%	0.01	1.0%	0.00	0.6%	0.02	0.7%	0.02	0.3%	0.00
	7	28	0.4%	0.00	0.3%	0.00	0.0%	**0.89**	0.3%	0.05	0.2%	0.01
	8	32	0.3%	0.01	0.2%	**0.12**	0.2%	0.04	0.2%	**0.11**	0.2%	**0.08**
	9	36	0.4%	0.00	0.3%	0.01	0.2%	**0.06**	0.3%	0.00	0.3%	**0.06**
	10	40	0.4%	0.01	0.2%	**0.07**	0.1%	**0.06**	0.1%	**0.21**	0.1%	**0.49**

Note: The bold numbers represent not significantly differing at a 5% level.

The small gap values in the above two tables imply that the flexible service policy plays a dominant role in reducing the number of relocations, and the relocations that cannot be avoided by out-of-order retrievals are also less likely to be avoided by optimising the relocation stacks. From the above two tables, it can be suggested that if the truck arrival behaviour conforms to the exact CPS, for the terminals that use narrower and higher bays (e.g., S = 5, 6; T =5, 6), it is beneficial to gather the customer preference information, which can lead to a 1.3% - 3.2% reduction in the total number of relocations. If the truck arrival behaviour conforms to the heterogeneous CPS, the value of the customer preference information might be negligible. We remark that when there are more trucks per batch, the value of customer preference information may be greater. However, in practice, if there are many containers booked to the same time window, it would be wise to avoid stacking these containers in the same bay as this may increase the risk of relocation. Also, in the literature (Galle et al., 2018; Feng et al., 2020), six trucks per batch is considered as the largest batch size.

3.6 Conclusions

This paper investigates the trade-off between the number of relocations and the service fairness and assesses the value of customer information in the SCRP under flexible service policies in the import

container retrieval process. To handle the issue of service unfairness, two measures are adopted in two phases. In phase 1, each appointment time window is divided into multiple sub-time windows and the flexible service policy is only applied to smaller sub-time windows. The problem is termed as the SCRP with Multiple sub-time windows-based Flexible service policy (SCRP-MFS). In phase 2, we develop a stochastic dynamic programming model with two lexicographically ordered objective functions, in which the primary objective is to minimise the expected total number of relocations, which reflects relocation efficiency, and the secondary objective is to minimise the maximum truck turnaround time, which reflects service fairness. The model is tackled using a hierarchical iterative procedure, where the number of sub-time windows is updated iteratively and the problem at each iteration is first solved by a heuristic algorithm and then the resulted solutions are evaluated by simulation. The SCRP-MFS enables us to achieve the trade-off between the reduction of the number of relocations and the concern of service fairness by finding an appropriate number of sub-time windows that should be used in the flexible service policy.

The computational experiments demonstrate that our model is effective in reducing the maximum truck turnaround time while incurring a moderate increase in the total number of relocations and only a slight increase in the average truck turnaround time. Meanwhile, the coefficient of variation of the truck turnaround time, which is another metric of service fairness, can be reduced greatly. We also found that the service fairness is not necessarily improving as the number of sub-time windows increases. This implies that although the commonly used FCFS policy appears to be fair in terms of service sequence, it does not always ensure service fairness in terms of turnaround time. Besides, the degree of the impacts of the number of sub-time windows depends on the dimension of the bay layout. As a result of increasing the number of sub-time windows, both the reduction in the maximum truck turnaround time and the increase in the total number of relocations exhibit a decreasing trend with the increase of the stack height, while the latter also shows a decreasing trend as the bay width increases. Moreover, it is found that the scenarios of customer preference under consideration do not have a significant impact on the results. In addition, the results between utilising and not utilising the customer information are significantly different for the vast majority of the problem classes. Specifically, when the customer preference coheres with the exact scenario (i.e., each truck will arrive at only a certain sub-time window), the availability of such information can lead to a 1.3% - 3.2% reduction in the total number of relocations for the terminals that use narrower and higher bays (e.g., $S = 5, 6$; $T = 5, 6$).

The above findings help generate some managerial insights for terminal operators. Firstly, when applying the flexible service policy, the number of sub-time windows should be carefully determined to balance the number of relocations and the performance of individual trucks, in order to mitigate the issue of service unfairness. This is especially important for customer-centric terminals that may promise a threshold of truck turnaround time to their customers. Multiple sub-time windows should be applied to ensure service quality to each customer, but it should be aware that there exists a turning point after which a larger number of sub-time windows does not necessarily guarantee better service fairness. Secondly, the suggested heuristic algorithm can obtain effective solutions within several milliseconds on a desktop computer. Extensive experiments show that the algorithm performs robustly for various scenarios of customer preference. Hence, the heuristic tool is suitable for use in real-time decision makings at container yards. Thirdly, when each truck only prefers a specific sub-time window, if the yard planning system is able to gather and make use of such information from customers, the number of relocations can be further reduced. This is especially applicable for the terminals that use a narrower and higher bay layout.

As future work, apart from the maximum truck turnaround time, more measures can be proposed to describe the service fairness for the container retrieval process and considered in the multi-objective

decision-making process. Moreover, future research can focus on investigating other operational strategies on mitigating the service unfairness among trucks and improving the service quality of import container retrievals.

3.7 Appendix

Appendix A. Essential notations

A.1 Modelling notations

S: the number of stacks in a bay;
T: the number of tiers in a bay;
C: the total number of containers in a bay;
K: the total number of batches;
B_k: the set of containers in batch *k*, $k \in \{1,...,K\}$;
C_k: the number of containers in batch *k*, $k \in \{1,...,K\}$;
W: the number of sub-time windows;

$p_{i,w}^{W}$: the probability of truck *i* arriving at sub-time window *w* of its appointed time window when each appointment time window is divided into *W* sub-time windows, which represents customer preference.

ζ_k^W: the scenario of the sub-batches of stage *k*, and stage *k* refers to the *k*th batch to be retrieved;

$p(\zeta_k^W)$: the probability of ζ_k^W;

S_k: the input state of stage *k*, $k \in \{1,...,K\}$;

$\mathbf{a}_k(S_k,\zeta_k^W)$: The actions (a decision variable) taken for retrieving the *k*th batches of containers given S_k and ζ_k^W. $\mathbf{a}_k(S_k,\zeta_k^W)=\{\mathbf{a}_k^S(S_k,\zeta_k^W),\mathbf{a}_k^R(S_k,\zeta_k^W)\}$, wherein $\mathbf{a}_k^S(S_k,\zeta_k^W)$ represents the retrieval sequence for the containers in each sub-batch at stage *k* given S_k and ζ_k^W, and $\mathbf{a}_k^R(S_k,\zeta_k^W)$ represents the relocation positions that respect $\mathbf{a}_k^S(S_k,\zeta_k^W)$;

$r_k(\mathbf{a}_k|S_k,\zeta_k^W)$: The number of relocations required during action $\mathbf{a}_k$ on the bay of state S_k given ζ_k^W;

$t_k(S_k,\zeta_k^W,\mathbf{a}_k)$: The state transition function that maps S_k, ζ_k^W, and $\mathbf{a}_k$ into the next state S_{k+1};

$f_k^W(S_k)$: The expected minimum total number of relocations to retrieve the remaining *K*-*k*+1 batches of containers from state S_k when each appointment time window is divided into *W* sub-time windows.

$\tilde{a}_i$: the arrival time of truck *i* at the terminal;

$\tilde{d}_i$: the departure time of truck *i* from the yard;

$g_{k,i}^{\zeta_k^W}$: the turnaround time of truck *i*, $i \in B_k$, in batch *k* under ζ_k^W;

$o_i^{\zeta_k^W}$: the service order of truck *i*, $i \in B_k$, under ζ_k^W;

$r_i^{\zeta_k^W}$: the number of relocations needed when serving truck *i*, $i \in B_k$, under ζ_k^W;

t^{ret}: the handling time per retrieval move;

t^{rel}: the handling time per relocation move;

e_k: the end of the appointed time window of batch *k*;

s_k: the service starting time of batch *k;*

c_k: the completion time of retrieving the last container in batch *k*.

A.2 Notations in the heuristic algorithm

BIS(s): the probability of a container being blocking if relocated to stack *s*.

DIS(s): the probability of the containers with the highest priority in stack *s* being the first one to be retrieved within its batch.

t_i: the *i*th target container to be retrieved, $i \in \{1,...,C\}$.

n(*u*): the number of relocations needed for retrieving container *u*.

X: the bay configuration (state). Let X_0 represent the initial configuration.

lmin: the smallest priority label of containers in X.

Θ: the set of containers labelled *lmin* in X.

Φ_{kw}: the trucks in batch *k* that have arrived at sub-time window *w*.

c: the blocking container that is to be relocated;

$\hat{s}$: the stack where the blocking container *c* is located before relocating;

m(*s*): the smallest priority label of a container in stack *s*, $s \in \{1,...,S\}$. For an empty stack, we let *m*(*s*) equal *C*+1;

h(*s*): the number of containers in stack *s*, $s \in \{1,...,S\}$;

S_C: the set of candidate stacks;

s^*: the selected relocation stack.

Appendix B. Additional results for the impact of the customer preference scenario

Note that in the following tables, "Ho vs. He" represents homogeneous vs. heterogeneous; "Ho vs. Ex" represents homogeneous vs. exact; "He vs. Ex" represents heterogeneous vs. exact. * Significance level =5%.

Table 3. B.1 Comparisons of the total number of relocations between different customer preference scenarios for the SCRP-MFS with *W* = 3.

Problem class			Gap			p-Value		
T	*S*	*C*	Ho vs. He	Ho vs. Ex	He vs. Ex	Ho vs. He	Ho vs. Ex	He vs. Ex
3	5	10	0.16%	-1.05%	-1.21%	0.67	0.14	**0.03***
	6	12	-1.22%	-1.68%	-0.46%	**0.03***	**0.01***	0.50
	7	14	-0.52%	-0.65%	-0.13%	0.23	0.13	0.85
	8	16	-0.08%	-0.18%	-0.09%	0.92	0.87	0.94
	9	18	-0.43%	-0.12%	0.31%	0.39	0.85	0.66
	10	20	-0.03%	-0.22%	-0.19%	0.94	0.71	0.80
4	5	13	-0.64%	-2.69%	-2.07%	0.28	**0.01***	**0.02***
	6	16	-0.48%	-1.26%	-0.78%	0.35	0.13	0.44
	7	19	0.19%	-1.24%	-1.43%	0.78	0.23	0.21
	8	21	0.04%	-0.86%	-0.90%	0.93	0.32	0.35
	9	24	-0.52%	0.08%	0.60%	0.22	0.84	0.27
	10	27	0.77%	-0.79%	-1.55%	0.10	0.27	**0.02***
5	5	17	-1.06%	-3.84%	-2.81%	0.10	**0.00***	**0.00***
	6	20	-0.14%	-1.54%	-1.41%	0.80	**0.01***	**0.01***

	7	23	0.27%	-0.89%	-1.16%	0.58	0.16	0.07
	8	27	-0.26%	-0.47%	-0.21%	0.53	0.40	0.68
	9	30	-0.39%	-1.54%	-1.16%	0.24	**0.01***	**0.04***
	10	34	0.29%	-0.65%	-0.93%	0.35	0.07	**0.00***
6	5	20	-0.93%	-2.50%	-1.59%	0.14	**0.00***	**0.00***
	6	24	-1.98%	-1.64%	0.35%	**0.01***	**0.05***	0.46
	7	28	0.03%	0.21%	0.19%	0.95	0.81	0.85
	8	32	0.40%	-0.59%	-1.00%	0.37	0.25	0.07
	9	36	-0.37%	-1.26%	-0.90%	0.47	**0.04***	0.16
	10	40	0.15%	-0.17%	-0.31%	0.64	0.76	0.51

Table 3. B.2 Comparisons of the total number of relocations between different customer preference scenarios for the SCRP- MFS with $W = 4$.

Problem class			Gap			p-Value		
T	*S*	*C*	Ho vs. He	Ho vs. Ex	He vs. Ex	Ho vs. He	Ho vs. Ex	He vs. Ex
3	5	10	0.19%	-0.47%	-0.65%	0.57	0.37	0.21
	6	12	-0.74%	-1.96%	-1.23%	0.14	**0.01***	0.15
	7	14	0.01%	-0.34%	-0.35%	0.97	0.47	0.50
	8	16	0.02%	-1.66%	-1.67%	0.98	0.28	0.38
	9	18	0.14%	0.05%	-0.10%	0.83	0.93	0.91
	10	20	-0.25%	-0.49%	-0.24%	0.48	0.21	0.64
4	5	13	-1.63%	-3.32%	-1.72%	**0.03***	**0.00***	**0.02***
	6	16	-1.22%	-2.06%	-0.86%	**0.03***	**0.00***	0.18
	7	19	-0.38%	0.59%	0.98%	0.57	0.53	0.33
	8	21	0.37%	-0.25%	-0.62%	0.48	0.69	0.31
	9	24	0.83%	0.11%	-0.72%	**0.04***	0.84	0.19
	10	27	-0.15%	-0.71%	-0.56%	0.71	0.29	0.39
5	5	17	-0.88%	-4.10%	-3.25%	0.23	**0.00***	**0.00***
	6	20	-0.81%	-1.61%	-0.81%	0.17	**0.01***	0.19
	7	23	0.17%	-1.55%	-1.72%	0.69	**0.02***	**0.00***
	8	27	-0.51%	-1.53%	-1.03%	0.19	**0.02***	0.10
	9	30	-0.98%	-1.34%	-0.37%	**0.01***	**0.04***	0.52
	10	34	0.17%	-0.26%	-0.43%	0.59	0.45	0.13
6	5	20	0.27%	-1.66%	-1.92%	0.61	**0.02***	**0.01***
	6	24	-0.99%	-1.99%	-1.01%	0.07	**0.01***	0.07
	7	28	-1.09%	-1.09%	0.00%	**0.04***	0.26	1.00
	8	32	-0.66%	0.09%	0.76%	0.10	0.89	0.15
	9	36	-0.23%	-0.96%	-0.73%	0.56	**0.03***	0.10
	10	40	0.27%	0.22%	-0.05%	0.45	0.71	0.91

Table 3. B.3 Comparisons of the total number of relocations between different customer preference scenarios for the SCRP- MFS with $W = 5$.

Problem class			Gap			p-Value		
T	*S*	*C*	Ho vs. He	Ho vs. Ex	He vs. Ex	Ho vs. He	Ho vs. Ex	He vs. Ex
3	5	10	0.70%	-0.69%	-1.38%	0.07	0.22	**0.01***
	6	12	-0.63%	-2.45%	-1.84%	0.29	**0.00***	**0.01***
	7	14	-0.14%	-1.09%	-0.95%	0.77	**0.06***	0.21
	8	16	-0.20%	-2.40%	-2.21%	0.60	**0.02***	0.07
	9	18	-0.13%	-0.42%	-0.29%	0.82	0.53	0.72
	10	20	-0.90%	-0.91%	-0.01%	**0.01***	**0.01***	0.98
4	5	13	-0.53%	-1.62%	-1.10%	0.56	0.13	0.22
	6	16	-0.79%	-1.76%	-0.97%	0.14	**0.03***	0.27
	7	19	0.08%	-1.32%	-1.39%	0.92	0.31	0.28
	8	21	-0.11%	-0.02%	0.09%	0.84	0.98	0.93
	9	24	0.33%	0.22%	-0.10%	0.51	0.67	0.84
	10	27	0.39%	-0.42%	-0.81%	0.46	0.58	0.33
5	5	17	-1.15%	-4.01%	-2.90%	0.13	**0.00***	**0.00***
	6	20	-0.33%	-1.86%	-1.53%	0.61	**0.01***	**0.00***
	7	23	0.75%	-0.20%	-0.94%	0.14	0.80	0.22
	8	27	0.34%	-1.32%	-1.66%	0.42	**0.03***	**0.00***
	9	30	-0.36%	-1.00%	-0.64%	0.28	0.10	0.24
	10	34	-0.11%	-0.41%	-0.31%	0.79	0.30	0.50
6	5	20	-0.81%	-2.83%	-2.03%	0.11	**0.00***	**0.00***
	6	24	-0.54%	-1.68%	-1.14%	0.28	**0.03***	**0.05***
	7	28	-0.45%	-2.62%	-2.18%	0.27	**0.02***	**0.03***
	8	32	0.01%	-0.07%	-0.09%	0.98	0.91	0.85
	9	36	-0.13%	-0.74%	-0.61%	0.72	0.15	0.13
	10	40	-0.02%	-0.04%	-0.02%	0.96	0.93	0.97

Table 3. B.4 Comparisons of the total number of relocations between different customer preference scenarios for the SCRP- MFS with $W = 6$.

Problem class			Gap			p-Value		
T	*S*	*C*	Ho vs. He	Ho vs. Ex	He vs. Ex	Ho vs. He	Ho vs. Ex	He vs. Ex
3	5	10	0.41%	-0.46%	-0.86%	0.26	0.28	**0.01***
	6	12	-1.07%	-2.23%	-1.17%	**0.02***	**0.00***	**0.04***
	7	14	-0.43%	-1.44%	-1.02%	0.42	**0.01***	0.17
	8	16	-0.22%	-1.58%	-1.36%	0.71	0.10	0.23
	9	18	0.01%	0.05%	0.04%	0.98	0.92	0.95
	10	20	-0.07%	-0.55%	-0.48%	0.84	0.26	0.40
4	5	13	-0.53%	-1.61%	-1.08%	0.49	0.16	0.27
	6	16	-0.26%	-1.71%	-1.45%	0.60	0.06	**0.05***
	7	19	-0.14%	-1.22%	-1.08%	0.79	0.34	0.35
	8	21	-0.03%	-0.92%	-0.89%	0.94	0.18	0.21
	9	24	-0.30%	0.18%	0.48%	0.42	0.63	0.35
	10	27	0.08%	-0.83%	-0.91%	0.86	0.20	0.27
5	5	17	-0.71%	-2.38%	-1.68%	0.27	**0.02***	0.06
	6	20	-0.19%	-1.49%	-1.30%	0.74	**0.01***	**0.05***
	7	23	0.28%	-1.59%	-1.86%	0.50	**0.03***	**0.01***
	8	27	-0.42%	-1.04%	-0.62%	0.27	0.07	0.33
	9	30	-0.23%	-1.41%	-1.19%	0.50	**0.03***	**0.02***
	10	34	0.71%	-0.39%	-1.09%	0.06	0.37	**0.02***
6	5	20	0.35%	-1.97%	-2.31%	0.54	**0.00***	**0.00***
	6	24	-0.38%	-1.83%	-1.45%	0.46	**0.00***	**0.02***
	7	28	-0.38%	-0.84%	-0.47%	0.29	0.18	0.44
	8	32	-0.09%	-0.53%	-0.45%	0.84	0.36	0.40
	9	36	-0.53%	-1.26%	-0.73%	0.24	**0.01***	0.08
	10	40	0.21%	-0.28%	-0.49%	0.62	0.58	0.27

References

Azab, A., & Morita, H. (2021). The block relocation problem with appointment scheduling. European Journal of Operational Research. https://doi.org/10.1016/j.ejor.2021.06.007

Bacci, T., Mattia, S., & Ventura, P. (2019). The bounded beam search algorithm for the block relocation problem. Computers & Operations Research, 103, 252-264.

Bacci, T., Mattia, S., & Ventura, P. (2020). A branch-and-cut algorithm for the restricted Block Relocation Problem. European Journal of Operational Research, 287(2), 452-459.

Bertsimas, D., Farias, V. F., & Trichakis, N. (2011). The price of fairness. Operations research, 59(1), 17-31.

Bertsimas, D., Farias, V. F., & Trichakis, N. (2013). Fairness, efficiency, and flexibility in organ allocation for kidney transplantation. Operations Research, 61(1), 73-87.

Borjian, S., Manshadi, V., Barnhart, C., & Jaillet, P. (2013). dynamic stochastic optimization of reshuffles in container terminals. In Manufacturing and Service Operations Management (MSOM) conference.

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). Managing relocation and delay in container terminals with flexible service policies. arXiv preprint 1503.01535.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. European journal of operational research, 235(2), 412-430.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. European Journal of Operational Research, 219(1), 96-104.

Caserta, M., Schwarze, S., & Voß, S. (2020). Container rehandling at maritime container terminals: A literature update. In J. W. Böse (Ed.), *Handbook of Terminal Planning (2nd ed.)*, Operations Research/Computer Science Interfaces Series 64 (pp. 343-382). Springer, Cham.

Chen, G., Govindan, K., & Golias, M. M. (2013a). Reducing truck emissions at container terminals in a low carbon economy: Proposal of a queueing-based bi-objective model for optimizing truck arrival pattern. Transportation Research Part E: Logistics and Transportation Review, 55, 3-22.

Chen, G., Govindan, K., Yang, Z. Z., Choi, T. M., & Jiang, L. (2013b). Terminal appointment system design by non-stationary M(t)/Ek/c(t) queueing model and genetic algorithm. International Journal of Production Economics, 146(2), 694-703.

Covic, F. (2017). Re-marshalling in automated container yards with terminal appointment systems. Flexible Services and Manufacturing Journal, 29(3), 433-503.

de Melo da Silva, M., Erdoğan, G., Battarra, M., & Strusevich, V. (2018). The block retrieval problem. European Journal of Operational Research, 265(3), 931-950.

Expósito-Izquierdo, C., Melián-Batista, B., & Moreno-Vega, J. M. (2015). An exact approach for the blocks relocation problem. Expert Systems with Applications, 42(17-18), 6408-6422.

Feng, Y., Song, D. P., Li, D., & Zeng, Q. (2020). The stochastic container relocation problem with flexible service policies. Transportation Research Part B: Methodological, 141, 116-163.

Galle, V., Manshadi, V. H., Boroujeni, S. B., Barnhart, C., & Jaillet, P. (2018). The stochastic container relocation problem. Transportation Science, 52(5), 1035-1058.

Gharehgozli, A., & Zaerpour, N. (2018). Stacking outbound barge containers in an automated deep-sea terminal. European Journal of Operational Research, 267(3), 977-995.

Hakan Akyüz, M., & Lee, C. Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Research Logistics (NRL), 61(2), 101-118.

Jang, D. W., Kim, S. W., & Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations. International Journal of Production Economics, 143(2), 256-262.

Jia, S. and Meng, Q. (2021). Equitable vessel traffic scheduling in a seaport. Transportation Science, Available at SSRN: http://dx.doi.org/10.2139/ssrn.3808857.

Jiang, T., Zeng, B., Wang, Y., & Yan, W. (2021, April). A new heuristic reinforcement learning for container relocation problem. In Journal of Physics: Conference Series (Vol. 1873, No. 1, p. 012050). IOP Publishing.

Jin, B., Zhu, W., & Lim, A. (2015). Solving the container relocation problem by an improved greedy look-ahead heuristic. European Journal of Operational Research, 240(3), 837-847.

Jovanovic, R., & Voß, S. (2014). A chain heuristic for the blocks relocation problem. Computers & Industrial Engineering, 75, 79-86.

Kang, J., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. Journal of Intelligent Manufacturing, 17(4), 399-410.

Kim, K. H., & Hong, G. P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research, 33(4), 940-954.

Kim, K. H., & Yi, S. (2021). Utilizing information sources to reduce relocation of inbound containers. Maritime Economics & Logistics, 1-24.

Knatz, G. (2017). How competition is driving change in port governance, strategic decision-making and government policy in the United States. Research in Transportation Business & Management, 22, 67-77.

Ku, D., & Arthanari, T. S. (2016a). Container relocation problem with time windows for container departure. European Journal of Operational Research, 252(3), 1031-1039.

Ku, D., & Arthanari, T. S. (2016b). On the abstraction method for the container relocation problem. Computers & Operations Research, 68, 110-122.

Law, A.M., & Kelton, W.D. (2000). Simulation Modeling and Analysis (3rd ed.). Boston: McGraw-Hill, pp. 511-515.

Li, D., Zhang, T., Dong, X., Yin, Y., & Cao, J. (2019). Trade-off between efficiency and fairness in timetabling on a single urban rail transit line under time-dependent demand condition. Transportmetrica B: Transport Dynamics.

Maglić, L., Gulić, M., & Maglić, L. (2020). Optimization of container relocation operations in port container terminals. Transport, 35(1), 37-47.

Matl, P., Hartl, R. F., & Vidal, T. (2018). Workload equity in vehicle routing problems: A survey and analysis. Transportation Science, 52(2), 239-260.

Notteboom, T., Pallis, T., & Rodrigue, J. P. (2021). Disruptions and resilience in global container shipping and ports: the COVID-19 pandemic versus the 2008–2009 financial crisis. Maritime Economics & Logistics, 23(2), 179-210.

Parreño-Torres, C., Alvarez-Valdes, R., Ruiz, R., & Tierney, K. (2020). Minimizing crane times in pre-marshalling problems. Transportation Research Part E: Logistics and Transportation Review, 137, 101917.

Petering, M. E., & Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. European Journal of Operational Research, 231(1), 120-130.

Phan, M. H., & Kim, K. H. (2015). Negotiating truck arrival times among trucking companies and a container terminal. Transportation Research Part E: Logistics and Transportation Review, 75, 132-144.

Qi, J. (2017). Mitigating delays and unfairness in appointment systems. Management Science, 63(2), 566-583.

Quispe, K. E. Y., Lintzmayer, C. N., & Xavier, E. C. (2018). An exact algorithm for the blocks relocation problem with new lower bounds. Computers & Operations Research, 99, 206-217.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: a literature update. OR spectrum, 30(1), 1-52.

Tanaka, S., & Takii, K. (2015). A faster branch-and-bound algorithm for the block relocation problem. IEEE Transactions on Automation Science and Engineering, 13(1), 181-190.

Tanaka, S., & Mizuno, F. (2018). An exact algorithm for the unrestricted block relocation problem. Computers & Operations Research, 95, 12-31.

Tang, L., Jiang, W., Liu, J., & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions, 47(7), 751-766.

Teye, C., & Bell, M. (2016). Dynamic discrete berth allocation in container terminals under four performance measures. The University of Sydney, working paper.

Ting, C. J., & Wu, K. C. (2017). Optimizing container relocation operations at container yards with beam search. Transportation Research Part E: Logistics and Transportation Review, 103, 17-31.

Tong, X., Woo, Y. J., Jang, D. W., & Kim, K. H. (2015). Heuristic rules based on a probabilistic model and a genetic algorithm for relocating inbound containers with uncertain pickup times. International Journal of Industrial Engineering, 22, 93–101.

van Asperen, E., Borgman, B., & Dekker, R. (2013). Evaluating impact of truck announcements on container stacking efficiency. Flexible Services and Manufacturing Journal, 25(4), 543-556.

Wan, Y. W., Liu, J., & Tsai, P. C. (2009). The assignment of storage locations to containers for a container stack. Naval Research Logistics (NRL), 56(8), 699-713.

Wang, Y., Xiao, Z., Huang, Y., Hao, Y., & Gu, T. (2017). Study of continuous berth allocation algorithm based on fairness maximization. Journal of Engineering Science & Technology Review, 10(5), 116-127.

Wu, Y., Yang, H., Zhao, S., & Shang, P. (2021). Mitigating unfairness in urban rail transit operation: A mixed-integer linear programming approach. Transportation Research Part B: Methodological, 149, 418-442.

Xie, Y., & Song, D. P. (2018). Optimal planning for container prestaging, discharging, and loading processes at seaport rail terminals with uncertainty. Transportation Research Part E: Logistics and Transportation Review, 119, 88-109.

Yang, M., Allen, T. T., Fry, M. J., & Kelton, W. D. (2013). The call for equity: simulation optimization models to minimize the range of waiting times. IIE Transactions, 45(7), 781-795.

Zehendner, E., & Feillet, D. (2014). A branch and price approach for the container relocation problem. International Journal of Production Research, 52(24), 7159-7176.

Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. European Journal of Operational Research, 245(2), 415-422.

Zehendner, E., Feillet, D., & Jaillet, P. (2017). An algorithm with performance guarantee for the online container relocation problem. European Journal of Operational Research, 259(1), 48-62.

Zeng, Q., Feng, Y., & Yang, Z. (2019). Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering, 127, 366-382.

Zhang, C., Guan, H., Yuan, Y., Chen, W., & Wu, T. (2020). Machine learning-driven algorithms for the container relocation problem. Transportation Research Part B: Methodological, 139, 102-131.

Zhang, X., Chen, X., Ji, M., & Yao, S. (2017). Vessel scheduling model of a one-way port channel. Journal of Waterway, Port, Coastal, and Ocean Engineering, 143(5), 04017009.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. Transportation Research Part E: Logistics and Transportation Review, 46(3), 327-343.

Zhou, C., Wang, W., & Li, H. (2020). Container reshuffling considered space allocation problem in container terminals. Transportation Research Part E: Logistics and Transportation Review, 136, 101869.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. IEEE Transactions on Automation Science and Engineering, 9(4), 710-722.

Zuidwijk, R. A., & Veenstra, A. W. (2015). The value of information in container transport. Transportation Science, 49(3), 675-685.

Chapter 4

Smart stacking for import containers using customer information at automated container terminals

Abstract: Container stacking strategies affect the number of unproductive relocations and the truck waiting times during the container retrieval process, which plays a vital role in the efficiency of container terminals. Container relocation is caused by the mismatch between containers' stacking positions and their retrieval sequences. Motivated by the practical import free-flow program that aims to expedite the container retrieval flow by eliminating container relocations, we conceptualise a new container stacking strategy, termed Smart Stacking (SS) strategy, which can greatly reduce the number of relations for import containers by utilising customer information. Some customers' containers are allocated to dedicated stacks (smart stacks) so that no relocations are required for them during the retrieval process, while other containers will share non-smart stacks. The problem is to determine the smart customers or containers, the number and locations of smart stacks (and non-smart stacks), and assign a batch of import containers to exact stacking positions in a yard block at an automated container terminal. The objective is to minimise the total retrieval time that is the sum of the relocation time and the crane travel time. Our problem can be regarded as the Storage Location Assignment Problem (SLAP) under the SS strategy. Two variants of SLAP are investigated under two SS policies, the non-split policy and the split policy, according to whether the containers from the same customer are allowed to be split between smart stacks and non-smart stacks. For the non-split variant, a mixed-integer programming (MIP) model is formulated first. By analysing the properties of the optimal solution, an improved formulation is then proposed, which leads to enhanced computational performance. To further improve the computational efficiency, a divide-and-conquer heuristic based on the structure of the model is proposed to solve the non-split variant, which can produce high-quality solutions with a gap of less than 0.6% from the optimal ones within a few seconds. For the split variant, we develop a MIP model under the optimal partitions from the non-split model. We prove that the split variant yields better performance than the non-split variant. Extensive experiments are carried out to illustrate the effectiveness of smart stacking including the comparison with the random stacking strategy that is commonly used in practice. It is found that customer information and yard utilisation rate have a great influence on the effectiveness of smart stacking. Significant benefits may be achieved by allowing splitting under certain conditions. [1]

Keywords: OR in maritime industry, import container stacking problem, smart stacking strategy, value of customer information

[1] Feng, Y., Song, D. P., & Li, D. (2021). Smart stacking for import containers using customer information at automated container terminals, European Journal of Operational Research. In Press. https://doi.org/10.1016/j.ejor.2021.10.044.

4.1 Introduction

With more than 80 percent of the world merchandise trade by volume being carried by sea (UNCTAD, 2020), maritime transport and ports have become essential components in global supply chains. Within the maritime transport industry (including tanker, dry bulk, container, and general cargo), 52 percent of cargoes by value were carried by container ships (Lee and Song, 2017). The global containerized trade has experienced a 5.8 percent average annual growth over the past two decades (UNCTAD, 2019). Maritime container terminals, where containers are transferred between seaborn transport and hinterland transport, provide crucial linkages in the global container shipping network, and their handling productivities and efficiencies are essential to ensure efficient supply chain operations. According to Drewry baseline forecast, although world port container throughput is expected to contract by 7.3 percent in 2020 due to the impact of the COVID-19 pandemic, a jump up to more than 10 percent is projected in 2021 (UNCTAD, 2020). In fact, major UK container ports (Felixstowe and Southampton) and US west coastal ports (Los Angeles and Long Beach) have experienced severe congestion in the second half of 2020. Therefore, container terminals need to prepare for a potential surge in container handling volumes.

A container terminal can be divided into three main areas: seaside, landside, and storage yard (de Melo da Silva, 2018). The storage yard serves as the buffer area for storing containers before their onward transportation and links the seaside and landside operations. Import containers are discharged from ships at the seaside, unloaded into the storage yard, and then loaded to external trucks or trains at the landside, whereas the export containers follow the reverse path (Kizilay and Eliiyi, 2020). Inefficient container unloading and loading operations at the yard lead to longer waiting times for external trucks, which is considered as the major source of delay that affects the truck turnaround time (Phan and Kim, 2016). The turnaround time of external trucks is one of the key performance measures of the efficiency of container terminals and also contributes to the evaluation of customer service levels and port competitiveness (de Melo da Silva, 2018). Longer turnaround time results in truck congestion at the container terminals, which has caused a serious environmental concern because of the emissions it generates (Phan and Kim, 2016). Terminal operators have been under enormous pressure from different stakeholders who require terminals to reduce the truck turnaround time, including governments (Giuliano and O'Brien, 2007), port authorities (e.g., Port Botany and Port Metro Vancouver (The Port of Long Beach and Port of Los Angeles, 2017)), and the stakeholders of the hinterland transport (Bonney, 2015). This paper focuses on improving the delivery (loading) efficiency of import containers to external trucks at the interface of the storage yard and the terminal's landside.

Container stacking, the theme of this paper, addresses the assignment of storage positions in the yard to containers, which directly affects the container delivery efficiency and the truck waiting time. A major source of inefficiency when retrieving containers from yards is container relocations (Ku and Arthanari, 2016a). Due to limited space in the yard, containers are piled up vertically in stacks (usually up to six tiers high). If a target container to be retrieved is not on the topmost tier, those above it – that is, the blocking containers - need to be moved out of the way in order to access the target one. Such moves of blocking containers are called relocation, reshuffling, or rehandling. Relocation is an unproductive operation that is costly to terminals and also contributes to truck waiting times. Researchers have been trying to reduce relocations by addressing a series of container stacking related problems, for example, the container relocation problem that determines the positions of relocated containers (e.g., Zhu et al., 2012; Zehendner et al., 2015), the container pre-marshalling problem that re-arranges the container stacking positions (e.g., Parreño-Torres et al., 2019; Tanaka et al., 2019), and the container stacking problem that pre-plans the initial stacking positions of containers (e.g., Zhang et al., 2014; Gharehgozli et al., 2014). In this paper, we

address one of these container stacking problems - the Storage Location Assignment Problem (SLAP), where a batch of import containers are allocated to exact locations in a storage area to minimise their future retrieval times.

The SLAPs for import containers have been under-studied. The main challenge may lie in the uncertainty regarding which container will be retrieved first since external trucks arrive at the terminal randomly to pick up a specific container (Saur í and Mart ń, 2011; Yu and Qi, 2013). A couple of studies attempt to reduce the number of relocations by stacking containers based on the information of retrieval times (e.g., Lee et al., 2008; Maldonado et al., 2019). However, in reality, in most cases, the retrieval times of containers are not yet known when stacking the containers, and thus containers are often randomly stacked, which can lead to a high relocation rate. Studies have shown that in busy ports such as Los Angeles-Long Beach, it takes on average two to three relocations to deliver one container to a truck (Mongelluzzo, 2015a).

With the development of port digitalization and in an effort to improve container delivery efficiency and reduce truck waiting times, an innovative container delivery and staging program — Import Free Flow (IFF) — has been initiated at Port of Los Angeles (Mongelluzzo, 2015b). The idea behind IFF is to eliminate the need of relocations and realise rapid retrieval flow through pre-staging large groups of containers being picked up by the same customer. With IFF, high-volume customers can have all their containers stored in dedicated stacks when they are unloaded from a vessel. These containers are called free-flow containers. Customers can pick up their free-flow containers from the top of the stacks on a last-in-first-out basis since the pickup sequences of containers from the same customer do not matter. As a result, no relocations are needed for retrieving free-flow containers.

The IFF program has resulted in significant improvements in truck turnaround times in practice. For example, free-flow containers reduced truck turnaround times by more than 50 percent at Port of Los Angeles in 2015 (Parker, 2015). However, the IFF program has not been widely adopted in practice, and academic research in this regard is rather scarce. In the current practice, terminal operators usually do not utilise the customer information of containers when stacking the containers either because the information is not available to them, or probably more importantly, because they have not recognised the value of the customer information and how to use the information to determine the container stacking positions. In addition, the current stacking strategy in the IFF program is rather heuristic and is inadequate for its mass application. For example, the free-flow service is only available to high-volume customers who own at least 50 containers, and the free-flow containers are simply pre-staged in specific stacks that are separated from the traditional containers (i.e., non-free-flow containers) (Dupin, 2015; Parker, 2015). Expanding the free-flow service to smaller customers could dramatically improve container retrieval efficiency, which is the next goal of the practitioners (Parker, 2015). However, which customers or containers should be free-flowed, how many free-flow stacks should be selected, and where these stacks should be located in the yard have never been examined.

Motivated by the IFF program, we conceptualised a new import container stacking strategy - Smart Stacking (SS) strategy - where import containers are grouped based on customer information, and they are classified into either smart (free-flow) or non-smart (non-free-flow) containers to be allocated to smart stacks and non-smart stacks respectively in a yard block in an optimal way. The smart containers of a customer will not share the stacks with containers from any other customers to guarantee zero relocation for them. The non-smart containers will share the stacks with other customers' containers as usual, which may incur relocations during the future retrieval process. This paper aims to investigate how customer information can be utilised to better plan the exact stacking positions for import containers so as to improve

retrieval efficiency. The research objectives are: i) to seek the optimal solution for stacking a batch of import containers with customer information into a limited (user-defined) storage area in a yard block under the proposed SS strategy; ii) to quantify the reduction in the total retrieval time by applying the SS strategy; iii) and to evaluate the impacts of relevant parameters (customer information and yard utilisation rate) on the effectiveness of the SS strategy.

Our contributions to the existing literature and practice can be summarized as follows: (i) We propose a new stacking strategy – Smart Stacking (SS) strategy - to improve the import container retrieval efficiency at container terminals. (ii) We introduce two forms of stacking policies under the framework of the SS strategy, depending on whether the containers from the same customer are allowed to be split between smart stacks and non-smart stacks or not. Correspondingly, we develop two variants of mathematical models for the Storage Location Assignment Problem (SLAP) at an Automated Container Terminal (ACT), that is, the non-split model and the split model. The proposed models enable terminal operators to determine which customers or containers should be selected to be free-flowed and to quantify the benefits of the splitting policy. (iii) We establish structural properties of the optimal solution to the non-split model and then make use of these properties to improve the computational efficiency of the non-split model. (iv) To overcome the computational complexity and enable the SS strategy to be applicable in real-time situations, we develop a heuristic algorithm to solve the non-split variant (which is the focus of this paper) based on the structure of the model. The heuristic algorithm can obtain near-optimal solutions in several seconds. (v) We conduct extensive experiments to demonstrate the effectiveness of the SS strategy, and the impact of the customer information and the yard utilisation rate on the results. The findings can help terminal operators to understand the effectiveness of the SS strategy under a variety of scenarios and assess the value of customer information to container retrieval efficiency, which could promote the vertical collaboration between terminal operators, trucking companies, and cargo owners to improve supply chain performance.

The remainder of the paper is organized as follows. In Section 4.2, we review existing stacking strategies, discuss the previous work related to the container stacking problems, and summarize the research gap. Section 4.3 describes the problem under consideration and presents two forms of smart stacking policies. Section 4.4 formulates the SLAPs under the two policies by using mixed-integer programming and analyses the structural properties of the optimal solution. Section 4.5 proposes a heuristic algorithm for the non-split model. In section 4.6, we conduct computational experiments to illustrate the effectiveness of the proposed strategies and generate managerial insights. Section 4.7 concludes the paper, discusses several extensions of this study, and envisages further research directions.

4.2 Literature review

Container stacking has attracted extensive attention over the last two decades (see reviews from Zhen et al., 2013; Carlo et al., 2014a, b; Lehnfeld and Knust, 2014). The problems related to container stacking include container stacking strategies, storage space allocation, storage location assignment, container relocation, and container pre-marshalling. This paper focuses on the short-term operational decision to assign import containers to exact storage locations, which belongs to the storage location assignment problem (SLAP). Relevant literature is organized into three topics: container stacking strategies, storage location assignment, and container relocation estimation.

4.2.1 Container stacking strategies

Container stacking strategies are a set of stacking rules or criteria that should be adhered to when determining the storage position of each container or the storage space of a group of containers. Container stacking strategies are tactical level decisions of container terminals (Maldonado et al., 2019), which influence the allocation of stacking positions at the operational level. Several types of stacking strategies have been applied in practice and studied in the literature, which usually differ between export containers and import containers due to their different arrival and departure characteristics.

Export containers

Export containers usually arrive at terminals individually and are loaded onto vessels in batch. For export containers, their arrival times are uncertain but their departure times are relatively fixed by the destination vessels. There are several types of stacking strategies for export containers, which differ by the type of information utilised to categorise containers. In the residence time stacking strategy, the containers' residence/departure times are used to determine whether a container can be stacked on top of others. Container relocation can be reduced by stacking the earlier-departure containers on top of the later-departure containers (e.g., Borgman et al., 2010). In the category stacking strategy, containers are categorised according to the containers' attributes, such as the weight class, the port of destination, and the type of container, in which the containers of the same category can be stacked on top of each other to avoid relocation (e.g., Dekker et al., 2006; Kim et al., 2000; Kang et al., 2006; Guerra-Olivares et al., 2018). This is because the containers of the same category are exchangeable in the ship loading plan that specifies only the container group. When a container from a group is requested for loading into a ship slot in the loading plan that specifies the group, any container from the same group can be loaded into the slot (Kim et al., 2000). For the category stacking, depending on whether containers to different ships are allowed to share a stack, dedicated stacking strategies and shared stacking strategies are differentiated. Under the dedicated stacking strategy, containers to different ships cannot stack on top of each other. On the other hand, the shared stacking strategy allows containers to different ships to be stacked on top of each other by considering the departure time of containers, which can better utilise the yard space (e.g., Gharehgozli et al., 2014; Gharehgozli and Zaerpour, 2018).

Import containers

Import containers are unloaded from vessels in large volumes and then picked up by customers individually and randomly. Due to the high uncertainty in the retrieval sequences of import containers, it is difficult to categorise import containers according to their departure times. Two types of stacking strategies for import containers are commonly used in practice and have been investigated in academic research: segregation strategy and non-segregation strategy. Under the segregation strategy, containers from different ships are stacked separately in the container yard. Under the non-segregation strategy, containers from different ships are mixed in the storage area such that newly discharged containers are stacked on top of old ones. The segregation strategy may have the advantage of reducing the number of relocations during the container retrieval process because earlier-arrived containers are likely to be retrieved earlier; but it requires additional clearing moves before each ship's arrival to create enough space for the new containers. On the other hand, the non-segregation strategy would increase relocation moves because the containers that have stayed for a longer time and thus tend to be picked up soon will be buried under recently arrived ones (De Castilho and Daganzo, 1993).

The segregation strategy and non-segregation strategy are first investigated by De Castilho and Daganzo

(1993) and then are further developed by Saur í and Mart ń (2011). Mathematical models have been developed to optimise the stacking height under the segregation strategy (Kim and Kim, 1999) and to optimise the number of import containers allocated to each bay under both segregation and non-segregation strategies (Yu and Qi, 2013). The segregation strategy separates import containers roughly by the arriving vessels but does not specify how the containers from the same vessel are stacked. A few studies develop more detailed stacking strategies based on the container departure dates (e.g., Guldogan, 2011) and the estimated dwell times of import containers (e.g., Lee et al., 2008; Gaete et al., 2017; Maldonado et al., 2019) where containers with longer dwell times are stored under those with smaller values to reduce the number of future relocations. However, in reality, the departure dates are often not available in advance, and accurate prediction of dwell times is difficult. Besides, the containers with the same dwell time will be picked up randomly, which still incur relocations. Different from the above studies, in this paper, the smart stacking strategy uses the customer information of import containers to create relocation-free stacks. This can avoid the difficulty to predict the container departure time.

4.2.2 Storage location assignment problem

The determination of container storage locations is usually addressed hierarchically in two decision problems: the Storage Space Allocation Problem (SSAP) and the Storage Location Assignment Problem (SLAP) (Kim and Park, 2003; Zhang et al., 2003). The SSAP determines the amount of yard storage space allocated to each vessel for their containers, which can be addressed at various levels according to the storage space unit considered: yard section, yard block, yard sub-block, and yard bay (Jin et al., 2016). The SSAPs mainly aim to improve the efficiency of the container stacking process with efficient use of the terminal resources (e.g., Zhang et al., 2003; Lee et al., 2007; Zhen, 2016; Jiang et al., 2012, 2013; Zhou et al., 2020). The SLAP deals with the assignment of individual containers to exact storage locations – which is specified by a bay number, a row number, and a tier - in blocks. The number of relocations during the future retrieval process is an important performance measure in the SLAPs when the container retrieval efficiency is the focus (e.g., Kim et al., 2000; Zhang et al., 2010; Saur íand Mart ń, 2011; Zhu et al., 2020). This paper falls into the SLAP, in which we determine the exact storage location of each import container in a given storage area of a block.

The SLAP may be classified into two broad categories according to the planning approaches: online planning and offline planning. The online planning approach allocates containers to slots in a real-time way by considering the dynamic characteristics of the problem and the uncertain information on containers. For example, online-rule-based heuristics are used to determine the stacking position of each container separately in real-time (e.g, Park et al., 2011; Lin et al., 2017; Petering et al., 2017; He et al., 2019). Simulation-based methods have been used to evaluate and optimise the performance measures (e.g., Dekker et al., 2006; Borgman et al., 2010; Guldogan, 2011). Moreover, facing the uncertainties and disturbances in the container stacking environment (e.g., equipment breakdown, breakage of machines, and a fault in a container placing), decentralized approaches such as case based reasoning (Rekik et al., 2018) and multi-agent approach (Rekik and Elkosantini, 2019) are developed for the reactive container stacking systems.

The offline planning approach focuses on finding an optimal plan at the beginning of the planning period for an offline environment where the input data of the defined problems are known. This paper follows this research stream. The relevant literature adopting the offline planning approach is reviewed in the following two sub-sections according to whether they deal with export containers or import containers.

4.2.2.1 Export containers

The loading sequence of export containers can be pre-determined to some extent by certain criteria, which makes the optimisation problem relatively well defined. Preston and Kozan (2001) assume that the containers' loading sequence is pre-determined by various loading schedules and develop a mixed integer programming model to minimise the total transfer time of each yard machine that is the sum of the travel time and the relocation time. Some studies assume that heavy containers will be loaded onto vessels before light ones and thus position heavier ones on top of lighter ones to reduce the number of relocations. Based on the (uncertain) weight class information of containers, dynamic programming models (Kim et al., 2000; Zhang et al., 2010) and simulated annealing algorithm (Kang et al., 2006) are proposed to minimise the expected number of relocations or the total punishment of relocations (Zhang et al., 2014). Gharehgozli and Zaerpour (2018) assume that the outbound barge containers' departure priorities are pre-determined jointly by the containers' attributes and the barges' arrival time windows. Containers with lower departure priorities are strictly not allowed to be stacked on top of those with higher priorities so that relocation is completely avoided. An integer programming model is proposed to determine the storage locations to minimise the total retrieval time. Moreover, a couple of papers decompose the SLAP into two stages, in which the bays are assigned to containers in the first stage and the slot for storing each container is determined in the second stage where the number of relocations is minimised (Chen and Lu, 2012; Li et al., 2017).

4.2.2.2 Import containers

Relatively fewer studies have been conducted on the SLAP for import containers. One of the challenging issues for import containers is the phenomenon of uncertainties in the retrieval sequence. Assuming that containers are retrieved with a certain probability distribution, Kim and Kim (1999) develop mathematical models to determine the optimal average stacking height to minimise the expected total number of relocations over a planning horizon. Assuming that import containers are arriving with constant rates and are retrieved with different probability of departure time, Saur ían d Mart ín (2011) propose a probabilistic distribution-based mathematical model to estimate the number of relocations for the whole block under specific stacking strategies. The stacking strategies define the rules of mixing different groups of containers and clearing containers to reduce unproductive moves. A few studies assume that the exact retrieval time/sequence is known so that the number of relocations can be measured exactly (Chang and Zhu, 2019; Wang et al., 2020) or relocations can be completely avoided (Razouk et al., 2016). Under this assumption, Chang and Zhu (2019) develop a two-stage model for the storage space assignment of inbound containers in rail–water intermodal container terminals, which selects the optimal block for containers to balance the workload in different blocks at the first stage and assigns containers into the optimal slots to reduce the amount of overlapping (number of relocations) at the second stage. Besides, Wang et al. (2020) develop a multi-objective optimisation model to minimise container overlapping amounts and crane moving distance for stacking both inbound and outbound containers in a rail-truck transhipment terminal. In addition, Razouk et al. (2016) develop a MIP model for the slot assignment of inbound containers, where the travelling distance between the berth and the storage bay is minimised and relocations are avoided. By assuming group retrieval priorities given by the truck arrival time windows, Zhu et al. (2020) combine the container stacking problem with the ship unloading problem - the inbound containers unloading and stacking problem (ICUSP) - to optimise both the container unloading sequence from the vessel and the container storage locations in the yard with the objective of minimising the expected number of relocations during the container retrieval process.

We remark that, in reality, due to the dynamic and random arrivals of external trucks, the exact retrieval sequence of import containers is difficult to know when containers are stacked. For terminals that are equipped with a truck appointment system, relative retrieval priorities may be obtained from the truck appointment information, but only after the containers have been stacked in the yard. This is because the fact that, in most cases, appointments are not bookable until the container has already been customs cleared for pick-up, which can be days after the container have been stacked in the terminal (e.g., DP World London Gateway; The Port of Long Beach and Port of Los Angeles, 2017). In this study, we do not assume the information about the container retrieval sequence.

4.2.2.3 Concepts and practice relevant to smart stacking

The smart stacking strategy proposed in this paper utilises the customer information to group containers so as to reduce relocations, which resembles the category-based stacking or grouped-storage. Similar concepts and practice can also be identified in the storage systems of other relevant industries, such as containership stowage systems (Monaco et al., 2014; Iris et al., 2018), warehousing systems (Zaerpour et al., 2015) and generic block stacking systems (Yang and Kim, 2006; Jang et. al, 2013). However, there is a fundamental difference between our smart stacking strategy and the existing category-based stacking. The category-based stacking relies on simple criteria (e.g. container attributes, departure time, and destinations) to categorise containers into groups, which is treated as pre-determined before stacking. It does not differentiate between smart and non-smart groups/containers because each stack is allowed to store multiple groups of containers. On the other hand, given the customer information, the smart stacking strategy incorporates intelligence into the stacking decision-making by simultaneously determining whether the groups/containers to be either smart or non-smart and optimising the locations of these smart stacks and smart containers.

4.2.2.4 Objective functions

Most of the above studies focus on conventional container terminals. In the recent decade, many terminals have been driven towards automated container terminals (ACTs) due to their low labour cost, low energy consumption, high safety, etc (Zhou et al., 2018; Wang et al., 2019). However, the container stacking problems at the ACTs are not yet adequately studied in the literature. This paper will address the SLAP at ACTs.

In conventional terminals, the yard blocks are typically positioned parallel to the quay, and containers are transferred between the yard crane and trucks at the empty lane at the side of each block. For export containers, this type of block layout can reduce the travel distance of the yard crane during the container retrieval process by storing export containers of the same group in the same bay to avoid the yard crane travelling across different bays. This is because export containers of the same group are usually loaded onto a ship consecutively (Kim and Park, 2003). However, for import containers, there is little chance to reduce such distance by optimising their storage locations because the retrieval sequence of import containers is unknown. Therefore, the objective functions of the SLAPs concerning the retrieval efficiency of import containers at conventional terminals mainly focus on the performance metric of the number of relocations (e.g., Kim and Kim, 1999; Saur íand Mart ń, 2011; Zhu et al., 2020).

In ACTs, the yard blocks are typically positioned perpendicular to the quay, and containers are transferred between the Automated Stacking Crane (ASC) and trucks at the ends of each block. Under this type of block layout, regardless of the retrieval sequence, to retrieve a container, the ASC needs to travel across a number of bays from the location of the container in the block to the transfer point at the end of the block. There is a trade-off between the ASC travel time and the number of relocations, which both

contribute to container retrieval times and truck waiting times. Some works have addressed the determination of container stacking positions at ACTs. For import containers, Yu and Qi (2013) propose three optimisation models under the non-segregation and segregation strategies to determine the number of import containers allocated to each bay. The objective function is to minimise the total retrieval time that is the sum of the expected relocation time and the ASC travel time. Park et al. (2011) propose an online search algorithm to first select the yard block with the lowest workload and then the yard stack with the minimum weighted sum of four criteria including the stacking cost, the retrieval cost, the relocation cost, and the waste of storage space. For export containers, Zhao et al. (2015) propose a simulation-based optimisation method for allocating outbound containers to yard bays aiming to minimise the quay crane waiting time. Gharehgozli and Zaerpour (2018) propose an integer programming model to determine the storage locations of outbound barge containers under a shared stacking strategy with the objective of minimising the total travel time of the ASC. Preston and Kozan (2001) develop a mixed integer programming model to determine the storage locations of export containers in a multimodal container terminal, in which it is implicitly assumed that the yard block is perpendicular to the quay. The objective is to minimise the total transfer time of each yard machine that is the sum of the travel time and the relocation time. In addition, a couple of studies consider both import and export containers. For example, Dekker et al. (2006) determine the container storage locations by several variants of category stacking rules and use a detailed simulation program to measure the workload of the ASC, the number of containers that cannot be stacked, and the number of relocations. Xia et al. (2016) determine the yard stacks allocated to each container group by a meta-heuristic to maximise the vessel handling efficiency by a weighted sum objective function involving the work balance among blocks and the travel distance between vessels and blocks.

It can be concluded that it is appropriate to use the total retrieval time (i.e. the sum of the crane travel time and the expected relocation time) as the objective function of the SLAP for import containers at ACTs (Yu and Qi 2013). Moreover, minimising the total retrieval time is also a good proxy for truck waiting times (Gharehgozli and Zaerpour, 2018). In the next sub-section, we will discuss some studies on estimating the number of relocations.

4.2.3 Container relocation estimation

Due to the uncertainty in the containers' retrieval sequence, the number of relocations during the retrieval process cannot be easily determined in advance (Bruns et al., 2016). Given an initial stacking configuration, the minimum number of relocations needed to retrieve all the containers depends on the retrieval sequence and the locations selected for the relocated containers. Such a problem is studied in the (Stochastic) Container Relocation Problem (c.f. Ku and Arthanari, 2016a, b, Galle et al., 2018a, Feng et al., 2020). Another relevant stream of research is to estimate the number of relocations based on the input parameters such as the stack dimensions, the container arrival and departure rates. In this stream, a few studies assume that new arrival containers are added to the stacks during the container retrieval process. For example, Sculli and Hui (1988) are among the first to explore the impacts of the store dimensions, the stacking policies, and the number of different types of containers on the relocation ratio by a simulation model, in which the arrival and departure rates of containers are assumed to be equal and random. De Castilho and Daganzo (1993) derive general formulas, which are functions of the total number stacks and the retrieval rate, to estimate the expected number of moves per container under the segregation and non-segregation stacking policies. On the other hand, some studies only consider retrievals. For example, under the assumption that containers are retrieved batch by batch in a random order, Zhou et al. (2020)

derive the number of relocations to retrieve all containers from a yard segment based on the storage space, the number of containers, and the number of container classes by a discrete event simulation. Assuming the probability for a container to be picked up next is the same among all the containers, Kim (1997) develop an approximated formula that is a function of the number of containers and the number of rows in the bay for the expected total number of relocations to retrieve all of the containers in a bay. Yu and Qi (2013) propose a more precise estimation than the one proposed by Kim (1997). These two formulas both rely on the number of rows/stacks in a bay.

All the above studies in the estimation stream simulate the number of actually incurred relocations based on a prescribed relocation strategy, that is, a realistic rule to select the stack for the relocated container. On the other hand, the expected number of blocking containers can also be a good proxy of the number of relocations, which can be derived without assuming any relocation strategy. In this respect, Galle et al. (2016) theoretically prove that the expected minimum number of relocations to retrieve all containers in a bay in a uniformly random order converges to a simple and intuitive lower bound when the number of stacks in a bay is large. The lower bound represents the expected number of blocking containers, which depends on only the number of containers in each stack. This lower bound is used in a recent study on yard crane scheduling by Galle et al. (2018b) to approximate the number of relocations to retrieval all the containers in a single stack when no information is assumed on the retrieval requests. In this paper, we do not assume any information on the retrieval sequence of non-smart containers, that is, they are retrieved in a uniformly random order, and no containers are added to the stacking area during the retrieval process. Such a problem environment is considered in Kim (1997), Yu and Qi (2013), and Galle et al. (2016). Given the advantage of the simplicity and the good approximation of the lower bound in Galle et al. (2016), we use this lower bound, i.e., the expected number of blocking containers, to approximate the expected number of relocations for retrieving the containers in a non-smart stack (see Section 4.3.4.2).

4.2.4 Research gap

There are only two most relevant studies that have investigated grouped-based stacking strategies for import containers aiming at minimising the expected number of relocations (Jang et al. 2013) or the total retrieval time (Yu and Qi 2013) during retrieval processes. In Jang et. al (2013), the unit loads (including inbound containers) are classified into multiple groups and the retrieval order is issued for a specific group. Each stack can store multiple groups of containers, and there is no decision on which groups should be relocation-free. In Yu and Qi (2013), import containers are categorised by the arrival times of incoming vessels. The emphasis is to analyse the long-term performance of various segregation and non-segregation strategies models by considering the dynamic of container arrivals and departures. There has been no research on allocating import containers to exact storage slots by utilising customer information alone while not relying on the container retrieval time. Motivated by the IFF program in practice, our work is the first academic research in this area. We propose a smart stacking strategy that explicitly differentiates smart stacks and non-smart stacks. A smart stack can only store containers from a single customer to guarantee zero relocation, while a non-smart stack can accept containers from multiple customers. Different from the existing stacking strategies, the smart stacking strategy incorporates intelligence into the stacking decision-making by simultaneously optimising the number of smart stacks, the customers or containers that are allocated to smart stacks and non-smart stacks, and the locations of these stacks by making use of the customer information of the containers. Specifically, we focus on the short-term operational decisions to allocate a batch of import containers to specific locations in a given storage area of a yard block under the smart stacking strategy, with the objective of minimising the total retrieval time including the ASC travel

time and the relocation time.

4.3 Problem Description

In this section, we describe the SLAP under consideration including the following aspect: the problem geometry, the problem definitions, two smart stacking policies, and the objective function.

4.3.1 Problem geometry

At an ACT, a yard block is oriented perpendicular to the quay, as shown in Fig. 4. 1. The configuration of a block consists of *M* bays, *R* rows, and *T* tiers. Bays are indexed by *b* from landside to seaside, $1 \le b \le M$, rows by *r* from left to right, $1 \le r \le R$, and tiers by *t* from bottom to top, $1 \le t \le T$. *R* is limited by the width of the ASC and *T* by the height of the ASC, which represents the maximum stacking height. Typically, *R* ranges from 6 to 13, *T* from 3 to 6, and *M* from 40 to 60. A stack is a vertical column located in a bay and a row, which can be characterised by a two-dimensional vector (*b*, *r*) representing its location on the ground. A slot is an unit space for storing a container located in a bay, a row and a tier, which can be characterised by a three-dimensional vector (*b*, *r*, *t*).

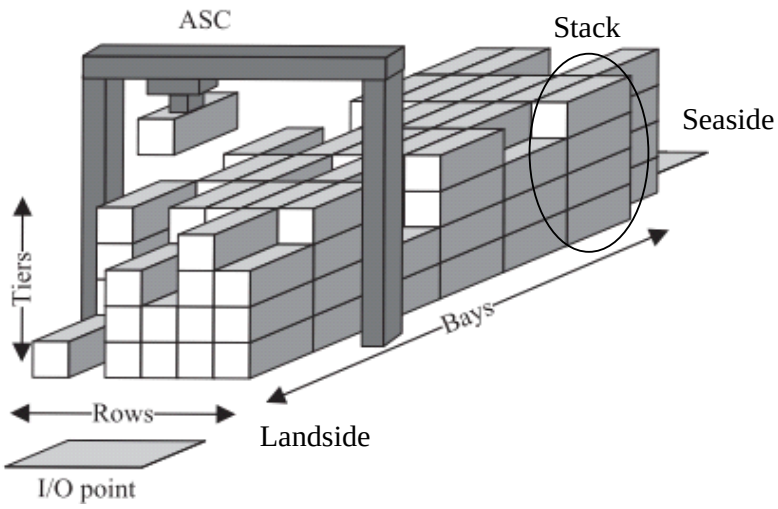


Fig. 4. 1. A stacking area in a yard block with a single ASC. Adapted from Gharehgozli and Zaerpour (2018)

At each end of the block, i.e., landside and seaside, there are several input/output (I/O) points where vehicles park waiting for the service of the ASC. During the import container retrieval process, an external truck with a retrieval request parks at one of the I/O points at the landside. The ASC picks up the required container in the block and then drop off it onto the truck at the I/O point. The location of the I/O point where the truck parks may affect the travel distance of the ASC along the row direction when serving the request. In this paper, since we do not schedule the ASC's working route, we assume that all trucks park at the middle I/O point, i.e., the middle point in front of the first bay, which is considered to be located at bay 0, row *R*/2 and tier 1. We name this delivery point as the depot, denoted by *o*. This assumption is in line with the literature (Gharehgozli and Zaerpour, 2018). It is also reasonable statistically because on average a truck can be regarded as being served in the middle I/O point. Moreover, we will show in Section 4.3.4 that after a certain bay, the travel time of the ASC is determined by the gantry crane travel time along the bay direction whereas the travel time along the row direction has no impact.

4.3.2 Problem definitions

We make the offline operational-level decision-making in a single planning period, in which we assign each import container to a slot in a given storage area in a single block. At the beginning of the planning period, we are given a batch of N incoming import containers to be stacked and a storage area where these containers are to be stored. The given storage area is composed of B empty bays (i.e., $B \times R$ empty stacks) in a block. Let Θ denote the set of the B bays, $|\Theta| = B$. Let b, $b \in \Theta$, denote both a bay and the actual bay number of itself in the block. Note that the B bays do not necessarily need to be located consecutively in the block. A smaller b corresponds to a bay closer to the landside and a bigger b corresponds to a bay closer to the seaside. It is worth noting that the models and solution approaches proposed in this paper can be easily extended to a non-empty storage area where some stacks have been occupied by existing containers.

As the yard is a scarce resource at container terminals (de Melo da Silva et al., 2018), terminal operators tend to make high utilisation of the yard storage space. Therefore, the number of available bays B should be limited depending on the yard utilisation rate. Usually, terminal operators would set an average utilisation rate for the yard space. Let u denote the utilisation rate of a bay, which is defined as the percentage of all its slots being occupied by the containers stored in the bay. In our experiments, given N, R and u, B is given by $B = \lceil N / \lfloor R \cdot T \cdot u \rfloor \rceil$, which represents the required number of empty bays to store N containers when each bay is utilised at its pre-set utilisation rate u.

The batch of N containers to be stacked belong to C customers. The containers are grouped by customers. Let $c \in \{1,...,C\}$ denote the index of customers or groups, and the number of containers in a group c is called group size, denoted by v_c. When all the containers in a stack are in the same group and the stack is not allowed to be used for relocation, no relocations are needed when retrieving the containers in this stack as containers are 'peeled off' from the top of the stack. We refer to such containers as *smart containers* (i.e., free-flow containers) and such stacks as *smart stacks*. In another situation, when the containers in a stack are from more than one group or all the containers in a stack are in the same group but the stack can be used for relocation, relocations may be needed for retrieving the containers in this stack. We refer to such containers as *non-smart containers* (i.e., non-free-flow containers) and such stacks as *non-smart stacks*. Accordingly, we refer to a bay in which all the stacks are smart as a *smart bay*, in which all the stacks are non-smart as a *non-smart bay*, and in which both smart and non-smart stacks exist as a *mixed bay*.

A smart stack will occupy the entire T slots of the stack regardless of how many containers are actually allocated to it. This is because a smart stack is dedicated to a single group and is not allowed to be used for relocation. If a bay is a smart bay, the capacity of the bay is equal to RT; otherwise, its capacity is equal to RT-(T-1) because at least (T-1) empty slots need to be reserved for relocation purpose in order to avoid deadlock (Tang et al., 2015; Chang and Zhu, 2019). To ensure the B bays are sufficient to store all the containers, the bay utilisation rate u must satisfy the condition $\lfloor R \cdot T \cdot u \rfloor \leq R \cdot T - (T-1)$.

The following assumptions are made for formulating the problem.

A1. Each container is associated with a customer. The containers belonging to the same customer form a group. If the customer information of a container is unknown, this container forms a group on its own.

A2. The containers in a smart stack are retrieved from the top to the bottom without requiring any relocation.

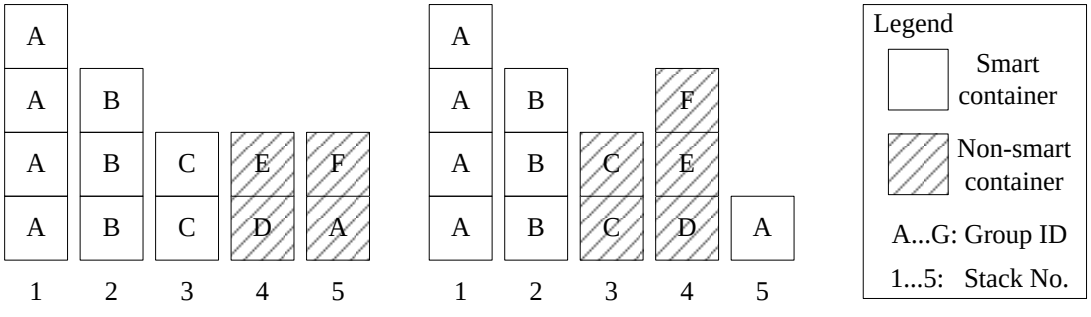
A3. The containers in a non-smart stack are retrieved in uniformly random order and relocations are needed.

4.3.3 Two smart stacking policies

Since a customer may have multiple containers and the capacity of a stack is restricted by the maximum stacking height *T*, more than one stacks may be needed to store the containers from a single customer. Note that customers may prefer to have all of their containers being smart or non-smart for the convenience of truck dispatching management. Two variants of the smart stacking strategy could be designed, depending on whether all or part of the containers of a customer are smart. Accordingly, we propose two smart policies, *non-split policy* and *split policy*.

Under the non-split policy, the group of containers is not allowed to be split between smart and non-smart stacks. In other words, they are either wholly allocated to smart stacks or wholly allocated to non-smart stacks; and we are concerned with which groups/customers should be smart. Under the split policy, a group of containers can be split between smart stacks and non-smart stacks; and we determine how many containers from a group should be smart. Under either policy, we need to determine where these smart stacks and non-smart stacks should be located in the block.

Fig. 4. 2 provides an example solution under the two policies respectively for a single bay to illustrate the influence of splitting on the number of relocations. Under both policies, the maximum stacking height *T*=4 forces the groups that have more than *T* containers to be divided and allocated to different stacks (e.g., group *A*). In Fig. 4. 2(a), under the split policy, the five containers of group *A* are split between smart stack 1 and non-smart stack 5 where one container of group *A* is mixed with one container from group *F*. In Fig. 4. 2(b), under the non-split policy, to make all the containers of group *A* smart, which are allocated to smart stacks 1 and 5, stacks 3 and 4 are determined to be non-smart stacks. Note that although the two containers in stack 3 are from the same group *C*, they are non-smart containers and stack 3 is a non-smart stack because stack 3 has to be used for accommodating relocations from stack 4 in order to avoid deadlock. Besides, one more container is added to stack 4 compared to that under the split policy, which increases the possibilities of relocation for stack 4. It can be seen that the split policy can save more relocations than the non-split policy.



(a) split policy (b) non-split policy

Fig. 4. 2. Illustration of the solutions under two smart policies for a single bay

In this paper, we optimise and evaluate the effects of smart stacking under both two policies. We focus more on analysing the non-split policy because it is closer to the current practice of container terminals and easier to implement from the perspective of customer administration. Although the flexibility of the split policy can bring more benefit in terms of the relocation reduction in retrieval time, it may cause inconvenience to customers, e.g., more complex truck dispatching and higher administration fee. In Section

4.4, we develop mathematical models under each policy, and then we compare the two models both theoretically (section 4.4.2) and computationally (section 4.6.5).

4.3.4 Objective function

The objective of our SLAP is to minimise the total retrieval time that is the sum of the ASC's travel time and relocation time. In the following two sub-sections, we present how the two components in the objective functions are measured.

4.3.4.1 ASC's travel time

We first introduce the ASC's working pattern, and then we give the mathematical expression of the travel time. In the end, we present two properties of the ASC's travel time.

The container retrieval operations are performed by a single ASC at the landside of the block. To serve a retrieval request, the ASC needs to perform both horizontal travel and vertical travel activities. During horizontal travels, the ASC moves its gantry along bays and its trolley along rows simultaneously in Chebyshev distance. During vertical travels, the ASC moves its spreader up and down. These travel activities can be divided into four phases as shown in Fig. 4. 3. Firstly, the crane performs an empty drive from its current position to the target stack where the requested container is stored (horizontal). Secondly, the crane lowers its spreader to pick up the container and then hoists the spreader up (vertical). Thirdly, the crane performs a loaded drive from the target stack to depot *o* (horizontal). Finally, the crane lowers its spreader to set down the container on the truck at the depot and then hoists the spreader up (vertical).

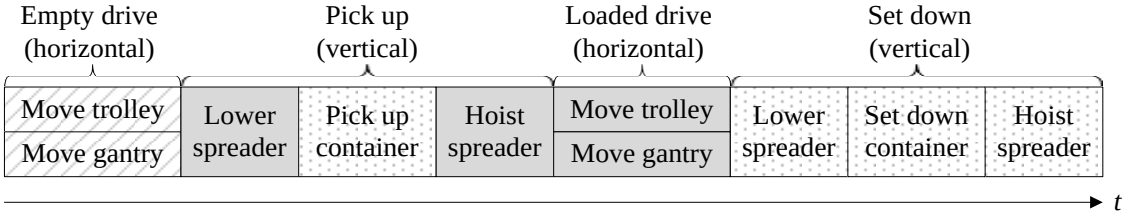


Fig. 4. 3. Pattern of typical yard crane movements for a retrieval cycle. Solid grey box indicates variable parts, dotted grey box indicates constant parts, and striped grey box indicates irrelevant parts. (Adapted from Galle et al. (2018b))

Based on this pattern, we now analyse the travel time spent in each phase and derive the relevant part that will be included in the optimisation model.

(1) Empty drive. The empty drive time depends on which stack the ASC currently stops when a retrieval request arrives, that is, where the ASC ended its previous request. This is usually optimised in the yard crane schedule problem (see e.g. Galle et al., 2018b) when multiple types (stacking and retrieval) of requests are considered and their stacking positions and service sequences are to be determined simultaneously. In this paper, since we only focus on the retrieval requests, the empty travel time is not relevant and thus is not included in the objective function.

(2) Loaded drive. The loaded drive time depends on the stack (b, r) where the requested container is stored, which is a variable part. Let T_b be the gantry moving time from bay b to bay zero (i.e., the artificial bay where the depot o is located), and T_r be the trolley moving time from row r to the middle of bay zero (i.e., the artificial row where the depot o is located). As the ASC moves along bays and rows simultaneously in Chebyshev distance, the ASC's loaded drive time is calculated as $T_{br} = \max\{T_b, T_r\}$.

(3) Pick up. The time spent in this phase consists of three parts: the spreader's lowering time (without a container), the time to handle and stabilize the container, and the spreader's hoisting time (with a container).

Among them, the time to handle and stabilize the container is constant (e.g., 20 seconds), and thus it is not included in the objective function. The times of the other two parts depend on the tier *t* where the requested container is stored, which is a variable part. Let T_t^E denote the time spent in lowering the spreader from the crane height (i.e., tier *T*) to tier *t*, and T_t^L the time spent in hoisting the spreader from tier *t* to tier *T*. Then, the variable part for the pick-up phase is expressed by $T_t = T_t^E + T_t^L$.

(4) Set down. Containers are dropped off to trucks at the depot. Since the locations of the trucks are fixed, the set-down time is constant and will not be included in the objective function.

To sum up, the variable part of the ASC's travel time to retrieve a container stored at slot (*b*, *r*, *t*) is expressed by $T_{brt} = \max\{T_b, T_r\} + T_t^E + T_t^L$. This variable part T_{brt} will be included in our objective function. Namely, only the relevant travel time of the ASC when retrieving a container is considered in the optimisation model.

In the following, we present two structural properties of the ASC's travel time. The properties will be utilised several times in the rest of the paper, such as for designing the heuristic algorithm and analysing the results of the experiments. Property 1 states that after a certain bay $\hat{b}$, the ASC's horizontal travel time depends only on the bay where the container is stored no matter which row it is stored; and after bay $\hat{b}$, the ASC's horizontal travel time increases with the bay where the container is stored. In our experiment in which a block has ten rows, $\hat{b} = 3$, indicating that the ASC's horizontal travel for retrieving a container located in a bay *b*, $b \geq 3$, is determined by only the bay index. Property 2 indicates that the ASC's vertical travel time decreases with the tier where the container is stored. The proofs of Property 1 and Property 2 are provided in Appendix A.

Property 1. Let l^x and l^y be the length and width of a 20-ft (twenty-foot equivalent unit) standard container, respectively. Let v^x be the ASC gantry moving speed with load and v^y be the ASC trolley moving speed with load. For a block with *R* rows and *M* bays, if there exists a bay $\hat{b}$ ($1 \leq \hat{b} \leq M$) that satisfies $\frac{\hat{b} \cdot l^x}{v^x} \geq \frac{R/2 \cdot l^y}{v^y} > \frac{b \cdot l^x}{v^x}$ ($1 \leq b < \hat{b} \leq M$), then $T_{br} = T_b$ for $1 \leq \hat{b} \leq b \leq M$ and $1 \leq r \leq R$, and $T_{br} > T_{b'r}$ for $\hat{b} \leq b' < b \leq M$ and $1 \leq r \leq R$.

Property 2. $T_t < T_{t'}$ for $1 \leq t' < t \leq T$.

4.3.4.2 Relocation time

Relocation moves are often inevitable when retrieving the containers from non-smart stacks since containers in these stacks are requested by external trucks in random order. As in the literature (Zhao and Goodchild, 2010; Yu and Qi, 2013), we assume that the number of relocations and the time needed to perform one relocation is independent, and thus the total relocation time is estimated by the expected number of relocations and the average time to relocate one container.

Since the exact number of relocations cannot be easily determined in advance, this part of the objective function is often replaced by a lower bound on the number of relocations (Bruns et al., 2016). When no information is available on the container retrieval sequence, the lower bound given by the expected number of blocking containers is regarded as a good proxy for the expected number of relocations when the number of stacks in a bay is large (Galle et al., 2016). Therefore, as in Galle et al. (2018b), we use the expected number of blocking containers to approximate the expected number of relocations. We define α_t to be the expected number of blocking containers in a stack of *t* containers in the case when no information on the containers' retrieval sequence is available. From Galle et al. (2016), we have $\alpha_0 = 0$ and

$$\alpha_t = t - \sum_{i=1}^{t} \frac{1}{i}, \quad \forall t \in \{1, ..., T\} \tag{4.1}$$

The expected number of relocations for a non-smart stack of *t* containers is calculated by Eq. (4.1). The advantage of using Eq. (4.1) is that it only requires the information of the number of containers per stack, which is easier to compute. Let $\overline{T}$ be the average time of relocating a container. Therefore, the total expected relocation time for a non-smart stack of *t* containers can be expressed by the product of α_t and $\overline{T}$.

4.4 Mathematical models

In this section, we present the mathematical models under the non-split policy and the split policy respectively. The notations used in both models as given as follows. The unique notations used in each model will be introduced when introducing the corresponding models.

Parameters:

N: the total number of containers to be stacked.

Θ: the set of empty bays, $|\Theta| = B$.

R: the number of rows (stacks) in a bay.

T: the maximum stacking height.

$\overline{T}$: the average time needed to perform a relocation (in seconds).

α_t: the expected number of relocations in a non-smart stack of *t* containers, which is defined by Eq. (4.1).

T_{brt}: the ASC's travel time to retrieve a container located at slot (*b*, *r*, *t*) (in seconds).

Auxiliary Variables:

z_b: equals zero if all the stacks in bay *b* are smart stacks, and one otherwise;

h_{br}: the number of non-smart containers in stack (*b*, *r*);

w_b: the expected number of relocations in bay *b*, which is a continuous variable;

f_{brt}: equals one if there are *t* non-smart containers stored in stack (*b*, *r*), and zero otherwise;

y_{brt}: equals one if there is a container stored at tier *t* of stack (*b*, *r*), and zero otherwise.

4.4.1 Non-split model

We now present the mathematical model under the non-split policy, that is, a group of containers is either wholly allocated to smart stacks or wholly allocated to non-smart stacks. We first develop an original formulation, and then we develop an improved formulation by enhancing the variable representation of the original one. Both formulations are mixed-integer programming (MIP) models. The key decision variables in the non-split model are which groups (customers) should be selected as *smart groups*, how the containers from a smart group should be distributed to multiple smart stacks, and where these smart stacks should be located in the block. Note that all the containers in a smart group should be allocated to some smart stacks.

4.4.1.1 Original formulation

The newly defined parameters and decision variables in the original formulation are as follows.

Parameters:

v_c: the size of group *c*.

Decision Variables:

x_{br}^{ct} : equals one if stack (*b*, *r*) is a smart stack that is allocated to *t* number of containers of group *c*, and zero otherwise;

s_c : equals one if group *c* is a smart group, and zero otherwise;

The original formulation (denoted by M1) is presented below:

[M1]:

$$\min \sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta} w_b \cdot \bar{T} \tag{4.2}$$

s.t.

$$\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct} \le 1, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\} \tag{4.3}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{ct}\cdot t = s_c \cdot v_c, \quad \forall c\in\{1,...,C\} \tag{4.4}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R}\left(\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}\cdot t + h_{br}\right) = N \tag{4.5}$$

$$z_b \le R - \sum_{r=1}^{R}\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}, \quad \forall b\in\Theta \tag{4.6}$$

$$z_b \cdot R \ge R - \sum_{r=1}^{R}\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}, \quad \forall b\in\Theta \tag{4.7}$$

$$\sum_{r=1}^{R}\left(\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}\cdot T + h_{br}\right) \le RT - (T-1)\cdot z_b, \quad \forall b\in\Theta \tag{4.8}$$

$$h_{br} \le \left(1 - \sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}\right)\cdot T, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\} \tag{4.9}$$

$$\sum_{t=1}^{T} y_{brt} = \sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct}\cdot t + h_{br}, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\} \tag{4.10}$$

$$y_{brt} \le y_{br,t-1}, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\}, \forall t\in\{2...,T\} \tag{4.11}$$

$$w_b = \sum_{r=1}^{R}\sum_{t=1}^{T} \alpha_t \cdot f_{brt}, \quad \forall b\in\Theta \tag{4.12}$$

$$\sum_{t=1}^{T} f_{brt} \le 1, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\} \tag{4.13}$$

$$\sum_{t=1}^{T} t\cdot f_{brt} = h_{br}, \quad \forall b\in\Theta, \quad \forall r\in\{1,...,R\} \tag{4.14}$$

$$y_{brt}\in\{0,1\}, \quad f_{brt}\in\{0,1\}, \quad \forall b\in\Theta, \forall r\in\{1,...,R\}, \forall t\in\{1...,T\} \tag{4.15}$$

$$z_b\in\{0,1\}, \quad w_b \ge 0, \quad \forall b\in\Theta \tag{4.16}$$

$$h_{br} \in Z+, \quad \forall b\in\Theta, \forall r\in\{1,...,R\} \tag{4.17}$$

$$s_c\in\{0,1\}, \quad \forall c\in\{1,...,C\} \tag{4.18}$$

$$x_{br}^{ct}\in\{0,1\}, \quad \forall b\in\Theta, \forall r\in\{1,...,R\}, \forall c\in\{1,...,C\}, \quad \forall t\in\{1...,T\} \tag{4.19}$$

The objective function (4.2) is to minimise the total retrieval time, which is the sum of the total ASC travel time and the total relocation time. Constraints (4.3) ensure that each smart stack is allocated to a specific number of containers of at most one group. Constraints (4.4) guarantee the non-split policy, that is, for a smart group, all the containers in the group are allocated to smart stacks. Constraints (4.5) ensure that

all given containers are stored in the given storage area. Constraints (4.6) and (4.7) define the auxiliary decision variables z_b that indicate whether all the stacks in a bay are smart. z_b is forced to equal zero if all the stacks in bay *b* are smart and equal one if there are non-smart stacks in bay *b*. Constraints (4.8) guarantee the capacity feasibility of each bay. For a smart bay (i.e, $z_b = 0$), the capacity of the bay is equal to *RT*; otherwise, its capacity is equal to *RT*-(*T*-1) because at least (*T*-1) empty slots need to be reserved for relocation in order to avoid deadlock. Constraints (4.9) guarantee that the height of a non-smart stack is not more than *T*. Constraints (4.9) also ensure that there is no non-smart container in a smart stack. If stack (*b*, *r*) is a smart stack, i.e., $\sum_{c=1}^{C}\sum_{t=1}^{T} x_{br}^{ct} = 1$, h_{br} is forced to equal zero. Constraints (4.10) and (4.11) determine the height of each stack and guarantee that containers are stacked from the ground and are stacked on top of one another. Constraints (4.12) calculate the total relocation time for retrieving the containers in a bay. Constraints (4.13) and (4.14) define the auxiliary decision variables f_{brt} by h_{br}. If $h_{br} = 0$, $f_{brt} = 0$, $\forall t \in \{1...,T\}$; otherwise, $f_{br,h_{br}} = 1$ and $f_{brt} = 0$, $\forall t \in \{1,...,T\} / h_{br}$. Finally, Constraints (4.15)-(4.19) specify the domains for the decision variables.

4.4.1.2 Properties of optimal solutions

We now propose two propositions regarding the properties of optimal solutions of the original formulation, based on which we will develop an improved formulation in the next sub-section.

First, we define a *pile* as the set of containers stacked in the same stack. We refer to a pile that is stacked to the maximum stacking height as a *full pile*. A smart pile is implied by the smart stack.

Proposition 1. Let h_{p_i} be the height of pile p_i. In the optimal solution to M1, for any two smart piles p_i and p_j which are located at stacks (b_i, r_i) and (b_j, r_j) respectively, if $h_{p_i} > h_{p_j}$, then $T_{b_i r_i} \le T_{b_j r_j}$.

Sketch of the proof. The proof is proved in Appendix A. We suppose by contradiction that $h_{p_i} > h_{p_j}$ and $T_{b_i r_i} > T_{b_j r_j}$ in the optimal solution σ^*. We construct a feasible solution σ' such that $h_{p_i} > h_{p_j}$ and $T_{b_i r_i} < T_{b_j r_j}$, and we show that σ' leads to a smaller objective value than σ^* which provides a contradiction to σ^* being an optimal solution.

Proposition 2. Let P_c be the number of piles of a smart group *c* in the optimal solution, then $P_c = \lceil v_c / T \rceil$, and the P_c piles are composed of P_c -1 full piles and one pile whose height equals $v_c - (P_c - 1) \cdot T$.

Sketch of the proof. The proof is provided in Appendix A. Let us rank the P_c piles in ascending order of the ASC's horizontal travel time needed to retrieve a container from the stack where the pile is located. According to Proposition 1, the P_c piles can be ordered as $p_1, p_2, ..., p_{P_c}$ with $h_{p_1} \ge h_{p_2} \ge .... \ge h_{p_{P_c}}$ and $T_{b_1 r_1} \le T_{b_2 r_2} \le .... \le T_{b_{P_c} r_{P_c}}$. There are two cases depending on the size of v_c, i.e., $v_c \ge T$ and $v_c < T$. In the case of $v_c \ge T$, it is sufficient to suppose by contradiction that in the optimal solution there is at least one non-full pile in the first P_c -1 piles. Let p_i be the highest non-full pile in the first P_c -1 piles. We can

construct a feasible solution such that the height of p_i is increased by one by moving one container from pile p_{P_c} to pile p_i. In the case of $v_c < T$, it is sufficient to suppose by contradiction that in the optimal solution $h_{p_1} < v_c$. We can construct a feasible solution by moving one container from pile p_{P_c} to pile p_1. In both cases, the feasible solution we construct can lead to a lower objective value than the optimal solution, which provides a contradiction to the supposing.

Proposition 2 implies that among the P_c piles of smart group c, the pile located at the stack with the greatest ASC's horizontal travel time has $v_c - (P_c - 1) \cdot T$ containers, and all the other piles are full piles.

4.4.1.3 Improved formulation

Based on the propositions of the optimal solution in Section 4.4.1.2, we can propose an improved formulation, which is easier to solve than [M1].

To simplify the narrative, we introduce the concept of optimal partition. The idea of the new formulation is to make decisions on which customers and partitions should be smart without the need of associating the smart partitions with specific smart customers. Proposition 2 indicates that in the optimal solution of the original formulation, each group is partitioned into $P_c = \lceil v_c / T \rceil$ piles such that P_c -1 piles have T containers and one pile has $v_c - (P_c - 1) \cdot T$ containers. We define the *optimal partition* of a group as the pattern that divides the group into the partitions each corresponding to a pile of the group in the optimal solution. Let $\pi_c = \{p_i \mid i \in [1, ..., P_c]\}$ denote the set of partitions of group c by using the optimal partition, where $P_c = \lceil v_c / T \rceil$ is the number of partitions of group c, p_i is the number of containers in partition i. Without the loss of generality, we let $p_i = T$, $i \in \{1, ..., \lceil v_c / T \rceil - 1\}$, and $p_{\lceil v_c/T \rceil} = v_c - (P_c - 1) \cdot T$.

We partition each group by using the optimal partition. With such pre-processing, the improved formulation is developed based on the unit of partition. The newly defined parameters and decision variables in the improved formulation are as follows.

Parameters:

u_{ct}: the number of partitions with t number of containers for group c, $c \in \{1, ..., C\}$, $t \in \{1, ..., T\}$, which is defined by

$$u_{ct} = \begin{cases} P_c - 1, t = T \\ 1, t = v_c - (P_c - 1) \cdot T \\ 0, t = \{1, ..., T\} / \{T, v_c - (P_c - 1) \cdot T\} \end{cases}$$

Decision Variables:

x_{br}^t: equals one if stack (b, r) is a smart stack that is allocated to a partition with t number of containers, $t \in \{1, ..., T\}$, and zero otherwise.

The improved formulation (denoted by M2) is presented below:

[M2]:
$$\min \sum_{b \in \Theta} \sum_{r=1}^{R} \sum_{t=1}^{T} T_{brt} \cdot y_{brt} + \sum_{b \in \Theta} w_b \cdot \bar{T} \tag{4.2}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} x_{br}^t \le 1, \quad \forall b \in \Theta, \quad \forall r \in \{1, ..., R\} \tag{4.20}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{t} = \sum_{c=1}^{C} s_c u_{ct}\,, \quad \forall t \in \{1,...,T\} \tag{4.21}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R}\left(\sum_{t=1}^{T} x_{br}^{t}\cdot t + h_{br}\right) = N \tag{4.22}$$

$$z_b \le R - \sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{t}\,, \quad \forall b \in \Theta \tag{4.23}$$

$$z_b \cdot R \ge R - \sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{t}\,, \quad \forall b \in \Theta \tag{4.24}$$

$$\sum_{r=1}^{R}\left(\sum_{t=1}^{T} x_{br}^{t}\cdot T + h_{br}\right) \le RT - (T-1)\cdot z_b\,, \quad \forall b \in \Theta \tag{4.25}$$

$$h_{br} \le \left(1 - \sum_{t=1}^{T} x_{br}^{t}\right)\cdot T\,, \quad \forall b \in \Theta\,, \quad \forall r \in \{1,...,R\} \tag{4.26}$$

$$\sum_{t=1}^{T} y_{brt} = \sum_{t=1}^{T} x_{br}^{t}\cdot t + h_{br}\,, \quad \forall b \in \Theta\,, \quad \forall r \in \{1,...,R\} \tag{4.27}$$

$$x_{br}^{t} \in \{0,1\}\,, \quad \forall b \in \Theta\,, \forall r \in \{1,...,R\}\,, \forall t \in \{1,...,T\} \tag{4.28}$$

Constraints (4.11) – (4.18)

The objective function (4.2) is the same as in [M1]. Constraints (4.20) ensure that each smart stack is allocated to a partition with a specific number of containers. Constraints (4.21) guarantee the non-split policy by forcing the total number of partitions with *t* number of containers for all the smart groups equals the total number of smart stacks each storing *t* containers. Constraints (4.22) ensure that all given containers are stored in the given storage area. Constraints (4.23) and (4.24) define the auxiliary decision variables z_b that indicate whether all the stacks in a bay are smart. Constraints (4.25) guarantee the capacity feasibility of each bay. Constraints (4.26) guarantee that the height of a non-smart stack is not more than *T* and there is no non-smart container in a smart stack. Constraints (4.27) determine the height of each stack. Constraints (4.11) – (4.18) inherit from [M1].

Proposition 3 states that the optimisation problem defined by [M1] and the optimisation problem defined by [M2] are equivalent.

Proposition 3. [M1] and [M2] are equivalent problems.

Sketch of the proof. The proof is provided in Appendix A and is in three parts. In order to prove the equivalence between [M1] and [M2], we first show there is an implied constraint for [M1] that can be derived by the definition of u_{ct}. By using this implied constraint, we can reformulate [M1] into an equivalent counterpart [M1-1]. Secondly, by using the transformation between x_{br}^{ct} and x_{br}^{t}, we can reformulate [M1-1] into an equivalent counterpart [M1-2]. Lastly, we show that [M1-2] and [M2] are equivalent. □

The improved formulation enhances the representation of the variables x_{br}^{ct} in the original one by using the new variables x_{br}^{t}. By such enhancement, the number of variables is reduced significantly as x_{br}^{t} is not associated with group *c*. When implementing the solution of [M2], terminal operators need the information regarding which group a smart stack is allocated to. As proved in Proposition 3 (in Appendix A), given the solutions of x_{br}^{t} and s_c in [M2], we can obtain such information through deriving x_{br}^{ct} by the following formulas:

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{ct} = s_c \cdot u_{ct}\,, \quad \forall c \in \{1,...,C\}\,, \quad \forall t \in \{1,...,T\} \tag{4.A.5}$$

$$x_{br}^{t} = \sum_{c=1}^{C} x_{br}^{ct}\ ,\quad \forall b \in \Theta\ ,\quad \forall r \in \{1,...,R\}\ ,\quad \forall t \in \{1,...,T\} \tag{4.A.7}$$

4.4.2 Split model

In this section, we develop the mathematical model under the split policy, that is, a group of containers can be split between smart stacks and non-smart stacks. Intuitively, selecting a full pile to be a smart stack is more beneficial than selecting a non-full pile to be a smart stack because of the saving of relocations and better utilisation of the stack space. The split policy offers the opportunities to increase the number of full smart piles and decrease the number of non-full smart piles. We restrict the split policy to the cases that all smart piles must be selected from the optimal partitions defined in Section 4.4.1.3. This treatment makes the split model easy to compare with the non-split model.

We pre-process each group by the optimal partition as that in Section 4.4.1.3, and as a result, we get a total of $\sum_{c=1}^{C}\sum_{t=1}^{T} u_{ct}$ partitions for all the containers. We define a new parameter $n_t = \sum_{c=1}^{C} u_{ct}$ that denotes the total number of partitions with *t* number of containers, $t \in \{1,...,T\}$, in which u_{ct} is defined in Section 4.4.1.3. The decisions of the split model focus on which partitions should be smart. The newly defined parameters and decision variables in the split model are as follows. Note that x_{br}^{t} has been defined for [M2], we introduce it here again for the purpose of differentiating it from the decision variable x_{br}^{ct} in [M1].

Parameters:

n_t : the total number of partitions with *t* number of containers, $t \in \{1,...,T\}$, which is defined by $n_t = \sum_{c=1}^{C} u_{ct}\ ,\quad t \in \{1,...,T\}$.

Decision variables:

x_{br}^{t} : equals one if stack (*b*, *r*) is a smart stack that is allocated to a partition with *t* number of containers, $t \in \{1,...,T\}$, and zero otherwise.

The split model (denoted by M3) is presented below:

[M3]:

$$\min \sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta} w_b \cdot \bar{T} \tag{4.2}$$

$$\text{s.t.}\quad \sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{t} \le n_t\ ,\quad \forall t \in \{1,...,T\} \tag{4.29}$$

Constraints (4.11) – (4.18), (4.20), and (4.22) – (4.28)

Constraints (4.29) ensure that the number of smart stacks allocated to the partitions with *t* number of containers is not more than the total number of partitions with *t* number of containers. The other constraints inherit from [M2].

The following two lemmas compare the split model and the non-split model theoretically. Lemma 1 states that the objective value of [M3] is not greater than that of [M2]. Lemma 2 states that [M2] and [M3] have the same objective value when the maximum group size is no more than the maximum stacking height *T*. The proofs of Lemma 1 and Lemma 2 are provided in Appendix A.

Lemma 1. Let $f(\text{M2})$ and $f(\text{M3})$ be the objective functions of the optimisation problems defined by [M2] and [M3], respectively, then $f(\text{M3}) \le f(\text{M2})$.

Lemma 2. Let $V^m = \max_{c\in\{1,...,C\}}\{v_c\}$, then $f(\text{M2}) = f(\text{M3})$ when $V^m \le T$.

4.4.3 Complexity of the SLAP

The SLAP is NP-hard in general. This can be proved by reducing the Set Partitioning Problem to a special instance of the SLAP. We define a 'bay profile' as a feasible assignment of containers to a bay, and each bay profile is associated with a total time for retrieving all the containers in this bay profile. For all available bays, a finite set of all feasible bay profiles can be defined in advance, denoted by *P*. Consider a special case of the SLAP, in which we are given only a subset of *P*, denoted by P'. Then, the remaining task of the SLAP is to find a subset of bay profiles from P' with the minimum total retrieval time subject to that it covers all of the containers and no two of the bay profiles share the same container. This instance is equivalent to the Set Partitioning Problem. Therefore, we are able to reduce the Set Partitioning Problem to a special case of the SLAP. Because the Set Partitioning Problem is known to be NP-hard (Rasmussen and Larsen, 2011), the SLAP is NP-hard.

Due to the NP-hardness of the SLAP, the proposed MIPs are computationally expensive to solve for large-scale problems. More importantly, the computational times required to solve the MIPs are not applicable for real-time decision-making (c.f. the results in Section 4.6.3). Therefore, in the next section, we will develop an efficient heuristic algorithm that is able to find near-optimal solutions within several seconds for realistic scaled SLAPs.

4.5 Divide-and-conquer heuristic

In this section, we develop a heuristic algorithm for the non-split variant by employing the divide-and-conquer strategy. The framework of the heuristic is introduced in Section 4.5.1 and the details are presented in Sections 4.5.2-4.5.5.

4.5.1 Framework

Divide-and-conquer is an important paradigm to design computationally efficient algorithms in computer science (Li et al., 2009). The principle underlying divide and conquer algorithms is that: the original problem is decomposed into two or more subproblems until they become sufficiently simple to be solved directly; the subproblems are solved independently and their solutions are composed to give a solution to the original problem (Smith et al., 1985). The divide-and-conquer paradigm has become a commonly used strategy to design efficient algorithms for complicated combinatorial optimisation problems (e.g., Reimann et al., 2004; Jin et al., 2016; Wei et al., 2019).

The structure of the non-split model [M2] motivates us to employ the divide-and-conquer strategy to decompose the original problem into several subproblems and solve them sequentially and iteratively. It is observed that in [M2], the decision vector $\boldsymbol{x}$ about the smart stacks and the decision vector $\boldsymbol{h}$ about the non-smart stacks are coupled only by Constraints (4.22) and (4.25) - (4.27). By relaxing these constraints, the original problem can be decomposed into two smaller subproblems: one involving only the smart stack decisions and the other only concerning the non-smart decisions, which are much easier to solve. The solution to the original problem can be obtained by solving a third subproblem in which the solutions to the first two subproblems are combined.

Fig. 4. 4 provides the framework of the divide-and-conquer (D&C) heuristic. At each iteration, three subproblems are solved sequentially based on the updated number of available bays. Initially, all the given *B* empty bays are taken into account; then at each later iteration, the number of available bays is reduced by one. Subproblem 1 is first solved to determine the smart piles. Here, we use "smart piles" rather than "smart stacks" because subproblem 1 only concerns the heights of smart stacks but does not determine their

locations; instead, the smart piles are considered to be temporarily stacked in a certain area in the block to guarantee the feasibility of the capacity constraint. After that, the number of non-smart containers and the remaining storage space for non-smart containers can be obtained and passed to subproblem 2 where the non-smart piles are determined subsequently. Finally, the smart piles and the non-smart piles resulted from subproblems 1 and 2 are passed to subproblem 3, in which the locations of each pile are determined and the objective function is updated. The algorithm terminates at the iteration with B^{min} bays, which is the minimum number of required bays supposing each bay can be fully utilised, or the iteration where the objective function stops improving.

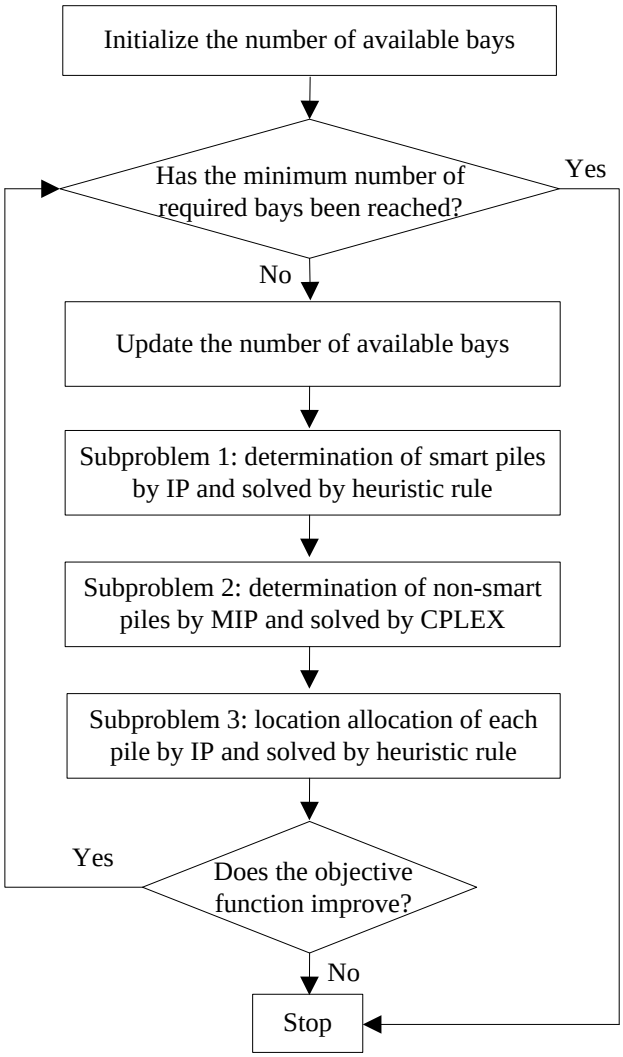


Fig. 4. 4 The framework of the divide-and-conquer algorithm

4.5.2 Updating scheme and stopping criteria

Let Θ_i denote the set of available bays at the *i*th iteration, and B_i the size of Θ_i. Θ_0 is initialized as Θ and B_0 is initialized as B. After solving the three subproblems at each iteration, Θ_{i+1} is updated by removing the bay closest to the seaside in Θ_i, and accordingly, the number of available bays is updated by $B_{i+1}=B_i - 1$. The idea behind this updating scheme is to minimise the objective function through the trade-off mechanism between the ASC travel time and the relocation time. When the available storage

space gets larger, the number of relocations tends to decrease but the retrieval time tends to increase. The objective function is minimised at a certain point where the number of occupied stacks is optimal. Therefore, by scanning the number of available stacks, we are able to minimise the objective function. However, instead of scanning the number of stacks, we can reduce the number of iterations and thus save the solution time by scanning the number of bays. Although this does not guarantee optimality, it is expected that the optimality gap is small because of the property of the ASC travel time stated in Property 1.

Let B_1 denote the set of bays in Θ_i located before bay $\hat{b}$ ($\hat{b}$ is the minimum bay in the block that satisfies $T_{\hat{b}r} = T_b$ for any $1 < r < R$), and B_2 the set of bays in Θ_i located after bay $\hat{b}$ (including bay $\hat{b}$ if $\hat{b}$ is in Θ_i). According to Property 1, for the bays in B_1, the ASC's horizontal travel time for retrieving a container depends on both the row index and the bay index of the stack where the container is stored; and for the bays in B_2, it only depends on the bay index. In the realistic situation, in most cases, $|B_1|$ is much smaller than B, which means that the bays in B_1 must be occupied. Since for the bays in B_2, the ASC's horizontal travel time depends only on the bay index, there would be no much difference among the solutions with one more stack available or one less stack available in such bays.

On solving the three subproblems at each iteration, the total retrieval time can be obtained. The search process terminates either if the iteration reaches the minimum number of required bays $B^{\min}$ or if the objective function does not improve. $B^{\min}$ is calculated by $B^{\min} = \lceil N/(R \cdot T) \rceil$, supposing each bay can be fully utilised.

4.5.3 Subproblem 1: smart piles

Subproblem 1 deals with the selection of smart piles. The objective of subproblem 1 is to maximise the number of smart containers. By considering the constraints associated with only smart containers, subproblem 1 for the *i*th iteration can be formulated as an integer programming model denoted by [Sub1] in Appendix B.1.

In [Sub1], the decisions on the heights of smart piles and the locations of smart piles are bound together by the decision variable $\boldsymbol{x}$. As the locations of smart piles will be determined in subproblem 3, here, we only need to determine smart groups $\boldsymbol{s}$. With $\boldsymbol{s}$, we can obtain the heights of each smart pile. In order to guarantee the feasibility of the original problem, we need to make sure that a feasible solution can be found in subproblem 2, and thus the feasibility of Constraints (4.22), (4.25) and (4.26) in [M2] should be maintained when solving [Sub1]. Therefore, smart groups should be determined such that there are as many smart containers as possible and the remaining storage space is still enough to accommodate the non-smart containers. For this purpose, we design a heuristic rule to select smart groups.

The basic idea of the heuristic rule is to give priority to the groups who can make more contribution to reducing the number of relocations and meanwhile can utilise the storage space more efficiently. For this purpose, we introduce a group score d_c to represent the average height of each pile of group c, which is defined by $d_c = v_c / P_c$. Groups with higher d_c are given priority over those with lower d_c when selecting smart groups. For the groups with the same value of d_c, the groups with a greater number of partitions (P_c) are given priority. The rationality behind this is that a group with a higher d_c has a higher utilisation rate of the stacks and thus more space can be saved for storing other smart containers, and a group with a greater

number of partitions can bring more smart containers and thus can lead to fewer relocations. Therefore, these groups are more promising to become smart groups in the optimal solution.

The details of the heuristic rule used for determining the smart groups are provided in Appendix B.2.

4.5.4 Subproblem 2: non-smart piles

After solving subproblem 1, suppose there are N_n non-smart containers, S_s stacks for storing smart containers and S_n stacks for storing non-smart containers. Subproblem 2 determines the heights of non-smart piles. The objective is to minimise the total retrieval time of the N_n non-smart containers. Subproblem 2 can be formulated as a MIP model denoted by [Sub2] in Appendix B.3. As [Sub2] only involves the decisions of non-smart containers, it can be solved by CPLEX efficiently.

[Sub2] is developed under the assumption that a specific storage area in the block has been pre-allocated to the non-smart containers. In order to guarantee the feasibility of Constraints (4.25) and (4.26), we temporarily divide the given storage area into smart bays, non-smart bays and mixed bays. The division method is described below, and an example illustrating the pre-allocated storage area is provided in Appendix B.4.

The bays in Θ_i are divided into three subsets, which are Θ_i^1, Θ_i^2 and Θ_i^3. Let Φ_T denote the set of full smart piles output from subproblem 2, that is, the smart piles with T number of containers. First, the first $\lfloor |\Phi_T| / R \rfloor$ bays near the landside in Θ_i, denoted by Θ_i^1, are allocated to Φ_T, which are smart bays. Second, the next $\lceil S_n / R \rceil$ bays in Θ_i, denoted by Θ_i^2, are allocated to all the non-smart containers, which are non-smart bays. Note that the last bay in Θ_i^2, denoted by $\overline{B}$, can be a mixed bay depending on the value of $S_n \% R$. Let $\hat{R} = S_n \% R$. Recall that we refer to a bay shared by both smart stacks and non-smart stacks as a mixed bay. If $\hat{R} = 0$, bay $\overline{B}$ is a non-smart bay. If $\hat{R} > 0$, bay $\overline{B}$ will be a mixed bay and $\hat{R}$ represent the number of non-smart stacks in this bay. In this case, a part of the stacks of bay $\overline{B}$ are reserved for smart containers and thus are not allowed to store non-smart containers. We suppose that the (R- $\hat{R}$) number of stacks at the right side of bay $\overline{B}$ are reserved for smart piles and the $\hat{R}$ number of stacks at the left side are allocated to non-smart piles. Last, the last $\lfloor S_s / R \rfloor - |\Theta_i^1|$ bays in Θ_i, denoted by Θ_i^3, are reserved for the remaining smart piles that are neither in Θ_i^1 nor in the mixed bay. Note that $S_n + S_s = B_i$ and $|\Theta_i^1| + |\Theta_i^2| + |\Theta_i^3| = B_i$. If $\hat{R} = 0$, $|\Theta_i^2| \cdot R = S_n$ and $|\Theta_i^1| \cdot R + |\Theta_i^3| \cdot R = S_s$; otherwise, if $\hat{R} > 0$, $\left(|\Theta_i^2| - 1\right) \cdot R + \hat{R} = S_n$ and $|\Theta_i^1| \cdot R + (R - \hat{R}) + |\Theta_i^3| \cdot R = S_s$.

The rationale behind this pre-allocation is that in the optimal solution, according to Proposition 1, a higher stack would be allocated to a bay with a smaller index. Since the height of a non-smart pile will be no greater than T, it is reasonable to pre-allocate the first $|\Theta_i^1|$ bays to full smart piles. In addition, the purpose of solving subproblem 2 is to obtain the heights of non-smart piles rather than their locations which will be determined in subproblem 3. Pre-allocating Θ_i^2 to be the non-smart area makes the non-smart piles have competitive heights to compete with the smart piles for the bays near the landside in subproblem 3.

4.5.5 Subproblem 3: location allocation

After solving subproblems 1 and 2, we have obtained smart piles and non-smart piles, and the relocation time has been determined. Subproblem 3 is to determine the locations of the smart piles and non-smart piles to minimise the ASC travel time, which can be formulated as an integer programming model denoted by [Sub3] in Appendix B.5.

After conducting a preliminary experiment by CPLEX, we found that [Sub3] is still computationally expensive for larger instances. Therefore, we design a heuristic rule to solve [Sub3]. The basic idea behind the heuristic rule is to allocate higher piles to the stacks near the landside. The rationale is that generally, the ASC's horizontal travel time tends to increase with the bay number as stated in Property 1. In the heuristic, first, we construct temporary bays. The smart and non-smart piles obtained from subproblems 1 and 2 are allocated to the smart and non-smart storage area respectively which are pre-allocated in subproblem 2. Second, we re-allocate these temporary bays based on the unit of bay by taking advantage of Property 1. These temporary bays are sorted in descending order according to the total number of containers in each bay and then the sorted bays are assigned to the locations of the bays in Θ_i from the landside to the seaside. Last, we re-allocate the piles in B_1 based on the unit of pile to seek possible savings on the travel time since for $b \in \mathrm{B}_1$, T_{br} also depends on r. A higher pile is re-allocated to a stack with greater T_{br}. The advantage of the heuristic is that we do not need to consider the capacity feasibility since all the bays meet the capacity constraints. The details of the heuristic rule used for allocating the locations of the smart piles and non-smart piles are provided in Appendix B.6.

Based on the locations of each pile, the total ASC travel time can be obtained. Then, the total retrieval time is returned by the sum of the relocation time obtained by [Sub2] and the ASC travel time obtained in subproblem 3.

4.6 Computational experiments

In this section, we present extensive computational experiments to validate the effectiveness and efficiency of the proposed solution approach. CPLEX 12.9 is used as the MIP solver to solve [M1], [M2], [M3] and [Sub2]. All the experiments are programmed in C++ (VS2015) and are performed on a desktop with Intel® Core ™ i5-7500 3.40 GHz CPU, 8 GB of RAM, and 64-bit Windows 10 Enterprise.

4.6.1 Experiment design and instance generation

We present four sets of experiments, which are illustrated in Fig. 4. 5. Firstly, we compare the computational efficiency of the two formulations of the non-split model, by which we show the superiority of the improved formulation over the original one. Secondly, we verify the effectiveness and efficiency of the proposed heuristic algorithm by comparing it with the improved formulation solved by CPLEX. Thirdly, we evaluate the effectiveness of the smart stacking strategy by comparing the proposed heuristic with a commonly used rule in practice. We also examine the impacts of customer information and bay utilisation rates on the performance and the effectiveness of the smart stacking strategy. Lastly, we compare the performances of the two variants of models to evaluate the benefit of the split policy.

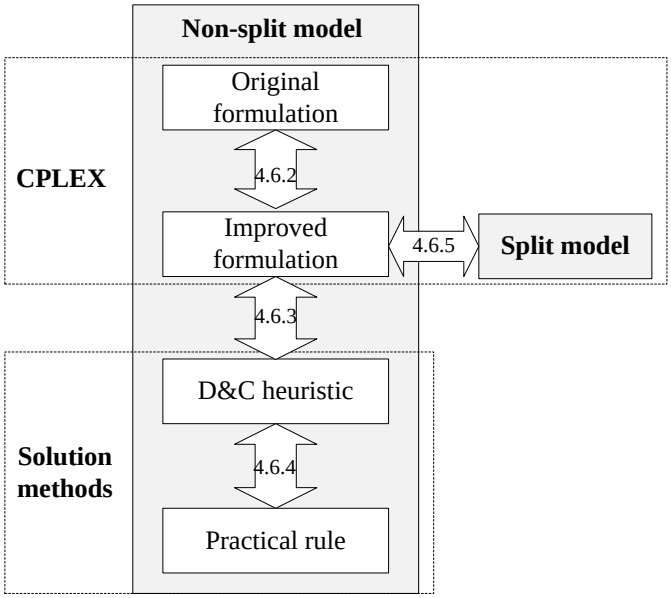


Fig. 4. 5 The diagram of the experiment design

Table 4. 1 lists the sets of parameters used in the experiments. To evaluate the performance of our proposed models and heuristic algorithm under different cases, each parameter is varied within a range of scenarios. Such variations can cover the dimension of a yard block in most of the modern ACTs (Galle et al., 2018b) and the practical scales of containers to be stacked into a block in a planning period (Yu and Qi, 2013). We also set a base instance that represents the dimension and the utilisation rate of a bay at a typical ACT. Given N, R, T, and u, the number of available bays (B) in the storage area is obtained by $B = \lceil N / \lfloor R \cdot T \cdot u \rfloor \rceil$. In the experiments, for the convenience of analysing and interpreting the results, the B bays are located consecutively in the block, that is, from bay 1 to bay B. We use three-tuple, "the number of containers to be stacked (N) | the dimension of the bay ($R \times T$) | the number of bays (B)", to represent the problem class. We do not distinguish the scales of the problem classes strictly because all the relevant factors – N, $R \times T$, and B – have an influence on the computational times of the exact solution. Instead, we regard a problem as a larger problem if it has a larger N while other factors are the same. The parameters associated with the operating speed of the ASC are based on Galle et al. (2018b). The relocation time per container is set to be 120 seconds according to the literature (Zeng et al., 2019) and the terminal practice.

Table 4. 1. Parameters setting for the experiments.

Parameter	Base	Range of scenarios	Fixed parameters
Number of containers (N)		[144, 1296]	
Dimension of bay ($R \times T$)	10×6	[5, 12] × [3, 8]	
Utilisation rate (u)	0.8	{0.67, 0.8, 0.85}	
Distribution of group sizes	U[1,10]	Introduced in corresponding sub-sections	
Relocation time per container			120 seconds
ASC speed:			
Trolley speed with load			1.17 meter/second
Gantry speed with load			1.17 meter/second
Hoisting speed without load			0.93 meter/second

Hoisting speed with load	0.47 meter/second
Container size:	
Container width	2.35 meter
Container length	5.90 meter
Container height	2.39 meter

Regarding the distribution of the group sizes, we consider three scenarios of complete information in which the customer information of all containers is known, and three scenarios of incomplete information in which only the customer information of a part of the containers is known. Details will be given in the corresponding sub-sections. Let V denote the size of a group. In the base instance, the group sizes are uniformly distributed in [1,10], which is represented by V~U[1,10]. For a problem class with the same number of containers and the same distribution of the group sizes, we randomly generated ten instances to vary the size of each group.

4.6.2 Comparison of two formulations of the non-split model

In this section, we verify the superiority of the computational efficiency of the improved formulation over the original formulation. The number of containers N takes the values in the set of {144, 336, 528, 720, 912}, which is sufficient to demonstrate the significant differences between the solution capacity of the two models. Other parameters are the same as the base instance, that is, R = 10, T =6, u = 0.8, and the group sizes follow the uniform distribution U[1,10]. Each problem class includes ten random instances. Both models are solved using CPLEX given a time budget of one hour. The results of the two models are reported in Table 4. 2. "%Opt" reports the percentage of instances solved to optimum within one hour. "Time (s)" reports the average computation time for the instances solved optimally by both models. "#Var" and "#Con" report the average number of variables and constraints for the instances solved optimally by both models, respectively.

As shown in Table 4. 2, both models obtain optimal solutions for all instances in the problem class with 144 containers, but the original model requires a longer computational time. With the increase in the number of containers, the problem becomes more difficult to solve and some instances cannot be solved to optimum within the one-hour time limit. For the problem classes with 336, 528 and 720 containers, the improved model only takes 0.6% to 1.5% of the time taken by the original model. In Table 4. 2, the problem class "912|10×6|19" is the most difficult to solve as shown by that the original model fails to verify the optimum for all the instances within the time limit. For this problem class, the improved model can obtain the optimal solution for 90% of the instances in 679 seconds on average per instance. As can be seen from the "#Var" and "#Con" columns, the improved model substantially reduces the number of variables and reduces the number of constraints a little, and thus the computational efficiency is improved. Given the superiority of the computational efficiency of the improved model, it is used as a benchmark to evaluate the performance of the proposed heuristic algorithm in Section 4.6.3.

Table 4. 2. Results of two formulations of the non-split model.

Problem class	Original formulation [M1]				Improved formulation [M2]			
$N\|R\times T\|B$	%Opt	Time(s)	#Var	#Con	%Opt	Time(s)	#Var	#Con
144|10×6|3	100	11.4	4936	338	100	3.7	598	319
336|10×6|7	60	1252.4	26528	790	100	13.3	1398	735
528|10×6|11	20	1173.7	66550	1244	100	17.2	2200	1151
720|10×6|15	10	2438.7	109184	1680	90	13.6	2984	1567

912|10×6|19*	0	3600	193721	2145	90	679.2	3796	1983

Note. For the problem class "912|10×6|19", the results of [M1] are for all the ten instances, and the results of [M2] are for the nine instances that are solved to optimality.

4.6.3 Comparison of the improved model and the heuristic

To validate the efficiency and effectiveness of the proposed D&C heuristic for solving the non-split model, we compare the performance of the heuristic with that of the improved formulation [M2] solved by CPLEX. CPLEX is given a time limit of one hour and returns the best solutions found so far, i.e., upper bounds when reaching the time limit. We use the default relative MIP gap of CPLEX that is 0.01%, which means CPLEX will stop as soon as it has found a feasible solution proved to be within 0.01% of optimal. Table 4. 3 reports the results of the base instances with a range of batch sizes (N). "LB" and "UB" report the lower bound and upper bound obtained by CPLEX within the time limit, respectively. "Gap_c" reports the average gap (in percentage) between the upper bound (UB) and the lower bound (LB) obtained by CPLEX. The solutions obtained by the heuristic are reported in "Obj". "LB", "UB" and "Obj" are all reported in seconds. "Gap_h" reports the average gap (in percentage) of the solutions obtained by the heuristic against the lower bound obtained by CPLEX. "CPU(s)" reports the average computational time for the ten instances in each problem class.

From Table 4. 3, it can be seen that CPLEX can obtain optimal solutions when the number of containers is small. As the number of containers increases, the solution time of CPLEX increases dramatically. However, the gaps between the lower bound and upper bound are negligible, which indicates that the improved model can provide near-optimal solutions for practically sized problems. As for the heuristic algorithm, it is much more efficient in that it can produce the solutions very close to that of CPLEX within just one second.

Table 4. 3. Comparison between CPLEX and the D&C heuristic for the base instances.

Problem class	CPLEX				Heuristic		
u=0.8	LB	UB	CPU(s)	Gap_c	Obj	CPU(s)	Gap_h
144|10×6|3	8127	8127	3.7	0.004	8179	0.1	0.644
336|10×6|7	21833	21835	15.8	0.010	21912	0.2	0.361
528|10×6|11	37448	37452	46.5	0.010	37479	0.5	0.084
720|10×6|15	56421	56427	638.7	0.010	56525	0.5	0.184
912|10×6|19	81336	81345	971.3	0.010	81523	0.6	0.229
1104|10×6|23	107811	107850	1069.6	0.036	108050	0.7	0.222
1296|10×6|27	137376	137407	1611.3	0.023	137515	0.7	0.101
Average			622.4	0.015		0.4	0.261

Note. Gap_c = (UB - LB)/LB×100%, Gap_h = (Obj - LB)/LB×100%.

In order to show the effectiveness of our heuristic under varying scenarios, we conduct more experiments with varying bay structures and ranges of group sizes. In these experiments, we focus on the problem classes with 1296 containers, which takes the longest solution time by CPLEX as shown in Table 4. 3. The results are reported in Table 4. 4-4.6. The effectiveness of our heuristic is confirmed by the small gaps that are less than 0.6% across all the instances in Table 4. 4-4.6, as shown in the column "Gap_h". Besides, the running times of the heuristic for all these instances are less than six seconds, which confirms its efficiency. The efficiency of the heuristic owes to the decoupling of the decisions of smart stacks and non-smart stacks, and the high solution efficiency of the sub-problems.

Table 4. 4. Comparison between CPLEX and the D&C heuristic for instances with large-scale containers, u=0.8 and V~U[1,10].

Problem class	CPLEX				Heuristic		
u=0.8	LB	UB	CPU(s)	Gap_c	Obj	CPU(s)	Gap_h
1296|10×3|54	169060	169071	1.8	0.007	169077	0.4	0.010
1296|10×4|41	141608	141622	10.8	0.010	141632	0.2	0.017
1296|10×5|33	128170	128182	64.9	0.009	128229	0.2	0.046
1296|10×6|27	137376	137407	1611.3	0.023	137515	0.7	0.101
1296|10×7|24	135920	136124	2749.6	0.150	136194	1.3	0.202
1296|10×8|21	136445	136530	1821.4	0.062	136881	1.8	0.320
1296|6×6|47	181529	181575	1215.2	0.025	182215	0.5	0.378
1296|8×6|35	150179	150233	1204.1	0.036	150470	0.6	0.194
1296|12×6|23	120467	120483	1615.7	0.013	120608	0.6	0.117
Average			1143.8	0.037		0.7	0.154

Note. Gap_c = (UB - LB)/LB×100%, Gap_h = (Obj - LB)/LB×100%.

Table 4. 5. Comparison between CPLEX and the D&C heuristic for instances with large-scale containers, u=0.85 and V~U[1,10].

Problem class	CPLEX				Heuristic		
u=0.85	LB	UB	CPU(s)	Gap_c	Obj	CPU(s)	Gap_h
1296|10×3|52	169060	169071	1.6	0.007	169072	0.3	0.007
1296|10×4|39	144856	144870	74.6	0.010	144918	0.2	0.043
1296|10×5|31	134490	134504	162.3	0.010	134608	0.4	0.088
1296|10×6|26	145167	145280	2761.0	0.078	145418	0.6	0.173
1296|10×7|22	154631	154830	3600.0	0.129	155187	2.3	0.360
1296|10×8|20	145723	145942	3270.3	0.150	146079	2.2	0.244
1296|6×6|44	197361	197528	1460.7	0.085	197995	0.7	0.321
1296|8×6|33	161959	162056	1498.3	0.060	162154	0.6	0.120
1296|12×6|22	128507	128520	494.2	0.010	128548	1.1	0.032
Average			1480.3	0.060		0.9	0.154

Note. Gap_c = (UB - LB)/LB×100%, Gap_h = (Obj - LB)/LB×100%.

Table 4. 6. Comparison between CPLEX and the D&C heuristic for instances with large-scale containers, u=0.85 and V~U[1,5].

Problem class	CPLEX				Heuristic		
u=0.85	LB	UB	CPU(s)	Gap_c	Obj	CPU(s)	Gap_h
1296|10×3|52	183838	183854	7.7	0.009	183854	0.4	0.009
1296|10×4|39	166948	166964	444.0	0.010	167068	0.4	0.072
1296|10×5|31	147366	147380	181.2	0.010	147445	0.5	0.054
1296|10×6|26	164986	165004	2421.2	0.011	165055	1.0	0.042
1296|10×7|22	180942	180998	3600.0	0.031	181886	4.3	0.522
1296|10×8|20	185912	185970	3600.0	0.031	186913	5.7	0.538
1296|6×6|44	226370	226392	103.0	0.010	226525	1.2	0.068
1296|8×6|33	185231	185286	3088.8	0.030	185420	0.9	0.102
1296|12×6|22	147294	147309	143.0	0.010	147398	0.8	0.071

Average	1509.9	0.017		1.7	0.164

Note. Gap_c = (UB - LB)/LB ×100%, Gap_h = (Obj - LB)/LB ×100%.

There is an interesting observation from the comparison of Tables 4.4 and 4.5. The problems with higher utilisation (0.85) are generally more computationally expensive than those with lower utilisation (0.8), although the former problems have fewer bays and thus a smaller number of variables and constraints. One possible reason is that the smart groups are determined in such a way that the groups whose average partition height is larger than T^*u are more likely to be smart groups, which we call "advantageous groups". Therefore, the greater the percentage of advantageous groups, the less the number of nodes that will be explored during the CPLEX branch-and-bound process, as those nodes that do not include the advantageous groups will be cut with a high probability. When the utilisation rate increases, the percentage of advantageous groups decreases, and thus more nodes need to be explored to reach optimality. The heuristic performs robustly, which can be seen by the very similar optimal gaps in columns Gap_h between Table 4. 4 and Table 4. 5.

Overall, the improved model solved by the standard commercial solver CPLEX performs well when the computational efficiency is not a concern. However, in realistic situations, the dynamic changing of the yard status and the uncertainties of the incoming containers may require the terminal operators to make more frequent decisions quickly according to the dynamically updated information. Our proposed heuristic can generate near-optimal solutions with fairly comparable accuracy in substantially reduced time. From the application point of view, our proposed heuristic is more readily available for real-time decision-making because it is able to provide high-quality and robust solutions within very short running times (in seconds).

Given the high solution quality and efficiency of the proposed heuristic, the heuristic will be used to conduct the experiments of sensitivity analysis in Section 4.6.4.

4.6.4 The effectiveness of smart stacking

In this section, we evaluate the effectiveness of smart stacking by comparing the proposed heuristic with the random stacking strategy. The random stacking strategy is commonly used in practice when no information can be used to support stacking decision-making (Dekker et al., 2006). In random stacking, the containers are evenly spread over the stacks in the storage area to reduce the number of relocations. Various instances with different scenarios of group information (Section 4.6.4.1) and yard utilisation rates (Section 4.6.4.2) are tested to investigate their impacts on the performance and the effectiveness of the smart stacking strategy.

4.6.4.1 Impact of group information

We first analyse the impact of the amount of group information and then the impact of the range of group size.

4.6.4.1.1 Impact of the amount of group information

In reality, terminal operators might have incomplete information when only a part of the containers are provided with the information of customer identities or only a part of the customers are willing to participate in the IFF program. To describe the amount of group information, we assume a certain percentage of containers (20%, 50%, and 80%) without group information. For these containers, each one forms a group on its own. For example, "*U*[1, 10]_20%" represents that for 20% of the containers, each one forms a group, while for the remaining 80% containers, their group sizes obey uniform distribution *U*[1,10].

Table 4. 7 presents the results of the scenarios with varying amount of group information. "*U*[1,10]"

represents the scenario of complete information where all the group sizes are generated from the uniform distribution [1,10]. The columns "Obj_1" and column "Obj_2" report the total retrieval time for the smart stacking and random stacking respectively. The columns "Gap(%)" report the relative difference (in percentage) between the two stacking strategies. The results show that the performance of smart stacking measured in total retrieval time depends on the amount of group information. The total retrieval time increases as the amount of information decreases, which is in agreement with intuition. Moreover, the less the amount of information, the smaller the gap is between the smart and random stacking strategies. Obviously, with limited storage space, containers without group information are very unlikely to be allocated to smart stacks (Fig. 4. 6), leading to an increasing percentage of relocation time (Fig. 4. 7), and thus the benefit of smart stacking is decreasing. Nevertheless, smart stacking can still bring an improvement by around 12% compared with random stacking even when only 20% of containers are provided with group information, as shown in the category of *U*[1,10]_80% in Table 4. 7.

Table 4. 7. The results of smart stacking under different amounts of group information.

Problem class	Smart stacking								Random stacking
	U[1,10]		*U*[1,10]_20%		*U*[1,10]_50%		*U*[1,10]_80%		
u=0.8	Obj_1	Gap(%)	Obj_1	Gap(%)	Obj_1	Gap(%)	Obj_1	Gap(%)	Obj_2
144|10×6|3	8179	46.1	9211	39.4	11045	27.3	13367	12.0	15187
336|10×6|7	21912	43.3	24028	37.8	28123	27.2	33961	12.1	38624
528|10×6|11	37479	43.0	41665	36.7	48980	25.6	57969	11.9	65803
720|10×6|15	56525	41.6	62325	35.6	71920	25.6	85304	11.8	96724
912|10×6|19	81523	38.0	87437	33.5	98739	24.9	115914	11.8	131396
1104|10×6|23	108050	36.4	113645	33.1	129424	23.8	149845	11.8	169811
1296|10×6|27	137515	35.1	145424	31.4	162873	23.2	187219	11.7	211967

Note. Gap(%) = (Obj_2- Obj_1)/ Obj_2×100%.

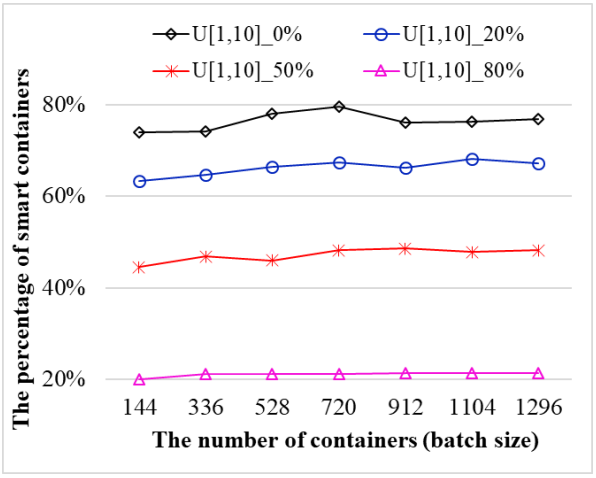


Fig. 4. 6. The percentage of smart containers in smart stacking

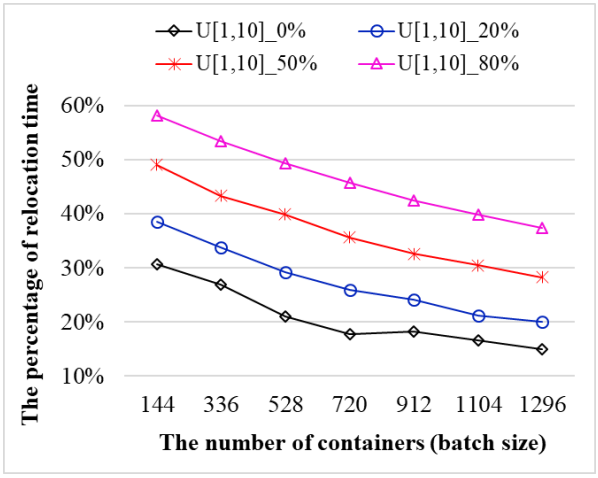


Fig. 4. 7. The percentage of relocation time in the total retrieval time in smart stacking

4.6.4.1.2 Impact of the range of group size

We now analyse the impact of the range of group size on the performance and the effectiveness of smart stacking. Table 4. 8 compares the smart stacking with the random stacking for three scenarios of the range of group size, where the maximum group size is 5, 10, and 20 respectively. We also report the average relocation time per container and the ASC's average travel time per container (in seconds) in smart stacking,

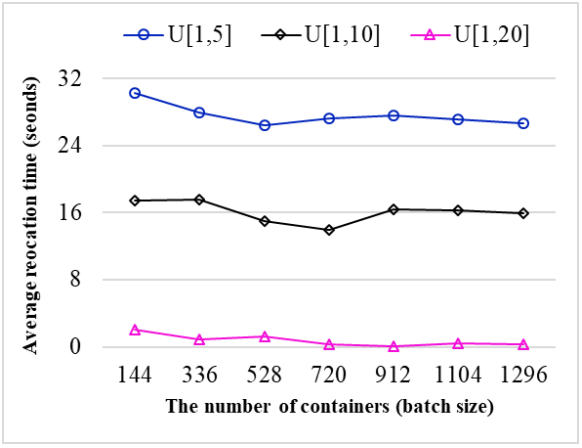
which are depicted in Fig. 4. 8 and Fig. 4. 9, respectively. The performances of smart stacking measured in terms of the total retrieval time (columns Obj_1) and average relocation time (Fig. 4. 8) improve as the range of group size enlarges. This is intuitive because a wider range of group size can bring more full partitions (the partitions whose heights equal *T*) and thus more smart containers. The more interesting observation is that the ASC's average travel time shows a slightly improved trend (Fig. 4. 9) with the enlarging of the group size. This indicates that smart stacking can also help in saving travel time. This observation can be understood from the two properties of the ASC travel time. When the range of group size gets larger, the heights of smart stacks increase, and thus a greater number of higher stacks will occupy the stacks closer to the landside (Property 1). Since a higher slot needs less hoisting time (Property 2) and the bays closer to the landside need less gantry moving time (Property 1), the ASC's total travel time will decrease.

Furthermore, when comparing the results of the smart stacking with the random stacking (columns Gap(%)), the wider the range of group size, the larger the gap is between the smart and the random stacking strategies. This observation can also be explained by the increased percentage of smart containers.

Table 4. 8. The results of smart stacking under different ranges of group size.

Problem class	Smart stacking						Random stacking
	U[1,5]		*U*[1,10]		*U*[1,20]		
u=0.8	Obj_1	Gap(%)	Obj_1	Gap(%)	Obj_1	Gap(%)	Obj_2
144|10×6|3	10171	33.0	8179	46.1	5828	61.6	15187
336|10×6|7	26115	32.4	21912	43.3	15913	58.8	38624
528|10×6|11	45071	31.5	37479	43.0	29460	55.2	65803
720|10×6|15	68832	28.8	56525	41.6	45419	53.0	96724
912|10×6|19	95927	27.0	81523	38.0	64672	50.8	131396
1104|10×6|23	125965	25.8	108050	36.4	87874	48.3	169811
1296|10×6|27	159364	24.8	137515	35.1	113606	46.4	211967

Note. Gap(%) = (Obj_2- Obj_1)/ Obj_2×100%.



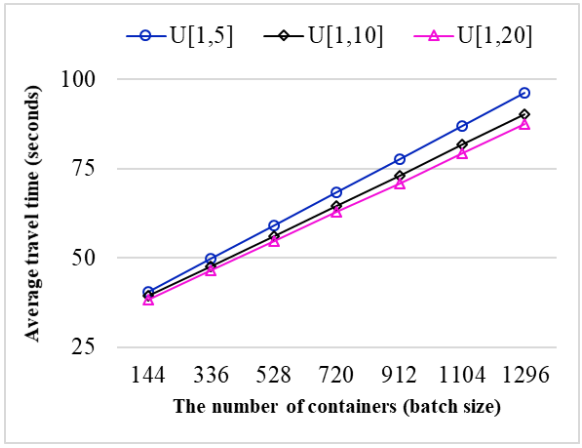


Fig. 4. 8. The average relocation time in smart stacking Fig. 4. 9. The average travel time in smart stacking

4.6.4.2 Impact of utilisation rate

The utilisation rate of the yard storage space varies in terminals and over time. Some terminals have adequate space and may prefer a lower utilisation rate because this reduces the risk of relocation. In contrast, others may prefer to make the best use of the stack height to accommodate more containers (Bruns et al., 2016). Table 4. 9 compares the smart stacking and the random stacking under three utilisation rates.

As the utilisation rate increases, the total retrieval time of smart stacking (column "Obj_1") increases, and the gap between the smart and the random stacking strategies (column "Gap(%)") decreases. This is mainly due to the decreasing percentage of smart containers (column "Sm(%)"). With the increase of the utilisation rate, groups with lower average partition heights are less possible to become smart groups because that will result in wasted space. Thus, the percentage of the smart containers decreases as the utilisation rate increases. As a result, the average relocation time (column "Avg_r") increases. However, the average travel time (column "Avg_t") decreases. This is because the higher the utilisation rate, the higher the average stacking height is, and the fewer the bays near the seaside are occupied. According to the properties of the ASC travel time in Section 4.3.4.1, a higher slot needs less hoisting time, and the bays closer to the landside need less gantry moving time. Therefore, the ASC travel time decreases.

It can be observed that the random stacking strategy and the smart stacking strategy have similar trends of the average relocation time and the average travel time in response to the change of the utilisation rate. However, the trends of the total retrieval time under the two strategies are different. This is because of the trade-off effect of two components. When the utilisation rate increases, under the random stacking, the reduction in the average travel time is sufficient (in most of the cases) to cancel out the increase in the average relocation time, whereas under the smart stacking, the increase in the average relocation time is much more than the reduction in the average travel time.

Table 4. 9. The results of smart stacking under different bay utilisation rates for instances with V~U[1,10].

Utilisation rate	Problem class	Smart stacking				Random stacking			Gap(%)
		Obj_1	Avg_r	Avg_t	Sm(%)	Obj_2	Avg_r	Avg_t	
0.67	912|10×6|**23**	70430	0.5	76.8	95.4	138480	57.2	94.6	49.1
	1104|10×6|**28**	95082	0.3	85.8	96.8	180887	57.0	106.8	47.4
	1296|10×6|**33**	123346	0.6	94.6	94.8	227963	56.9	119.0	45.9
0.80	912|10×6|**19**	81523	16.3	73.1	76.2	131396	63.9	80.2	38.0
	1104|10×6|**23**	108050	16.3	81.6	76.3	169811	63.9	89.9	36.4
	1296|10×6|**27**	137515	15.9	90.2	76.9	211967	63.9	99.6	35.1
0.85	912|10×6|**18**	89850	26.6	71.9	61.3	130817	65.7	77.8	31.3
	1104|10×6|**22**	116051	24.7	80.5	64.1	169709	65.3	88.4	31.6
	1296|10×6|**26**	145418	23.2	89.0	66.2	212156	65.1	98.6	31.5

Note. Gap(%) = (Obj_2- Obj_1)/ Obj_2×100%.

4.6.5 Computational comparison of the non-split model and the split model

In this section, we compare the performances of the two variants of smart stacking strategy to evaluate the benefit of allowing splitting a group between smart stacks and non-smart stacks. Table 4. 10 presents the results of the two models under two scenarios of the range of group size. Both models are solved by CPLEX given a time limit of one hour, and their best objective functions found within the time limit are reported in columns "Obj_1" and "Obj_2" respectively. Columns "Sm(%)" report the average of the percentage of the smart containers for the instances that are solved to optimality by both models so that the two models are comparable by this performance. Columns "Gap(%)" report the relative difference between the objective functions of the two models.

Table 4. 10. Comparison of the two variants of smart stacking models under two scenarios of group size.

Problem class	*U*[1,10]					*U*[1,20]				
	Non-split		Split		Gap	Non-split		Split		Gap
u=0.8	Obj_1	Sm(%)	Obj_2	Sm(%)	(%)	Obj_1	Sm(%)	Obj_2	Sm(%)	(%)
144|10×6|3	8127	74.4	6624	89.7	18.5	5821	96.3	5608	98.0	3.7
336|10×6|7	21835	73.7	18185	90.3	16.7	15900	98.2	15710	98.5	1.2
528|10×6|11	37452	77.6	32765	91.2	12.5	29451	97.9	29122	98.3	1.1
720|10×6|15	56427	83.4	50880	93.1	9.8	45383	99.3	45311	98.8	0.2
912|10×6|19	81345	75.0	72717	90.4	10.6	64620	99.5	64620	98.8	0.0
1104|10×6|23	107850	76.7	97350	90.6	9.7	87834	99.1	87711	98.5	0.1
1296|10×6|27	137407	78.5	125073	91.0	9.0	113552	99.4	113486	98.3	0.1

Note. Gap(%)= (Obj_1 - Obj_2)/ Obj_1 ×100%.

Based on Table 4. 10, we have two observations. First, the superiority of the spit model over the non-split model is verified. However, the benefit of allowing splitting is highly related to the range of group size and is less sensitive to the problem class. For the scenarios of *U*[1,10], the relative difference between the two models is in the range of 9% and 18.5%, which is quite significant. For the scenarios of *U*[1,20], the difference is much smaller (less than 4%). This is because the non-split scenarios of *U*[1,20] have a very high percentage of smart containers, which leaves not much space to improve by allowing splitting.

Second, the reduction in the total retrieval time when allowing splitting (column "Gap(%)") is the result of an increase in the percentage of smart containers (comparing the columns "Sm(%)" of the two models). This is because, in the split model, the smart container decision is not bound to groups but partitions. Such flexibility enables more containers to merit smart stacks. When comparing the columns "Sm(%)" of the two models under the scenarios of *U*[1,20], it is observed that in the last four problem classes, the percentage of smart containers for the split model is even smaller than that of the non-split model. This counterintuitive phenomenon can be explained as follows. The experiments' results show that there exist some stacks that are assigned with only one container. For these stacks, they can be regarded either as non-smart stacks by h_{br}=1 or as smart stacks by x^1_{br}=1, which essentially leads to multiple optimal solutions with different "Sm(%)" performances.

4.7 Conclusions

This paper considers the Storage Location Assignment Problem (SLAP) for import containers at an automated container terminal. We proposed a new concept, the Smart Stacking (SS) strategy, which is motivated by a practical container delivery program, import free flow (IFF), aiming at eliminating the need for relocations and realising rapid retrieval flow. Under the SS strategy, the free-flow (smart) containers of a customer are stored at dedicated stacks to guarantee zero relocation when retrieving them, while the non-free-flow (non-smart) containers are sharing stacks. We focus on the offline operational-level decision making, in which a batch of import containers are to be assigned to exact slots in a given storage area in a single block to minimise the total retrieval time that is the sum of the relocation time and the crane travel time.

Two policies, non-split policy and split policy, are proposed under the SS strategy according to whether a single customer's containers are allowed to be split between smart stacks and non-smart stacks. Mixed-integer programming models are developed under each policy, which are compared both theoretically and computationally. For the non-split policy, we develop two formulations: original and

improved formulations. The improved formulation utilises the structural properties of the optimal solution to enhance the representation of certain variables. As a result, the number of variables can be significantly reduced and the solution efficiency is improved. To further improve the solution efficiency, a divide-and-conquer heuristic algorithm is designed that can solve the non-split model in several seconds achieving within 0.6% gap from the optimal solution for the experimented instances that are of realistic problem sizes. For the split policy, we develop a MIP model assuming that all smart piles must be selected from the optimal partitions of the non-split model. The objective of the split model is proved to be not greater than the objective of the non-split variant. Extensive computational experiments demonstrate the effectiveness of the proposed SS strategy, models and algorithm.

This paper is the first to propose the concept of smart stacking that enables the terminal operator to incorporate the customer information to optimise import container storage and allocation so that the total retrieval time can be minimised. The proposed SS strategy and the SLAPs under consideration contribute to the literature of container stacking problems for import containers, and more importantly bring about novelty in developing new thinking and research opportunities in this field. On the practical side, this paper produces useful managerial implications. Firstly, the SS strategy can significantly reduce the total retrieval time compared with the traditional practice. The effectiveness of the SS strategy is more significant in the following situations: a larger amount of customer information, a wider range of group sizes, and a lower utilisation rate. It is thus recommended that terminal operators should seek collaboration with the customers to access the customer information of the import containers so that port congestion can be mitigated in the retrieval processes, especially with those high-volume customers. Secondly, the proposed heuristic algorithm is fast enough to produce high-quality solutions, which can be applied in practice to support the operational-level decision-making of free-flow containers, non-free-flow containers, and their stacking positions. Thirdly, the retrieval time could be further reduced if the same customers' containers are allowed to be split into smart and non-smart in certain conditions. The proposed models can help terminal operators evaluate the trade-off between the additional benefit and the extra cost of negotiating with customers when adopting the split policy.

Further research may be conducted in the following directions. Firstly, the split policy deserves more research, e.g. investigating the optimality of the split variant and examining the relationship between the characteristics of a specific problem environment (e.g., the distribution of customer volumes/group sizes, the maximum stacking height and the utilisation rate) and the reduction in retrieval time from the split variant. Secondly, more information about containers could be incorporated into our models such as container retrieval time and customer priority. By incorporating predictive retrieval times or advanced appointments (e.g. via vehicle booking system/truck appointment system), the smart stacking strategy can be further enhanced and there are more opportunities to generate benefit and save storage space because containers of different customers can share stacks according to their retrieval times. It is therefore interesting to incorporate the temporal information of containers into the models and evaluate its value. Thirdly, the SLAP studied in this paper is oriented toward a short-term operational problem. It is interesting to study the long-term storage space allocation problems that concern the storage and allocation of import containers from multiple vessels to different blocks in the yard over a planning horizon. The models and solution method developed in this paper will function as a building block for this higher-level decision. Fourthly, another meaningful research direction is to apply the smart stacking strategy at rail-water intermodal container terminals. Note that a single rail carrier could have hundreds of containers from each vessel and these containers may go for the same rail head; this makes such intermodal terminals suitable for smart stacking.

4.8 Appendix

Appendix A. Technical proofs

Property 1. Let l^x and l^y be the length and width of a 20-ft (twenty-foot equivalent unit) standard container, respectively. Let v^x be the ASC gantry moving speed with load and v^y be the ASC trolley moving speed with load. For a block with R rows and M bays, if there exists a bay $\hat{b}$ ($1 \le \hat{b} \le M$) that satisfies $\frac{\hat{b} \cdot l^x}{v^x} \ge \frac{R/2 \cdot l^y}{v^y} > \frac{b \cdot l^x}{v^x}$ ($1 \le b < \hat{b} \le M$), then $T_{br} = T_b$ for $1 \le \hat{b} \le b \le M$ and $1 \le r \le R$, and $T_{br} > T_{b'r}$ for $\hat{b} \le b' < b \le M$ and $1 \le r \le R$.

Proof. This proof is straightforward. It suffices to identify the condition (i.e., bay $\hat{b}$) under which the gantry moving time is not less than the trolley moving time when retrieving a container from a slot to the landside transfer point. Let (b, r) be the stack where the requested container is stored. $T_{br} = \max\{T_b, T_r\}$ is calculated by $T_{br} = \max\left\{\frac{b \cdot l^x}{v^x}, \frac{|r - R/2| \cdot l^y}{v^y}\right\}$, $1 \le b \le M$ and $1 \le r \le R$. For all the stacks r ($1 \le r \le R$) in bay b, they have the same gantry moving time, which is calculated by $T_b = \frac{b \cdot l^x}{v^x}$, whereas their trolley moving times depend on the stacks r, which are calculated by $T_r = \frac{|r - R/2| \cdot l^y}{v^y}$. $\max_{1 \le r \le R} T_r = \frac{|R - R/2| \cdot l^y}{v^y}$. Therefore, for any b ($1 \le b \le M$), if $\frac{b \cdot l^x}{v^x} \ge \frac{R/2 \cdot l^y}{v^y}$, then $T_{br} = \frac{b \cdot l^x}{v^x} = T_b$, $1 \le r \le R$. Suppose there exists a minimum bay $\hat{b}$ that satisfies $\frac{\hat{b} \cdot l^x}{v^x} \ge \frac{R/2 \cdot l^y}{v^y} > \frac{b \cdot l^x}{v^x}$ ($1 \le b < \hat{b} \le M$). For any b ($\hat{b} < b \le M$), since $\frac{b \cdot l^x}{v^x} > \frac{\hat{b} \cdot l^x}{v^x}$, $T_{br} = \frac{b \cdot l^x}{v^x}$ still holds. Therefore, we have $T_{br} = T_b$ for $1 \le \hat{b} \le b \le M$ and $1 \le r \le R$. In addition, for any two b and b' ($\hat{b} \le b' < b \le M$), since $\frac{b \cdot l^x}{v^x} > \frac{b' \cdot l^x}{v^x}$, we have $T_{br} > T_{b'r}$. This completes the proof. □

Property 2. $T_t < T_{t'}$ for $1 \le t' < t \le T$.

Proof. This proof is straightforward. Let l^z be the height of a 20-ft standard container. Let $v^{z,E}$ and $v^{z,L}$ be the speed of the ASC lowering its spreader without a container and hoist its spreader with a container, respectively. Then, $T_t = T_t^E + T_t^L$ is calculated by $T_t = \frac{(T+1-t) \cdot l^z}{v^{z,E}} + \frac{(T+1-t) \cdot l^z}{v^{z,L}}$. Therefore, we have $T_t < T_{t'}$ for $1 \le t' < t \le T$, which completes the proof. □

Proposition 1. Let h_{p_i} be the height of pile p_i. In the optimal solution to M1, for any two smart piles p_i and p_j which are located at stacks (b_i, r_i) and (b_j, r_j) respectively, if $h_{p_i} > h_{p_j}$, then $T_{b_i r_i} \le T_{b_j r_j}$.

Proof. We suppose by contradiction that there exists an optimal solution σ^* such that $h_{p_i} > h_{p_j}$ and $T_{b_i r_i} > T_{b_j r_j}$. We construct a σ^* where p_i is located at stack $(b_i, r_i) = (b_i^*, r_i^*)$ and p_j is located at stack $(b_j, r_j) = (b_j^*, r_j^*)$ with $h_{p_i} > h_{p_j}$ and $T_{b_i r_i} > T_{b_j r_j}$. Let us consider only the stacks that accommodate p_i and p_j. Then, we can construct a feasible solution σ' by swapping the positions of p_i and p_j in σ^*. As a result, in σ', p_i is located at stack (b_j^*, r_j^*) and p_j is located at stack (b_i^*, r_i^*), which means $T_{b_i r_i} < T_{b_j r_j}$.

Now we consider the total times t^* and t' for retrieving the containers in p_i and p_j under σ^* and σ', respectively. t^* and t' can be calculated by Eqs. (4.A.1) and (4.A.2), respectively.

$$t^* = T_{b_i^* r_i^*} \cdot h_{p_i} + \sum_{t=1}^{h_{p_i}} T_t + T_{b_j^* r_j^*} \cdot h_{p_j} + \sum_{t=1}^{h_{p_j}} T_t \tag{4.A.1}$$

$$t' = T_{b_j^* r_j^*} \cdot h_{p_i} + \sum_{t=1}^{h_{p_i}} T_t + T_{b_i^* r_i^*} \cdot h_{p_j} + \sum_{t=1}^{h_{p_j}} T_t \tag{4.A.2}$$

By Eqs. (4.A.1) and (4.A.2), we have $t^* - t' = \left(T_{b_i^* r_i^*} - T_{b_j^* r_j^*}\right) \cdot \left(h_{p_i} - h_{p_j}\right)$. Since $h_{p_i} > h_{p_j}$ and $T_{b_i^* r_i^*} > T_{b_j^* r_j^*}$, we have $\left(T_{b_i^* r_i^*} - T_{b_j^* r_j^*}\right) \cdot \left(h_{p_i} - h_{p_j}\right) > 0$ and thus $t^* > t'$. Since σ^* and σ' differ only in the positions of the stacks that accommodate p_i and p_j, the total times for retrieving the containers in other piles are the same. Therefore, $t^* > t'$ leads to a contradiction that σ^* is an optimal solution and concludes the proof. □

Proposition 2. Let P_c be the number of piles of a smart group c in the optimal solution, then $P_c = \lceil v_c / T \rceil$, and the P_c piles are composed of P_c -1 full piles and one pile whose height equals $v_c - (P_c - 1) \cdot T$.

Proof. Let us rank the P_c piles of smart group c in ascending order of the ASC's horizontal travel time needed to retrieve a container from the stack where the pile is located. According to Proposition 1, the P_c piles can be ordered as $p_1, p_2, ..., p_{P_c}$ with $h_{p_1} \geq h_{p_2} \geq .... \geq h_{p_{P_c}}$ and $T_{b_1 r_1} \leq T_{b_2 r_2} \leq .... \leq T_{b_{P_c} r_{P_c}}$. There are two cases depending on the the size of v_c : #1) $v_c \geq T$; #2) $v_c < T$.

In the case of $v_c \geq T$, we suppose by contradiction that in the optimal solution σ^* there is at least one non-full pile in the first P_c -1 piles. Let p_i be the highest non-full pile in the first P_c -1 piles. We can construct a feasible solution σ' that differs σ^* only with regards to piles p_i and p_{P_c} of group c such that the height of p_i is increased by one and the height of pile p_{P_c} is decreased by one. In σ', the piles corresponding to p_i and p_{P_c} are denoted by p_i' and p_{P_c}', and $h_{p_i'} = h_{p_i} + 1$ and $h_{p_{P_c}'} = h_{p_{P_c}} - 1$.

Let $f(\sigma^*)$ and $f(\sigma')$ be the objective function of σ^* and σ', respectively. Next, we show that $f(\sigma^*)-f(\sigma')>0$. Let t^* denote the total times for retrieving the containers in piles p_i and p_{P_c} under σ^*, and t' denote the total times for retrieving the containers in p_i' and p_{P_c}' under σ'. Since σ^* and σ' differ only in the heights of p_i and p_i', and the heights of p_{P_c} and p_{P_c}', the total times for retrieving the containers in other piles are the same. Hence, $f(\sigma^*)-f(\sigma')$ equals t^*-t'. Now, we calculate t^* and t' by Eqs. (4.A.3) and (4.A.4), respectively.

$$t^* = T_{b_i r_i}\cdot h_{p_i} + \sum_{t=1}^{h_{p_i}} T_t + T_{b_{P_c} r_{P_c}}\cdot h_{p_{P_c}} + \sum_{t=1}^{h_{p_{P_c}}} T_t \tag{4.A.3}$$

$$t' = T_{b_i r_i}\cdot h_{p_i'} + \sum_{t=1}^{h_{p_i'}} T_t + T_{b_{P_c} r_{P_c}}\cdot h_{p_{P_c}'} + \sum_{t=1}^{h_{p_{P_c}'}} T_t \tag{4.A.4}$$

By Eqs. (4.A.3) and (4.A.4), we have $t^*-t' = \left(T_{b_{P_c} r_{P_c}} - T_{b_i r_i}\right) + \left(T_{h_{p_{P_c}}} - T_{h_{p_i}+1}\right)$. For the first term, since $T_{b_i r_i} \le T_{b_{P_c} r_{P_c}}$, we have $\left(T_{b_{P_c} r_{P_c}} - T_{b_i r_i}\right) \ge 0$. For the second term, since $h_{p_{P_c}} \le h_{p_i}$, we have $h_{p_{P_c}} < h_{p_i}+1$. By Property 2, we have $T_{h_{p_{P_c}}} > T_{h_{p_i}+1}$. Therefore, $t^*-t'>0$, which leads to a contradiction that σ^* is an optimal solution.

In the case of $v_c < T$, we suppose by contradiction that in the optimal solution σ^*, $h_{P_1} < v_c$. We can construct a feasible solution σ' that differs σ^* only with regards to piles p_1 and p_{P_c} of group c such that the height of p_1 is increased by one and the height of pile p_{P_c} is decreased by one. Same as the first case, we can prove that $t^*-t'>0$, which leads to a contradiction that σ^* is an optimal solution.

Therefore, in the optimal solution, in the case of $v_c \ge T$, all the first P_c -1 piles must be full piles; and in the case of $v_c \ge T$, there must be only one pile whose height equals v_c. This means that in both cases, $P_c = \lceil v_c / T \rceil$, and the P_c piles are composed of P_c -1 full piles and one pile whose height equals $v_c - (P_c - 1)\cdot T$, which completes the proof. □

Proposition 3. [M1] and [M2] are equivalent problems.

Proof. We first show an implied constraint for [M1]. By the definition of u_{ct}, we have the following implied constraints for [M1] which ensures that if group c is smart, it will be allocated to u_{ct} number of smart stacks for storing its partitions with t number of containers.

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{ct} = s_c \cdot u_{ct}, \quad \forall c \in \{1,...,C\}, \quad \forall t \in \{1,...,T\} \tag{4.A.5}$$

We now show that Constraints (4.4) are redundant with Constraints (4.A.5).

By summing both sides of Eq. (4.A.5) by the domain of *t*, we have

$$\sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{c=1}^{T} x_{br}^{ct}\cdot t = s_c\sum_{t=1}^{T} u_{ct}\cdot t\,,\quad \forall c\in\{1,...,C\} \tag{4.A.6}$$

By the definition of u_{ct}, we have $\sum_{t=1}^{T} u_{ct}\cdot t = v_c$. Then, by replacing $\sum_{t=1}^{T} u_{ct}\cdot t$ in (4.A.6) by v_c, we can obtain Constraints (4.4): $\sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{ct}\cdot t = s_c\cdot v_c\,,\quad \forall c\in\{1,...,C\}$.

Therefore, [M1] is equivalent to the following formulation defined by [M1-1].

[**M1-1**] $$\min \sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta} w_b\cdot\bar{T} \tag{4.2}$$

s.t. Constraints (4.3), (4.A.5), and (4.5)-(4.19)

Next, we reformulate [M1-1] into an equivalent problem denoted by [M1-2].

We define variables x_{br}^{t} by

$$x_{br}^{t} = \sum_{c=1}^{C} x_{br}^{ct}\,,\quad \forall b\in\Theta\,,\quad \forall r\in\{1,...,R\}\,,\quad \forall t\in\{1,...,T\} \tag{4.A.7}$$

Then, we replace all $\sum_{c=1}^{C} x_{br}^{ct}$ in the constraints of [M1-1] by x_{br}^{t}, after which [M1-1] is re-formulated as the following equivalent problem defined by [M1-2].

[**M1-2**] $$\min \sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta} w_b\cdot\bar{T} \tag{4.2}$$

s.t. Constraints (4.11) – (4.18), (4.20), (4.22) – (4.28), (4.A.5), and (4.A.7)

Recall that [M2] is formulated as

[**M2**] $$\min \sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta} w_b\cdot\bar{T} \tag{4.2}$$

s.t. Constraints (4.11) – (4.18) and (4.20) – (4.28)

We now prove that [M2] is equivalent to [M1-2] by showing that (i) any optimal solution of [M1-2] can be transferred to a feasible solution of [M2] (i.e., $f(\text{M1-2}) \geq f(\text{M2})$); and (ii) any optimal solution of [M2] can be transferred to a feasible solution of [M1-2] (i.e., $f(\text{M1-2}) \leq f(\text{M2})$), and thus $f(\text{M1-2}) = f(\text{M2})$.

(i) Any optimal solution of [M1-2] can be transferred to a feasible solution of [M2].

We first show that by combining Constraints (4.A.5), and (4.A.7), we can derive Constraints (4.21).

By summing both sides of Eq. (4.A.5) by the domain of *c*, we get

$$\sum_{b\in\Theta}\sum_{r=1}^{R}\sum_{c=1}^{C} x_{br}^{ct} = \sum_{c=1}^{C} s_c\cdot u_{ct}\,,\quad \forall t\in\{1,...,T\} \tag{4.A.8}$$

Using Eq. (4.A.7), we replace $\sum_{c=1}^{C} x_{br}^{ct}$ in (4.A.8) by x_{br}^{t}, and then we obtain Constraints (4.21): $\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{t} = \sum_{c=1}^{C} s_c u_{ct}\,,\quad \forall t\in\{1,...,T\}$.

We denote $\boldsymbol{\mathcal{X}}$ by the polyhedron defined by Constraints (4.11) – (4.18), (4.20), (4.22) – (4.28), (4.A.5) and (4.A.7), and $\boldsymbol{\mathcal{Y}}$ by the polyhedron defined by Constraints (4.11) – (4.18) and (4.20) – (4.28). Since Constraints (4.A.5) and (4.A.7) can derive Constraints (4.21), $\boldsymbol{\mathcal{Y}}$ contains $\boldsymbol{\mathcal{X}}$. Since the feasible set of [M1-2] is defined by $\boldsymbol{\mathcal{X}}$ and the feasible set of [M2] is defined by $\boldsymbol{\mathcal{Y}}$, we have $f(\text{M1-2}) \geq f(\text{M2})$. Given any optimal solution of [M1-2], (s_c , x_{br}^{ct}, x_{br}^{t}, z_b , h_{br}, w_b, f_{brt}, y_{brt}), we can transfer it to a solution of [M2], (s_c , x_{br}^{t}, z_b , h_{br}, w_b, f_{brt}, y_{brt}).

(ii) Any optimal solution of [M2] can be transferred to a feasible solution of [M1-2].

Given any optimal solution of [M2], (s_c , x_{br}^t , z_b , h_{br} , w_b , f_{brt} , y_{brt}), we show that we can construct a solution of [M1-2], (s_c , x_{br}^{ct} , x_{br}^t , z_b , h_{br} , w_b , f_{brt} , y_{brt}). Since the set of variables of x_{br}^t , z_b , h_{br} , w_b , f_{brt} and y_{brt} verify the Constraints (4.11) – (4.18) and (4.20) – (4.28), they also satisfy the Constraints (4.11) – (4.18), (4.20) and (4.22) – (4.28) in [M1-2]. Therefore, the remaining task is to prove that the solution of s_c and x_{br}^t satisfy the Constraints (4.A.5) and (4.A.7) in [M1-2]. In the following, given the solutions of s_c and x_{br}^t in [M2], we construct a feasible solution x_{br}^{ct} that satisfies the Constraints (4.A.5) and (4.A.7) in [M1-2].

Let *G* denote the set of groups with $s_c = 1$ and G' the set of groups with $s_c = 0$. Then, according to Constraints (4.21), we have Eq. (4.A.9)

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t = \sum_{c\in G} 1\cdot u_{ct} + \sum_{c\in G'} 0\cdot u_{ct} \ , \quad \forall t \in \{1,...,T\} \tag{4.A.9}$$

Let P_t be the set of stacks (*b*, *r*) that satisfies $x_{br}^t = 1$, then we have $|P_t| = \sum_{c\in G} u_{ct}$, and Eq. (4.A.9) can be written by

$$\sum_{(b,r)\in P_t} x_{br}^t = \sum_{c\in G} u_{ct} \ , \quad \forall t \in \{1,...,T\} \tag{4.A.10}$$

We construct the relationship between x_{br}^{ct} and x_{br}^t by Eq. (4.A.11), which correspond to Constraints (4.A.7).

$$\sum_{c=1}^{C} x_{br}^{ct} = \sum_{c\in G} x_{br}^{ct} + \sum_{c\in G'} x_{br}^{ct} = x_{br}^t \ , \quad \forall t \in \{1,...,T\} \ , \quad \forall (b,r) \in P_t \tag{4.A.11-1}$$

$$\sum_{c=1}^{C} x_{br}^{ct} = 0 \ , \quad \forall t \in \{1,...,T\} \ , \quad \forall (b,r) \notin P_t \tag{4.A.11-2}$$

We construct Eq. (4.A.12) which corresponds to the case of $\forall c \in G'$ in Constraints (4.A.5).

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^{ct} = 0 \ , \quad \forall c \in G' \ , \quad \forall t \in \{1,...,T\} \tag{4.A.12}$$

By Eq. (4.A.12) and Eq. (4.A.11-1) we have

$$\sum_{c\in G} x_{br}^{ct} = x_{br}^t \ , \quad \forall t \in \{1,...,T\} \ , \quad \forall (b,r) \in P_t \tag{4.A.13}$$

By summing (4.A.13) by the domain of (*b*, *r*), we get

$$\sum_{(b,r)\in P_t}\sum_{c\in G} x_{br}^{ct} = \sum_{(b,r)\in P_t} x_{br}^t \ , \quad \forall t \in \{1,...,T\} \tag{4.A.14}$$

Combining Eqs. (4.A.10) and (4.A.14), we obtain

$$\sum_{(b,r)\in P_t}\sum_{c\in G} x_{br}^{ct} = \sum_{c\in G} u_{ct} \ , \quad \forall t \in \{1,...,T\} \tag{4.A.15}$$

We divide P_t into $|G|$ sub-sets such that $|P_{ct}| = u_{ct}$, $\forall c \in G$, and $P_t = \bigcup_{c\in G} P_{tc}$. Then, combing Eq. (4.A.15) and the definition of P_{tc} , we get

$$\sum_{(b,r)\in P_{tc}} x_{br}^{ct} = u_{ct} \ , \quad \forall c \in G \ , \quad \forall t \in \{1,...,T\} \tag{4.A.16}$$

Eqs. (4.A.12) and (4.A.16) together construct a feasible solution corresponding to Constraints (4.A.5). Therefore, Eqs. (4.A.11), (4.A.12) and (4.A.16) together provide a feasible solution that satisfies the Constraints (4.A.5) and (4.A.7) in [M1-2]. By now, we have transferred any optimal solution of [M2] to a feasible solution of [M1-2], and thus we have $f(\text{M1-2}) \le f(\text{M2})$. Since $f(\text{M1-2}) \ge f(\text{M2})$ as proved in (i),

we have $f(\text{M1-2}) = f(\text{M2})$.

Therefore, we have proved that [M1-2] and [M2] are equivalent. Since [M1] and [M1-2] are equivalent, the equivalence between [M1] and [M2] follows, which completes the proof. □

Lemma 1. Let $f(\text{M2})$ and $f(\text{M3})$ be the objective functions of the optimisation problems defined by [M2] and [M3], respectively, then $f(\text{M3}) \le f(\text{M2})$.

Proof. In order to prove $f(\text{M3}) \le f(\text{M2})$, we show that any optimal solution of [M2] is a feasible solution to [M3]. It is observed that [M2] and [M3] share the same Constraints (4.11) – (4.18), (4.20), and (4.22) – (4.28). In addition, [M2] has one more Constraints (4.21), and [M3] has one more Constraints (4.29). Therefore, the remaining task is to prove that any solution x_{br}^t in [M2] satisfies Constraints (4.29).

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t = \sum_{c=1}^{C} s_c u_{ct}, \quad \forall t \in \{1,...,T\} \tag{4.21}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t \le n_t, \quad \forall t \in \{1,...,T\} \tag{4.29}$$

Since $\sum_{c=1}^{C} s_c u_{ct} \le \sum_{c=1}^{C} u_{ct} = n_t$, from Constraints (4.21), we can derive Constraints (4.29) by $\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t = \sum_{c=1}^{C} s_c u_{ct} \le n_t$, $\forall t \in \{1,...,T\}$. Thus, any solution x_{br}^t in [M2] satisfy Constraints (4.29). Therefore, any optimal solution of [M2] is a feasible solution to [M3], that is, $f(\text{M3}) \le f(\text{M2})$, which completes the proof. □

Lemma 2. Let $V^m = \max_{c\in\{1,...,C\}} \{v_c\}$, then $f(\text{M2}) = f(\text{M3})$ when $V^m \le T$.

Proof. In order to prove Lemma 2, we only need to prove that any optimal solution of [M3] is a feasible solution to [M2] when $V^m \le T$ and thus $f(\text{M3}) \ge f(\text{M2})$ when $V^m \le T$. Since $f(\text{M3}) \le f(\text{M2})$ by Lemma 1, $f(\text{M2}) = f(\text{M3})$ when $V^m \le T$ follows.

When $V^m \le T$, according to the definition of optimal partition, any group *c* has only one partition in which the number of containers equals its group size v_c. Thus, u_{ct}, $c \in \{1,...,C\}$, $t \in \{1,...,T\}$, is defined by

$$u_{ct} = \begin{cases} 1, t = v_c \\ 0, t = \{1,...,T\} / v_c \end{cases}$$

We now divide all the groups into *T* mutually exclusive and collectively exhaustive sub-sets G_t, $t \in \{1,...,T\}$, such that G_t be the set of groups with $v_c = t$ and $\sum_{t=1}^{T} |G_t| = C$. Then, Constraints (4.21) in [M2] and Constraints (4.29) in [M3] can be re-written as follows

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t = \sum_{c\in G_t} s_c \cdot u_{ct}, \quad \forall t \in \{1,...,T\} \tag{4.21′}$$

$$\sum_{b\in\Theta}\sum_{r=1}^{R} x_{br}^t \le \sum_{c\in G_t} u_{ct}, \quad \forall t \in \{1,...,T\} \tag{4.29′}$$

The remaining task is to prove that any optimal solution x_{br}^t of [M3] satisfies Constraints (4.21′). Suppose in the optimal solution of [M3], $\sum_{b\in\Theta}\sum_{r=1}^{R} \tilde{x}_{br}^t \le \sum_{c\in G_t} u_{ct} = X_t$, $t \in \{1,...,T\}$. By the definition of G_t

and u_{ct}, we have $X_t \le |G_t|$. Next, we show that we can construct $\tilde{s}_c$ that satisfies $\sum_{b\in\Theta}\sum_{r=1}^{R}\tilde{x}_{br}^{t} = \sum_{c\in G_t}\tilde{s}_c \cdot u_{ct}$, $\forall t \in \{1,...,T\}$. Since G_t, $t \in \{1,...,T\}$, are mutually exclusive and $X_t \le |G_t|$, we can always find a subset of groups from G_t, denoted by G_t^*, such that $|G_t^*| = X_t$ and G_t^*, $t \in \{1,...,T\}$, are mutually exclusive. Then, we can construct a feasible $\tilde{s}_c$ by letting $\tilde{s}_c = 1$, $c \in G_t^*$, $t \in \{1,...,T\}$ and $\tilde{s}_c = 0$, $c \notin G_t^*$, $t \in \{1,...,T\}$.

By now, we have proved that any optimal solution of [M3] is a feasible solution of [M2] when $V^m \le T$, which indicates $f(\text{M3}) \ge f(\text{M2})$ when $V^m \le T$. □

Appendix B. Details of the heuristic

B.1. The integer programming model for Subproblem 1

[**Sub1**]
$$\max \sum_{c=1}^{C} s_c \cdot v_c \tag{4.B.1}$$

s.t.
$$\sum_{t=1}^{T} x_{br}^{t} \le 1, \quad \forall b \in \Theta_i, \quad \forall r \in \{1,...,R\} \tag{4.B.2}$$
$$\sum_{b\in\Theta_i}\sum_{r=1}^{R} x_{br}^{t} = \sum_{c=1}^{C} s_c u_{ct}, \quad \forall t \in \{1,...,T\} \tag{4.B.3}$$
$$z_b \le R - \sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{t}, \quad \forall b \in \Theta_i \tag{4.B.4}$$
$$z_b \cdot R \ge R - \sum_{r=1}^{R}\sum_{t=1}^{T} x_{br}^{t}, \quad \forall b \in \Theta_i \tag{4.B.5}$$
$$z_b \in \{0,1\}, \quad \forall b \in \Theta_i \tag{4.B.6}$$
$$s_c \in \{0,1\}, \quad \forall c \in \{1,...,C\} \tag{4.B.7}$$
$$x_{br}^{t} \in \{0,1\}, \quad \forall b \in \Theta_i, \forall r \in \{1,...,R\}, \forall t \in \{1,...,T\} \tag{4.B.8}$$

B.2. The heuristic rule for solving Subproblem 1

The heuristic rule used for determining the smart groups is described below. The following notations are defined.

g_c: the group at the cth position in the sorting list.

N_s: the number of smart containers.

N_n: the number of non-smart containers.

S_s: the number of stacks for storing smart containers.

S_n: the number of stacks for storing non-smart containers.

Cap: the capacity of the remaining stacks for storing non-smart containers.

Step 0: Calculate $d_c = v_c / P_c$ for each group. Sort the groups according to d_c in descending order. Ties are broken by sorting them according to P_c in descending order. Initialize, $c = 0$; $N_s = 0$; $N_n = N$; $S_s = 0$; $S_n = B_i \cdot R$.

Step 1: c = c+1. If $c \le C$, calculate the capacity of the remaining stacks for storing non-smart containers *Cap* by Eq. (4.B.9); otherwise, stop.

$$Cap = \begin{cases} \left(S_n - P_{g_c}\right)/R \cdot \left((R-1)\cdot T+1\right), \text{ if } \left(S_n - P_{g_c}\right)\%R = 0; \\ \left(S_n - P_{g_c}\right)/R \cdot \left((R-1)\cdot T+1\right) + \left(\left(S_n - P_{g_c}\right)\%R - 1\right)\cdot T + 1, \text{ otherwise} \end{cases} \quad (4.B.9)$$

Step 2: If $Cap \ge N_n - v_{g_c}$, determine g_c to be a smart group, update S_s = S_s + P_c, S_n = S_n - P_c, $N_s = N_s + v_{g_c}$, and $N_n = N_n - v_{g_c}$, and then go to Step 1; otherwise, go to Step 1.

In the above procedures, Steps 1 and 2 are to maintain the feasibility of Constraints (4.22) and (4.25). In considering whether a group can be selected as a smart group, Step 1 first calculates the remaining capacity assuming that the group has been determined to be a smart group, and then Step 2 judges whether this capacity is enough to store the remaining non-smart containers. If the remaining capacity is not enough, the group under consideration is skipped and we move on to examine the next group in the sorting list.

B.3. The MIP model for Subproblem 2

The objective function of subproblem 2 is as follows:

$$\min \sum_{b\in\Theta_i^2}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt}\cdot y_{brt} + \sum_{b\in\Theta_i^2} w_b \cdot \overline{T} \quad (4.B.10)$$

The following constraints need to be satisfied:

$$\sum_{b\in\Theta_i^2}\sum_{r=1}^{R} h_{br} = N_n \quad (4.B.11)$$

$$\sum_{r=1}^{R} h_{br} \le RT - (T-1), \quad \forall b \in \Theta_i^2 \quad (4.B.12)$$

$$\sum_{t=1}^{T} y_{brt} = h_{br}, \quad \forall b \in \Theta_i^2, \quad \forall r \in \{1,...,R\} \quad (4.B.13)$$

$$y_{brt} \le y_{br,t-1}, \quad \forall b \in \Theta_i^2, \quad \forall r \in \{1,...,R\}, \forall t \in \{2...,T\} \quad (4.B.14)$$

$$w_b = \sum_{r=1}^{R}\sum_{t=1}^{T} \alpha_t \cdot f_{brt}, \quad \forall b \in \Theta_i^2 \quad (4.B.15)$$

$$\sum_{t=1}^{T} f_{brt} \le 1, \quad \forall b \in \Theta_i^2, \quad \forall r \in \{1,...,R\} \quad (4.B.16)$$

$$\sum_{t=1}^{T} t \cdot f_{brt} = h_{br}, \quad \forall b \in \Theta_i^2, \quad \forall r \in \{1,...,R\} \quad (4.B.17)$$

$$y_{brt} \in \{0,1\}, \quad f_{brt} \in \{0,1\}, \quad \forall b \in \Theta_i^2, \forall r \in \{1,...,R\}, \forall t \in \{1,...,T\} \quad (4.B.18)$$

$$h_{br} \in \{1,...,T\}, \quad \forall b \in \Theta_i^2, \forall r \in \{1,...,R\} \quad (4.B.19)$$

$$w_b \ge 0, \quad \forall b \in \Theta_i^2 \quad (4.B.20)$$

If $\hat{R} > 0$, the $\overline{B}$ th bay will be a mixed bay and we have the following constraints for this bay:

$$h_{\overline{B}r} = 0, \quad \forall r \in \left\{\hat{R}+1,...,R\right\} \quad (4.B.21)$$

Then, subproblem 2 can be formulated as follows:

[Sub2-1] if $\hat{R} > 0$:

min (4.B.10)

s.t. Eqs. (4.B.11) – (4.B.21)

[Sub2-2] if $\hat{R} = 0$:

min (4.B.10)

s.t. Eqs. (4.B.11) – (4.B.20)

The objective function (4.B.10) is to minimise the total retrieval time of the non-smart containers. Constraints (4.B.11) ensure that all the non-smart containers are stored in the non-smart area. Constraints (4.B.12) guarantee the capacity feasibility of each bay. Constraints (4.B.13) and (4.B.14) determine the height of each non-smart stack and guarantee that containers are stacked from the ground and are stacked on top of one another. Constraints (4.B.15) calculate the total relocation time for retrieving the containers in a bay. Constraints (4.B.16) and (4.B.17) define the auxiliary decision variables f_{brt}. Constraints (4.B.18)-(4.B.20) define the domains of the decision variables. Constraints (4.B.21) ensure that the stacks reserved for smart containers in the mixed bay are not occupied by non-smart containers.

B.4. An illustrative example for Subproblem 2

Fig. 4. B.1 illustrates the storage area that is temporarily pre-allocated to smart containers and non-smart containers in subproblem 2. In this example, the given stacking area consists of six consecutive bays from bay 1 to bay 6 (i.e., B_i = 6), six rows (R=6) and five tiers (T=5). The first bay is allocated to full smart piles whose heights equal the maximum stacking height of five. The middle four bays from bay 2 to bay 5 are allocated to non-smart containers, in which the rightmost two stacks in bay 5 are allocated to smart piles as bay 5 is a mixed bay. The last bay is allocated to smart piles.

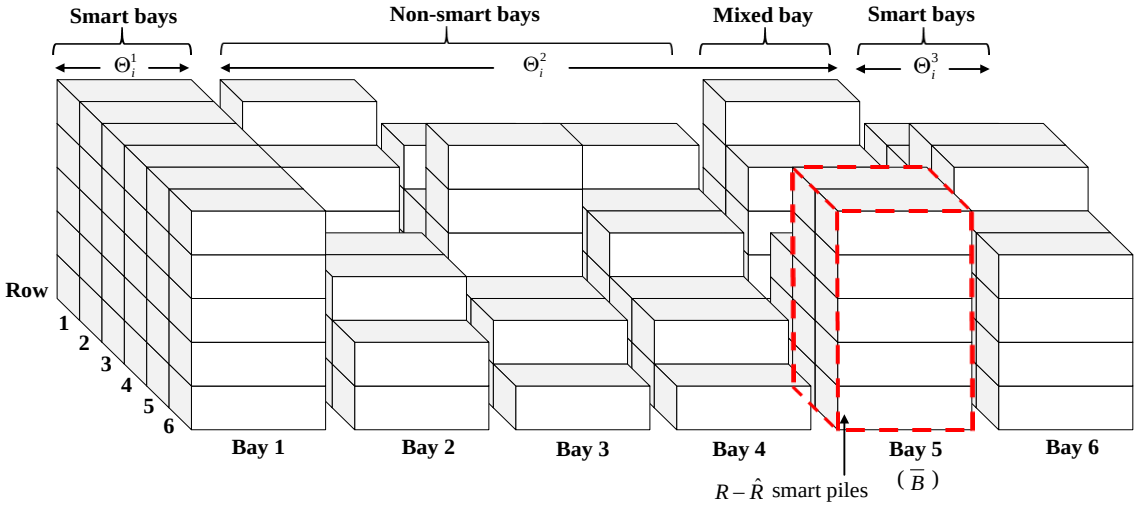


Fig. 4. B.1. Illustration of the pre-allocated area to smart and non-smart containers in subproblem 2

B.5. The integer programming model for Subproblem 3

The new parameters and decision variables for subproblem 3 are defined as follows:

Parameters:

Φ : the set of smart piles;

Ω : the set of non-smart piles;

h_k: the height of pile k, $k \in \Phi \bigcup \Omega$.

Decision variables:

u_{br}^k : equals one if smart pile *k*, $k \in \Phi$, is allocated to stack (*b*, *r*), , $b \in \Theta_i$, $r \in \{1,...,R\}$; and zero otherwise;

v_{br}^k : equals one if non-smart pile *k*, $k \in \Omega$ is allocated to stack (*b*, *r*), $b \in \Theta_i$, $r \in \{1,...,R\}$; and zero otherwise;

Then, subproblem 3 for the *i*th iteration can be formulated as follows:

[Sub3]:

$$\min \sum_{b\in\Theta_i}\sum_{r=1}^{R}\sum_{t=1}^{T} T_{brt} \cdot y_{brt} \tag{4.B.22}$$

s.t.

$$\sum_{b\in\Theta_i}\sum_{r=1}^{R} u_{br}^k = 1, \quad \forall k \in \Phi \tag{4.B.23}$$

$$\sum_{b\in\Theta_i}\sum_{r=1}^{R} v_{br}^k = 1, \quad \forall k \in \Omega \tag{4.B.24}$$

$$\sum_{k\in\Phi} u_{br}^k + \sum_{k\in\Omega} v_{br}^k \le 1, \quad \forall b \in \Theta_i, \quad \forall r \in \{1,...,R\} \tag{4.B.25}$$

$$\sum_{r=1}^{R}\left(\sum_{k\in\Phi} u_{br}^k \cdot T + \sum_{k\in\Omega} v_{br}^k \cdot h_k\right) \le RT - (T-1)\cdot z_b, \quad \forall b \in \Theta_i \tag{4.B.26}$$

$$z_b \le R - \sum_{r=1}^{R}\sum_{k\in\Phi} u_{br}^k, \quad \forall b \in \Theta_i \tag{4.B.27}$$

$$z_b \cdot R \ge R - \sum_{r=1}^{R}\sum_{k\in\Phi} u_{br}^k, \quad \forall b \in \Theta_i \tag{4.B.28}$$

$$\sum_{t=1}^{T} y_{brt} = \sum_{k\in\Phi} u_{br}^k h_k + \sum_{k\in\Omega} v_{br}^k \cdot h_k, \quad \forall b \in \Theta_i, \quad \forall r \in \{1,...,R\} \tag{4.B.29}$$

$$y_{brt} \le y_{br,t-1}, \quad \forall b \in \Theta_i, \quad \forall r \in \{1,...,R\}, \forall t \in \{2...,T\} \tag{4.B.30}$$

$$u_{br}^k \in \{0,1\}, \quad \forall b \in \Theta_i, \forall r \in \{1,...,R\}, \forall k \in \Phi \tag{4.B.31}$$

$$v_{br}^k \in \{0,1\}, \quad \forall b \in \Theta_i, \forall r \in \{1,...,R\}, \forall k \in \Omega \tag{4.B.32}$$

$$y_{brt} \in \{0,1\}, \quad \forall b \in \Theta_i, \forall r \in \{1,...,R\}, \forall t \in \{1,...,T\} \tag{4.B.33}$$

$$z_b \in \{0,1\}, \quad \forall b \in \Theta_i \tag{4.B.34}$$

The objective function (4.B.22) is to minimise the ASC travel time. Constraints (4.B.23) ensure that each smart pile is allocated to a stack in the given storage area. Constraints (4.B.24) ensure that each non-smart pile is allocated to a stack in the given storage area. Constraints (4.B.25) make sure that each stack can only store either a smart pile or a non-smart pile. Constraints (4.B.26) guarantee the capacity feasibility of each bay. Constraints (4.B.27) and (4.B.28) enforce that z_b equals zero if all the piles allocated to bay *b* are smart and equals one if there are non-smart piles allocated to bay *b*. Constraints (4.B.29)-(4.B.30) determine the height of each stack and guarantee that containers are stacked from the ground and are stacked on top of one another. Constraints (4.B.31)-(4.B.34) are the binary constraints.

B.6. The heuristic rule for solving Subproblem 3

The heuristic rule used for allocating the locations of the smart piles and non-smart piles is described below. The following notations are used for describing the heuristic rule. The first three notations are

newly defined, and the remaining ones have been defined in Section 4.5, which are recalled here.

θ_S : the sorting list of smart piles;

θ_N : the sorting list of non-smart piles;

I_b: the total number of containers in bay b, $b \in \Theta_i$;

Φ : the set of smart piles;

Ω : the set of non-smart piles;

Φ_T : the set of full smart piles with T number of containers;

B_1 : the set of bays in Θ_i located before bay $\hat{b}$ ($\hat{b}$ is the minimum bay in the block that satisfies $T_{\hat{b}r} = T_b$ for any $1 < r < R$);

Θ_i^1: the first $\left\lfloor \left|\Phi_T\right| / R \right\rfloor$ bays in Θ_i, which are allocated to full smart piles and are smart bays;

Θ_i^2: the next $\lceil S_n / R \rceil$ bays in Θ_i, which are allocated to the non-smart containers and are non-smart bays;

$\overline{B}$: the last bay in Θ_i^2 ;

$\hat{R}$: the number of non-smart piles in bay $\overline{B}$ if $\overline{B}$ is a mixed bay, $\hat{R} = S_n \% R$;

Θ_i^3: the last $\lfloor S_s / R \rfloor - \left|\Theta_i^1\right|$ bays in Θ_i, which are allocated to the smart piles that are neither in Θ_i^1 nor in $\overline{B}$;

Step 1: *Constructing temporary bays*.

Step 1.1: Sort the smart piles in Φ according to h_k in descending order and obtain the sorting list θ_S .

Step 1.2: Position the first $\left|\Theta_i^1\right| \cdot R$ smart piles in the sorting list θ_S to the bays in Θ_i^1 , starting from the leftmost row to the rightmost row in each bay.

Step 1.3: Sort the non-smart piles in Ω according to their locations (b, r) obtained in subproblem 2 in ascending order of b and ties are broken in ascending order of r, and obtain the sorting list θ_N .

Step 1.4: Position the non-smart piles in the sorting list θ_N to the bays in Θ_i^2, starting from the leftmost row to the rightmost row in each bay.

Step 1.5: If $\hat{R} = 0$, allocate the remaining smart piles in the sorting list θ_S to the bays in Θ_i^3, starting from the leftmost row to the rightmost row in each bay; otherwise, first, allocate the first (R- $\hat{R}$) in the remaining piles in θ_S to bay $\overline{B}$, starting from row $\hat{R}+1$ to row R, and then allocate all the remaining piles in θ_S to the bays in Θ_i^3, starting from the leftmost row to the rightmost row in each bay.

Step 2: *Re-allocating temporary bays*. Calculate the total number of containers I_b in each bay $b \in \Theta_i$. Sort the bays in descending order of I_b and then re-allocate the sorted bays to the locations of the bays in Θ_i from the landside to the seaside.

Step 3: *Re-allocating piles in* B_1. For each bay $b \in \mathrm{B}_1$, re-allocate its piles based on T_{br} such that a higher pile is allocated to a stack with greater T_{br}.

Acknowledgments

This study is partially supported by the China Scholarship Council, the Royal Society (Grant No. IEC\NSFC\170100), and the EU H2020 (Grant No. 777742, EC H2020-MSCA-RISE-2017).

Reference

Borgman, B., van Asperen, E., & Dekker, R. (2010). Online rules for container stacking. OR spectrum, 32(3), 687-716.

Bruns, F., Knust, S., & Shakhlevich, N. V. (2016). Complexity results for storage loading problems with stacking constraints. European Journal of Operational Research, 249(3), 1074-1081.

Bonney, J. (2015). US ports move toward truck appointment model. JOC.com. Apr 27, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-new-york-and-new-jersey/us-ports-move-toward-truck-appointment-model_20150427.html. Accessed January 22, 2021.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014a). Storage yard operations in container terminals: Literature overview, trends, and research directions. European journal of operational research, 235(2), 412-430.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014b). Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. European journal of operational research, 236(1), 1-13.

Chang, Y., & Zhu, X. (2019). A Novel Two-Stage Heuristic for Solving Storage Space Allocation Problems in Rail–Water Intermodal Container Terminals. Symmetry, 11(10), 1229.

Chen, L., & Lu, Z. (2012). The storage location assignment problem for outbound containers in a maritime terminal. International Journal of Production Economics, 135(1), 73-80.

De Castillo, B., & Daganzo, C. F. (1993). Handling strategies for import containers at marine terminals. Transportation Research Part B: Methodological, 27(2), 151-166.

de Melo da Silva, M., Erdoğan, G., Battarra, M., & Strusevich, V. (2018). The block retrieval problem. European Journal of Operational Research, 265(3), 931-950.

Dekker, R., Voogd, P., & van Asperen, E. (2006). Advanced methods for container stacking. OR spectrum, 28(4), 563-586.

Dupin, C. (2015). Improving on 'Free-Flow'. Freightwaves.com. March 6, 2015. Retrieved from https://www.freightwaves.com/news/improving-on-free-flow. Accessed January 13, 2021.

DP World London Gateway. (2018). DP World's UK logistics facilities already have the customs clearance, inspection facilities and infrastructure in place to keep trade flowing. Retrieved from https://www.londongateway.com/news-media/blogs/unpacking-brexit. Accessed January 14, 2021.

Feng, Y., Song, D. P., Li, D., & Zeng, Q. (2020). The stochastic container relocation problem with flexible service policies. Transportation Research Part B: Methodological, 141, 116-163.

Gaete, M., González-Araya, M. C., González-Ramírez, R. G., & Astudillo, C. (2017). A Novel Storage Space Allocation Policy for Import Containers. International Conference on Operations Research and Enterprise Systems, vol. 884, Springer, 2017, pp. 293–316.

Galle, V., Boroujeni, S. B., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2016). An average-case asymptotic analysis of the Container Relocation Problem. Operations Research Letters, 44(6), 723-728.

Galle, V., Manshadi, V. H., Boroujeni, S. B., Barnhart, C., & Jaillet, P. (2018a). The stochastic container relocation problem. Transportation Science, 52(5), 1035-1058.

Galle, V., Barnhart, C., & Jaillet, P. (2018b). Yard Crane Scheduling for container storage, retrieval, and relocation. European Journal of Operational Research, 271(1), 288-316.

Gharehgozli, A. H., Yu, Y., de Koster, R., & Udding, J. T. (2014). A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal. International Journal of Production Research, 52(9), 2592-2611.

Gharehgozli, A., & Zaerpour, N. (2018). Stacking outbound barge containers in an automated deep-sea terminal. European Journal of Operational Research, 267(3), 977-995.

Giuliano, G., & O'Brien, T. (2007). Reducing port-related truck emissions: The terminal gate appointment system at the Ports of Los Angeles and Long Beach. Transportation Research Part D: Transport and Environment, 12(7), 460-473.

Guldogan, E. U. (2011). Simulation-based analysis for hierarchical storage assignment policies in a container terminal. Simulation, 87(6), 523-537.

Guerra-Olivares, R., Smith, N. R., González-Ramírez, R. G., García-Mendoza, E., & Cárdenas-Barrón, L. E. (2018). A heuristic procedure for the outbound container space assignment problem for small and midsize maritime terminals. International Journal of Machine Learning and Cybernetics, 9(10), 1719-1732.

He, Y., Wang, A., & Su, H. (2019). The impact of incomplete vessel arrival information on container stacking. International Journal of Production Research, 1-15.

Iris, Ç., Christensen, J., Pacino, D., & Ropke, S. (2018). Flexible ship loading problem with transfer vehicle assignment and scheduling. Transportation Research Part B: Methodological, 111, 113-134.

Jang, D. W., Kim, S. W., & Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations. International Journal of Production Economics, 143(2), 256-262.

Jiang, X., Lee, L. H., Chew, E. P., Han, Y., & Tan, K. C. (2012). A container yard storage strategy for improving land utilization and operation efficiency in a transshipment hub port. European journal of operational research, 221(1), 64-73.

Jiang, X., Chew, E. P., Lee, L. H., & Tan, K. C. (2013). Flexible space-sharing strategy for storage yard management in a transshipment hub port. OR spectrum, 35(2), 417-439.

Jin, J. G., Lee, D. H., & Cao, J. X. (2016). Storage yard management in maritime container terminals. Transportation Science, 50(4), 1300-1313.

Kang, J., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. Journal of Intelligent Manufacturing, 17(4), 399-410.

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. Computers & Industrial Engineering, 32(4), 701-711.

Kim, K. H., & Kim, H. B. (1999). Segregating space allocation models for container inventories in port container terminals. International Journal of Production Economics, 59(1-3), 415-423.

Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 124(1), 89-101.

Kim, K. H., & Park, K. T. (2003). A note on a dynamic space-allocation method for outbound containers. European Journal of Operational Research, 148(1), 92-101.

Kizilay, D., & Eliiyi, D. T. (2020). A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals. Flexible Services and Manufacturing Journal, 1-42.

Ku, D., & Arthanari, T. S. (2016a). Container relocation problem with time windows for container departure. European Journal of Operational Research, 252(3), 1031-1039.

Ku, D., & Arthanari, T. S. (2016b). On the abstraction method for the container relocation problem.

Computers & Operations Research, 68, 110-122.
Lee, C. Y., & Song, D. P. (2017). Ocean container transport in global supply chains: Overview and research opportunities. Transportation Research Part B: Methodological, 95, 442-474.
Lee, L. H., Chew, E. P., Tan, K. C., & Han, Y. (2007). An optimization model for storage yard management in transshipment hubs. In Container Terminals and Cargo Systems (pp. 107-129). Springer, Berlin, Heidelberg.
Lee, W. S., Ottjes, J. A., Veeke, H. P., & Rijsenbrij, J. C. (2008). Using container call time information for restacking reduction. In Proc. 6th International Industrial Simulation Conference, pp. 293- 298.
Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. European Journal of Operational Research, 239(2), 297-312.
Li, Y., Zhu, X., Wang, L., & Chen, X. (2017). Stowage plan based slot optimal allocation in rail-water container terminal. Journal of Control Science and Engineering, 2017.
Li, Z., Zhu, S., & Zhou, M. (2009). A divide-and-conquer strategy to deadlock prevention in flexible manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 39(2), 156-169.
Lin, D. Y., & Chiang, C. W. (2017). The storage space allocation problem at a container terminal. Maritime Policy & Management, 44(6), 685-704.
Maldonado, S., González-Ramírez, R. G., Quijada, F., & Ramírez-Nafarrate, A. (2019). Analytics meets port logistics: A decision support system for container stacking operations. Decision Support Systems, 121, 84-93.
Monaco, M. F., Sammarra, M., & Sorrentino, G. (2014). The terminal-oriented ship stowage planning problem. European Journal of Operational Research, 239(1), 256-265.
Mongelluzzo, B., 2015a. Free-flow program finds success in Los Angeles port. JOC.com. May 22, 2015. Retrieved from https://www.joc.com/port-news/us-ports/port-los-angeles/free-flow-program-finds-success-los-angeles-port_20150522.html. Accessed January 13, 2021.
Mongelluzzo, B., 2015b. Free-flow container operation launched in Port of Los Angeles. JOC.com. Feb 27, 2015. https://www.joc.com/port-news/us-ports/port-los-angeles/initiative-aimed-reducing-los-angeles-port-congestion-launched_20150227.html. Accessed January 13, 2021.
Park, T., Choe, R., Kim, Y. H., & Ryu, K. R. (2011). Dynamic adjustment of container stacking policy in an automated container terminal. International Journal of Production Economics, 133(1), 385-392.
Petering, M. E., Wu, Y., Li, W., Goh, M., de Souza, R., & Murty, K. G. (2017). Real-time container storage location assignment at a seaport container transshipment terminal: dispersion levels, yard templates, and sensitivity analyzes. Flexible Services and Manufacturing Journal, 29(3-4), 369-402.
Preston, P., & Kozan, E. (2001). An approach to determine storage locations of containers at seaport terminals. Computers & Operations Research, 28(10), 983-995.
Parreño-Torres, C., Alvarez-Valdes, R., & Ruiz, R. (2019). Integer programming models for the pre-marshalling problem. European Journal of Operational Research, 274(1), 142-154.
Parker, B. (2015). Reduce Port Congestion By Expanding Container 'Free-Flow' Systems. SupplyChainBrain.com, December 14, 2015. Retrieved from https://www.supplychainbrain.com/articles/22934-reduce-port-congestion-by-expanding-container-free-flow-systems. Accessed January 13, 2021.
Phan, M. H., & Kim, K. H. (2016). Collaborative truck scheduling and appointments for trucking

companies and container terminals. Transportation Research Part B: Methodological, 86, 37-50.

Razouk, C., Benadada, Y., & Boukachour, J. (2016). New approaches for solving the container stacking problem. In 2016 3rd International Conference on Logistics Operations Management (GOL) (pp. 1-7). IEEE.

Rasmussen, M. S., & Larsen, J. (2011). Optimisation-based solution methods for set partitioning models (Doctoral dissertation, Ph. D. thesis, Technical University of Denmark).

Rekik, I., & Elkosantini, S. (2019). A multi agent system for the online container stacking in seaport terminals. Journal of Computational Science, 35, 12-24.

Rekik, I., Elkosantini, S., & Chabchoub, H. (2018). A case based heuristic for container stacking in seaport terminals. Advanced Engineering Informatics, 38, 658-669.

Reimann, M., Doerner, K., & Hartl, R. F. (2004). D-ants: Savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research, 31(4), 563-591.

Sauri, S., & Martin, E. (2011). Space allocating strategies for improving import yard performance at marine terminals. Transportation Research Part E: Logistics and Transportation Review, 47(6), 1038-1057.

Sculli, D., & Hui, C. F. (1988). Three dimensional stacking of containers. Omega, 16(6), 585-594.

Smith, D. R. (1985). The design of divide and conquer algorithms. Science of Computer Programming, 5, 37-58.

Tang, L., Jiang, W., Liu, J., & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions, 47(7), 751-766.

Tanaka, S., Tierney, K., Parreño-Torres, C., Alvarez-Valdes, R., & Ruiz, R. (2019). A branch and bound approach for large pre-marshalling problems. European Journal of Operational Research, 278(1), 211-225.

The Port of Long Beach and Port of Los Angeles, 2017. San Pedro Bay Ports Clean Air Action Plan 2017 Draft Final. https://kentico.portoflosangeles.org/getmedia/9d371f7b-9812-4c75-bcfd-23e83a191435/CAAP_2017_Draft_Document-Final. Accessed January 22, 2021.

UNCTAD (2019). Review of Maritime Transport 2019. Technical report, United Nations Conference on Trade and Development, New York and Geneva.

UNCTAD (2020). Review of Maritime Transport 2020. Technical report, United Nations Conference on Trade and Development, New York and Geneva.

Wang, N., Chang, D., Shi, X., Yuan, J., & Gao, Y. (2019). Analysis and design of typical automated container terminals layout considering carbon emissions. Sustainability, 11(10), 2957.

Wang, L., Zhu, X., & Xie, Z. (2020). Efficient container stacking approach to improve handling: efficiency in Chinese rail–truck transshipment terminals. Simulation, 96(1), 3-15.

Wei, C., Gao, W. W., Hu, Z. H., Yin, Y. Q., & Pan, S. D. (2019). Assigning customer-dependent travel time limits to routes in a cold-chain inventory routing problem. Computers & Industrial Engineering, 133, 275-291.

Xia, M., Zhao, N., & Mi, W. (2016). Storage allocation in automated container terminals: the upper level. Polish Maritime Research, 23(s1), 160-174.

Yang, J. H., & Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems. Journal of Intelligent Manufacturing, 17(4), 453-463.

Yu, M., & Qi, X. (2013). Storage space allocation models for inbound containers in an automatic container terminal. European Journal of Operational Research, 226(1), 32-45.

Zaerpour, N., Yu, Y., & de Koster, R. B. (2015). Storing Fresh Produce for Fast Retrieval in an Automated

Compact Cross‐Dock System. Production and Operations Management, 24(8), 1266-1284.
Zeng, Q., Feng, Y., & Yang, Z. (2019). Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering, 127, 366-382.
Zehendner, E., Caserta, M., Feillet, D., Schwarze, S., & Voß, S. (2015). An improved mathematical formulation for the blocks relocation problem. European Journal of Operational Research, 245(2), 415-422.
Zhao, W., & Goodchild, A. V. (2010). Impact of truck arrival information on system efficiency at container terminals. Transportation Research Record, 2162(1), 17-24.
Zhao, N., Xia, M., Mi, C., Bian, Z., & Jin, J. (2015). Simulation-based optimization for storage allocation problem of outbound containers in automated container terminals. Mathematical Problems in Engineering, 2015.
Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. Transportation Research Part B: Methodological, 37(10), 883-903.
Zhang, C., Chen, W., Shi, L., & Zheng, L. (2010). A note on deriving decision rules to locate export containers in container yards. European Journal of Operational Research, 205(2), 483-485.
Zhang, C., Wu, T., Kim, K. H., & Miao, L. (2014). Conservative allocation models for outbound containers in container terminals. European Journal of Operational Research, 238(1), 155-165.
Zhen, L., Jiang, X., Lee, L. H., & Chew, E. P. (2013). A review on yard management in container terminals. Industrial Engineering and Management Systems, 12(4), 289-304.
Zhen, L. (2016). Modeling of yard congestion and optimization of yard template in container ports. Transportation Research Part B: Methodological, 90, 83-104.
Zhou, C., Chew, E. P., & Lee, L. H. (2018). Information-based allocation strategy for GRID-based transshipment automated container terminal. Transportation Science, 52(3), 707-721.
Zhou, C., Wang, W., & Li, H. (2020). Container reshuffling considered space allocation problem in container terminals. Transportation Research Part E: Logistics and Transportation Review, 136, 101869.
Zhu, H., Ji, M., & Guo, W. (2020). Two-stage search algorithm for the inbound container unloading and stacking problem. Applied Mathematical Modelling, 77, 1000-1024.
Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. IEEE Transactions on Automation Science and Engineering, 9(4), 710-722.

Chapter 5

Conclusion

In this chapter, we summarise the executed work and research findings, outline possibilities to extend this work and discuss some open challenges for future research.

5.1 Summary and discussions

Container terminal operations encompass three main logistic sub-systems where the storage yard is the central intersection of seaside and landside operations within the terminal. The management of yard operations is of paramount importance in determining the efficiency and competitiveness of a terminal. As container relocation is the major source of inefficiency in yard operations, the optimisation of container handling in yard blocks to reduce relocations has been an active research stream in Operations Research in the past two decades. Five problem types have been defined and addressed in the literature, which involves the process of container storage, container marshalling and container retrieval. The optimisation of these problems is highly dependent on the retrieval times of containers. Although such data is often uncertain, the majority of research assumes deterministic settings, which could jeopardise the practicality of the solutions. A recent research trend has seen an increasing interest in taking into account uncertainties but it remains less studied compared to the dominant deterministic research stream. This thesis contributes to the literature on the optimisation of container handling operations in yard blocks under uncertainties. As there is a strong drive for addressing issues such as over-utilisation of yard storage space and port congestion, besides pursuing optimisation methods, there is a need for seeking innovative container handling strategies and utilising potential information. This thesis proposes new strategies for container retrieval and stacking and address the corresponding optimisation problems to improve the import container retrieval performance. The research work is presented in Chapters 2 - 4 in the format of three papers (published and unpublished). Chapter 2 and Chapter 3 addresses the Container Relocation Problem (CRP), and Chapter 4 addresses the Storage Slot Assignment Problem (SLAP).

In the following, Section 5.1.1 summarises the work and main findings of each paper, followed by the discussions of the findings in the three papers in relation to each other and in relation to wider literature in Section 5.1.2.

5.1.1 Summary of each paper

In Chapter 2 and Chapter 3, we extend an uncertain version of the CRP – the Stochastic Container Relocation Problem (SCRP), where the randomness of truck arrival within a time window is modelled by

scenarios with given probabilities and the goal is to find a globally optimal container relocation sequence that minimises the expected total number of relocations, which is originally introduced in Ku and Arthanari (2016) and extended by Galle et al. (2018). Our novelty is the introduction of flexible service policies that allow the terminal operator to determine the service sequence of the external trucks to some extent during the container retrieval process, as opposed to the traditional first-come-first-serve policy. In addition, we relax the assumption of uniformly distributed truck arrivals within a time window by a more general probabilistic model that considers the customers (trucks)' arrival preference over a time window.

In Chapter 2 (**Paper 1**), the flexible service policy is applied in a two sub-time windows-based situation where each appointment time window is divided into two sub-time windows with equal length, and the trucks that have arrived at the same sub-time window are allowed to be served out-of-order. The arising problem is termed as the SCRP with Flexible Service policies (SCRP-FS). We present a stochastic dynamic programming model with two lexicographically ordered objectives, where the primary objective is to minimise the expected total number of relocations and the secondary objective is to minimise the total truck waiting times of each batch. Correspondingly, tree search-based exact algorithms and heuristic algorithms are developed to obtain optimal solutions and near-optimal solutions respectively. Efficiencies of all algorithms are evaluated through computational experiments by a discrete event-driven simulation model. Computational results show that the computation times of the exact algorithm deviate greatly for different instances even their problem classes are the same. For some instances, an optimal solution can be obtained in less than one second, while for others, optimal solutions are not able to be found within an hour time limit. The look-ahead heuristic algorithm can solve all problems in a matter of milliseconds. For the problems that we have access to optimal solutions, the accuracy of the heuristic algorithm is within 2% gaps in most cases. In addition, both the number of relocations and the average truck waiting time can be significantly reduced by applying the flexible service policy. The main contribution of this work is the novelty and effectiveness of the flexible service policy and the adaptability of the methodological framework. The proposed methodology is applicable to solve multi-stage stochastic optimisation problems in which decisions are made sequentially and multiple objectives are lexicographically ordered.

Flexible service policies provide more opportunities for reducing the number of relocations but may raise the concern of service fairness because some earlier arriving trucks can be served later than some later arriving trucks. In Chapter 3 (**Paper 2**), we extend the SCRP-FS to the case with multiple sub-time windows, that is, the SCRP with Multiple sub-time windows-based Flexible Service policy (SCRP-MFS). In the SCRP-MFS, an appointment time window is divided into multiple sub-windows such that the level of service flexibility can be controlled more accurately to mitigate the issue of service fairness. The problem is formulated into a stochastic dynamic programming model with two lexicographically ordered objectives minimising the expected total number of relocations (primary) and the maximum truck turnaround time (secondary), which is solved via a hierarchical iterative approach. The computational results show interesting trade-offs between relocation efficiency and service fairness. A counter-intuitive finding is that the considered service fairness does not necessarily improve as the number of sub-time windows increases. Another important finding is that the information on the customer arrival preference over a time window can be valuable in reducing the number of relocations, especially when each truck indicates a certain arrival sub-time window. These findings can help the terminal operator to better determine the number of sub-time windows when applying the flexible service to achieve a balance between relocation efficiency and service fairness, and decide whether it is worth committing effort to capture the customer preference information.

In Chapter 4 (**Paper 3**), we deal with the determination of the initial storage locations of import

containers into a yard block by addressing the Storage Slot Assignment Problem (SLAP). Our novelty is the introduction of the Smart Stacking (SS) strategy that intelligently categorises containers into smart containers and non-smart containers by utilising customer identity information. Under the SS strategy, relocations can be avoided for the smart stacks that store containers only from the same customer (smart containers); on the other hand, relocations are required for the non-smart stacks that store containers from multiple customers (non-smart stacks). Depending on whether the containers from the same customer are allowed to be split between smart stacks and non-smart stacks, two SS policies arise, that is, the non-split policy and the split policy. Under each policy, the problem is to determine the smart customers or containers, the number and locations of smart stacks and non-smart stacks, and assign a batch of import containers to exact stacking positions in a yard block. The objective is to minimise the total retrieval time that is the sum of the relocation time and the yard crane travel time. Both the non-split variant and the split variant are formulated by a mixed-integer programming model. For the non-split variant, we leverage the structural properties of the optimal solution to develop an improved formulation with enhanced computational performance. By analysing the structure of the model, we then develop a divide-and-conquer heuristic algorithm to solve the non-split variant. The computational results show that the heuristic algorithm is efficient in that it can obtain near-optimal solutions within 0.6% gaps from the lower bound for sufficiently large problems (with more than one thousand containers), with the runtimes ranging from only less than one second to about six seconds. In addition, the SS strategy can significantly reduce the total retrieval time compared to current practice, and the effectiveness is more significant in situations with a larger amount of customer information and a wider range of group sizes. Moreover, it is proved that the split policy yields better benefits than the non-split policy both theoretically and computationally. These findings demonstrate the value of customer information to container stacking and highlight the importance of the information quality, which can motivate the collaboration between terminal operations and their customers especially those with high-volume containers. The superiority of the split policy can provide an insight into whether or not to allow splitting when applying the SS strategy for terminal operators.

5.1.2 Discussions

In the following, we first discuss the findings in the three papers in relation to each other, and then discuss the findings in relation to wider literature.

5.1.2.1 Findings in relation to each other

Firstly, the findings in all three papers are related to the import container retrieval performance.

- Papers 1 and 2 focus on the performance of the number of relocations and the waiting time of external trucks. Paper 1 finds that both the number of relocations and the average truck waiting time can be significantly reduced by applying the flexible service policy.
- Based on Paper 1, Paper 2 further shows that there is a trade-off between relocation efficiency (the number of relocations) and service fairness (the maximum truck turnaround time) when applying the flexible service policy.
- Paper 3 focuses on the total time needed for retrieving containers. Besides the time spent on relocation activities, the yard crane travel time for moving containers from the storage slots to the transfer points is also incorporated into the objective function. It finds that the total retrieval time, which is the sum of the relocation time and yard crane travel time, can be significantly reduced by applying the smart stacking strategy compared to the current practice.

Secondly, all three papers prove that customer information is valuable to the terminal operators.

- Papers 1 and 2 focuses on the customer arrival preference information and utilises it in the container retrieval and relocation process. Paper 1 finds that if all customers prefer the same sub-time window, the number of relocations will be significantly different compared to the case of equal preference.
- Paper 2 further demonstrates that utilising the customer arrival preference information can lead to a statistically significant reduction in the number of relocations in comparison with not utilising such information, especially when each truck indicates a certain arrival sub-time window.
- Paper 3 focuses on the customer identity information and utilises it in the container stacking process. It finds that the customer identity information is useful for determining the container initial storage positions, which can reduce the future retrieval time significantly.

5.1.2.2 Findings in relation to wider literature

Overall, the findings in the thesis (i) complement and advance the knowledge of the effects of flexible service policies on import container retrieval performance, and (ii) advance the knowledge of the value of customer information to import container retrieval performance. Specifically, our findings complement and advance the knowledge in the literature in the following two aspects:

(i) The effects of flexible service policies.

Only a few studies have applied the concept of flexible service or out-of-order service to the CRP for import containers and have discussed the effects (Zhao and Goodchild, 2010; Borjian et al., 2013; Borjian et al., 2015; Zeng et al., 2019). Table 5.1 presents these studies from three key aspects: the flexibility in service sequence, the characteristics of the truck arrival information, and the main findings regarding the effects of flexible service.

Table 5.1 Closely relevant literature on the CRP for import containers with flexible retrieval.

Literature	Flexibility in service sequence	Truck arrival information	Effects of flexible service
Zhao and Goodchild (2010)	Out-of-order retrieval within the first group	Arrival group and arrival order within the first group is known; exact arrival order is updated on a truck unit basis	Fewer relocations
Borjian et al. (2013)	Out-of-order retrieval s.t. a maximum delay	Retrieval time window for each container is known	Fewer relocations and less delay
Borjian et al. (2015)	A limited number of out-of-order retrievals before each truck	Retrieval time window for each container is known	Fewer relocations, less delay, and unaffected service equity
Zeng et al. (2019)	Out-of-order retrieval within each group	Arrival group is known	Fewer relocations and longer average truck waiting times
Paper 1 in the thesis	Out-of-order retrieval within each sub-group (two sub-time windows-based)	Arrival group is known; exact arrival order is updated on a group size basis	Fewer relocations and shorter truck waiting times
Paper 2 in the thesis	Out-of-order retrieval within each sub-group (multi-sub-time windows-based)	Arrival group is known; exact arrival order is updated on a group size basis	*Increased relocations, and reduced maximum truck turnaround time and coefficient of variation of truck turnaround time (but not guaranteed)

Note: "*" represents in comparison with the results of Paper 1.

As shown in Table 5.1, in previous studies, the effects of flexible service on the CRP have been discussed regarding the performances of the number of relocations, retrieval delays, truck waiting times, and service equity. Our findings in Paper 1 and Paper 2 complement the existing findings in the following ways, and they together advance the knowledge of the effects of flexible service policies on import container retrieval performance.

- Paper 1 and Paper 2 complement Zhao and Goodchild (2010) by adding the effects on truck waiting times and service fairness.
- Paper 1 and Paper 2 complement Borjian et al. (2013), Borjian et al. (2015) and Zeng et al. (2019) by generating findings in an uncertain decision-making context.
- Paper 1 complements Zeng et al. (2019) by concluding a reduced average truck waiting times.
- Paper 2 complements Borjian et al. (2015) by demonstrating the trade-off effects between relocation efficiency and service fairness.

(ii) The value of customer information.

Customer information has been increasingly utilised to seek more efficient yard operations, which span over container stacking, marshalling, retrieving and relocating. Table 5.2 presents the relevant studies from three key aspects: the type of operations, the information utilised, and the main findings regarding the value of information.

Table 5.2 Closely relevant literature on the value of customer information to container handling at yards.

Literature	Operation type	Utilised information	Value of information
Zhao and Goodchild (2010)	Retrieving and relocating	Truck arrival group; updated exact truck arrival order on a truck unit basis	Fewer relocations by utilising the arrival group information; Further fewer relocations when updating the arrival order
Borjian et al. (2013)	Retrieving and relocating	Probabilities of the container departure time scenarios	Fewer relocations by having more accurate information and knowing the information earlier
Galle et al. (2018)	Retrieving and relocating	Updated exact truck arrival order on a group basis	Fewer relocations by utilising the within group information
van Asperen et al. (2013)	Stacking and re-marshalling	Container departure time	The earlier the announcement of the information, the better the stacking efficiency
Gharehgozli and Zaerpour (2018)	Stacking	Barges arrival time windows	Reduced total retrieval time
Covic (2017)	Re-marshalling	Truck appointment information	Reduced truck waiting time by utilising the imprecise truck arrival information that does not deviate above a certain threshold
Kim and Yi (2021)	Stacking and pre-marshalling	Truck arrivals (dwell time distribution of containers, truck dispatching notice, truck appointment, and real-time position of trucks).	Fewer relocations and reduced truck system time by utilising all these sources of information
Paper 1 in the thesis	Retrieving and relocating	Customer arrival preference (i.e., truck arrival probabilities over a time window)	Fewer relocations if all customers prefer the same sub-time window
Paper 2 in the thesis	Retrieving and relocating	Customer arrival preference (i.e., truck arrival probabilities over a time window)	Fewer relocations
Paper 3 in the thesis	Stacking	Customer identity information	Reduced total retrieval time

The findings in our three papers together advance the knowledge of the value of customer information to import container retrieval performance. The ways how they advance are specified below.

As shown in Table 5.2, three studies have proved the value of the information about the *container departure times* to the import container retrieval performance by utilising the information in the container

retrieving and relocating process (Zhao and Goodchild, 2010; Borjian et al., 2013; Galle et al., 2018). Different from them, our work in Paper 1 and Paper 2 develops new knowledge on the value of *customer arrival preference information* to the import container retrieval performance when the information is applied to the retrieving and relocating process.

In addition, some studies have proved the value of the *arrival information of carriers* (trucks and barges) to container stacking or/and retrieval efficiencies by utilising the information in the container *stacking* or/and marshalling processes (van Asperen et al., 2013; Gharehgozli and Zaerpour, 2018; Covic, 2017; Kim and Yi, 2021). Different from these studies, our work in Paper 3 develop new knowledge on the value of *customer identity information* to the import container retrieval efficiency when the information is applied to the stacking process.

5.2 Overall contribution

The thesis makes theoretical, methodological and practical contributions.

Theoretical contributions

(1) The thesis is theoretically relevant to academics by adding a series of new variants of problems to the literature on container handling problems in yard blocks, which helps to obtain efficiency improvement compared to the traditional variants and practices and brings many new research directions.

(2) The thesis makes an incremental theoretical contribution by: (i) generating some knowledge (lower bound, properties, NP-hardness, etc) on the new variants of problems, (ii) demonstrating the value of customer information (customer arrival preference information and customer identification information), and (iii) demonstrating the benefits and pitfalls of flexible service policies.

Methodological contributions

(1) The proposed mathematical formulations and solution methods generate methodology rigour and scientific usefulness to the optimisation of container terminal operations (under uncertainties). The developed exact algorithm enriches the application of tree search-based algorithms to solving stochastic dynamic programming. The developed heuristic algorithms enrich the optimisation methods for solving terminal operation problems.

(2) The proposed methods can be applied to other problems with similar structures and goals by appropriate modifications.

Practical contributions

(1) The new variants of problems consider the uncertainties in the real context, which increases the practicability and significance of optimisation.

(2) The alternative service policy for container retrieval and the novel strategy for container stacking help to improve both the terminal operation efficiency and the customer service level.

(3) The proven value of customer information provides practical implications to terminal operators and their customers, which could promote industry-wide collaboration between terminal operators, trucking companies and cargo owners to improve the import container retrieval performance.

(4) The developed heuristic algorithms are efficient that can generate effective solutions in a matter of milliseconds for the container retrieval problem and within a few seconds for the container stacking problem, which can be applied to facilitate the decision making for terminal operators even for real-time decisions.

5.3 Future research directions

We now present some possible extensions from the thesis and then envision some future research directions for addressing open challenges.

5.3.1 Extensions from the thesis

5.3.1.1 Chapter 2

There are many ways to extend the work in Chapter 2.

(i) From the perspective of theoretical and methodological contribution:

- The lower bound of the expected number of relocations could be tightened to be used as a more efficient node pruner in the tree search-based algorithm. This can refer to the progress in the state-of-the-art lower bounds for the deterministic CRPs (e.g., Tanaka et al., 2016; Tanaka and Mizuno, 2018; Tricoire et al., 2018).
- Another approach could be using machine learning-driven algorithms, such as deriving pruners from machine learning techniques (Zhang et al., 2020), for solving the SCRP.
- With an orientation to heuristic solution methods, heuristic tree search procedures that explore only a subset of the search tree (e.g., Ting and Wu, 2017) and metaheuristics (e.g., Jovanovic et al., 2019) could be devised to balance the computational time and the solution quality.

(ii) From the perspective of problem definition, more realistic situations regarding the truck arrival uncertainties could be considered. In the real world, it cannot be guaranteed that a truck will arrive within its appointed time window for sure, due to the stochastic nature of the road travel time and the planning changing within the truck companies, etc. The arrival of a truck may be prior to or later than the appointed time window or even not arriving (no-show). This will lead to different problem settings and correspondingly, may require different modelling techniques and resulting solution approaches.

- The model in this work may be extended to account for such uncertainties to approach the new problem by taking the stochastic optimisation paradigm.
- Alternatively, a robust optimisation approach may be devised to hedge against the worst-case scenario. The application of robust optimisation has been studied in the closely related pre-marshalling problem in Rendl and Prandtstetter (2013), Tierney and Voß (2016), and Boge et al. (2020) but has not appeared in the CRP.
- It would also be interesting to target the problem from the recoverable robustness viewpoint with a focus on the recovery cost by applying the recoverable robust optimisation approach (e.g., Iris and Lam, 2019a).
- However, with many alternative approaches available, it is needed to critique the merits and defects of different approaches. With respect to this, a further question should be developing an evaluation framework in order to allow a fair comparison of the performance of different approaches in addressing the same problem setting.

(iii) Finally, it would be possible to investigate the application of the ideas and methodology provided in this work to other variants of the CRP, such as the unrestricted CRP under uncertainties.

5.3.1.2 Chapter 3

The work in Chapter 3 can be extended in several directions.

(i) From the perspective of methodological contribution, exact solution methods need to be developed in order to allow a benchmarking for evaluating the performance of the heuristic solution method. This is

expected to be computationally challenging because the size of the search tree would be very large when multiple sub-time windows are considered due to more possibilities of scenarios. Besides the best-first-search strategy used in Chapter 2, alternative branching strategies and pruning strategies could be tested to speed up the search procedure. The computational time might also be improved if more efficient abstract techniques are used for checking repeated nodes, such as using a hash table that compares only the hash value of two nodes instead of their exact configurations (Russell and Norvig, 2016).

(ii) From the perspective of theoretical contribution, it would be interesting to derive the theoretical properties of optimal solutions by analysing the structure of the problem, as this work obtains some computational results that are not straightforward (e.g., the non-monotonicity of maximum turnaround time with the number of sub-windows).

(iii) From the perspective of problem definition, it would be interesting to introduce alternative measures, such as sequence fairness (Borjian et al., 2015) and delay fairness (Wang et al., 2017), for characterising the service fairness in the container retrieval process, and study their impacts on the multi-criteria decision-making of the SCRP.

(iv) Lastly, it would be worthwhile to investigate other operational strategies to improve the service efficiency and quality of import container retrieval and mitigate the service unfairness among trucks. Recently, Azab and Morita (2021) incorporate truck appointment scheduling into the optimisation of the deterministic CRP to reduce the number of relocations by rescheduling the container retrieval times requested in the truck appointment system. Future research could integrate this idea with the concept of flexible service and apply it in the SCRP. In this context, it would be possible to develop a dual-channel communication mechanism where the information on the trucks' preferred or estimated arrival time is updated to the terminal system and the scheduled arrival time by the terminal is feedbacked to the truck, both dynamically.

5.3.1.3 Chapter 4

The work in Chapter 4 is the first in the literature that addresses the initial storage for import containers under the smart stacking strategy. Many directions are worthwhile studying in the future.

(i) To continue this work, the split variant may be further investigated. The current split model provides an upper bound for the split policy by assuming optimal partition. A future research direction is investigating the optimality of the split variant. This would provide an insight into the quality of the upper bound at hand.

(ii) The problem studied in the current work is oriented toward a short-term operational problem in order to derive insights into the structural properties of the problem. A future research direction could focus on the long-term storage space allocation problem under the smart stacking strategy, which aims to allocate the import containers from multiple vessels to the storage slots in different yard blocks over a longer planning horizon. This problem involves two-level decisions for each container, that is, block allocation and slot assignment, which is usually decomposed into two stages in the literature (e.g., Chen and Lu, 2012; Chang and Zhu, 2019). In this respect, the modelling and solution methods developed in this chapter can function as a building block for this long-term planning.

(iii) In order to efficiently utilise the yard storage space, it may require to incorporate the temporal information (e.g., retrieval time) of containers into the optimisation such that containers of different vessels can share stacks without causing mis-overlays. Because such information is unavailable when containers are to be stacked into the yard, a data-driven framework can be established. In the framework, data mining techniques can be used to predict the dwell time of containers (e.g., Moini et al., 2012; Kourounioti et al., 2016), so that it is possible to differentiate the retrieval priority of containers, which is then input to the

optimisation model. This would contribute to the application of data-driven approaches in the context of terminal planning and management, which is still in its infancy but is expected to have more promising research as identified by Heilig et al. (2020).

(iv) Lastly, another meaningful research direction is applying the methodology developed in this work to sea-rail intermodal container terminals. This would increase the potential of smart stacking for practical application because a single rail carrier could have hundreds of containers from a vessel and these containers may go for the same rail head.

5.3.2 Open challenges

There are some open questions as identified in several review papers on yard operations by Caserta et al. (2011, 2020), Luo et al. (2011), Carlo et al. (2014) and Covic (2018), on energy management and operations planning in seaport by Iris and Jam (2019b), and on planning problems in the container shipping supply chain by Song (2021). Based on some of those open questions, we discuss future research directions. In the end, the challenges under the Covid-19 pandemic in the research topic and potential solutions are also discussed.

5.3.2.1 Integrated planning

The first challenge is how to make integrated planning of the container handling operations in order to move towards a global optimum for yard productivity. Some efforts in this respect can be observed where two types of container handling are jointly optimised, such as dynamic CRP that consider both container storage and retrieval (e.g., Hakan Akyüz and Lee, 2014; Tang et al., 2015), integrated optimisation of container storage and marshalling by Choi and Kim (2016), and integrated optimisation of container premarshalling and relocation by Zweers et al (2020). However, because there are interaction effects between different problems and the operating environment is highly uncertain, approaching a holistic optimisation is hardly intractable from a computational perspective. In this regard, one can turn to online optimisation. An integrated simulation-optimisation framework could be established, where optimisation tools are called to solve a particular or several consecutive container handling problems while simulation tools are used to capture the real-time data outside the system and model the interactions between different container handling processes.

5.3.2.2 Information availability and reliability

Another challenge is the availability and reliability of information. Efficient cargo flows rely on efficient information flows along the logistics chains. The maritime industry has increasingly recognised the value of information and sought decision support tools to gain competitive advantages (Heilig et al., 2020). However, terminal operators usually only possess estimated information or do not have the information at the required moment.

One of the crucial aspects for making better decisions on container stacking is the availability of reliable information on containers (container arrival and departure times, modes of onwards transport, etc). The existing studies that take the uncertainty of temporal information into account rely extensively on the knowledge of the container departure probabilities or truck arrival probabilities (e.g., Ku and Arthanari, 2016; Galle et al., 2018; Chapters 2 and 3 in the thesis), which would be hard to obtain in practice. Some studies analyse the effects of different levels of information availability and reliability on the results (e.g., Gharehgozli and Zaerpour, 2018; Covic, 2019; Chapter 4 in this thesis). Besides, some work focuses on demonstrating the value of information by computational results, which can help to motivate information collaboration between terminals and customers (e.g., Kim and Yi, 2021; Chapter 4 in the thesis).

Apart from these previous attempts, a further avenue is to revisit the issue by taking a data-driven optimisation perspective. On one hand, a possible approach is applying a two-stage framework that combines data mining and optimisation method (e.g., van Nguyen, 2020). In the first stage, by using data mining techniques, one could derive useful knowledge out of the massive flow of accessible data in terminal systems and as a result, lead to problem settings with more realistic data. In the second stage, the derived information is integrated into the optimisation model as input data. At this stage, the available methods and knowledge from the existing relevant container handling problems can be transferred or adapted to address the resulting optimisation problems. On another hand, Distributionally Robust Optimisation (DRO) is an emerging optimisation approach that could effectively leverage partial data information (such as support set and moment statistics) without any assumption about the probability distribution of uncertainties, which has attracted immense research attentions in the operations research community (Delage and Ye, 2010; Shang and You, 2018; Esfahani and Kuhn, 2018). The DRO approach has the advantage of avoiding over-conservative solutions by incorporating partial stochastic information. It has been used to address the uncertainty in the container transportation system in a few papers (e.g., Dai and Yang, 2020; Liu et al., 2021), while its application to terminal operations problems seems missing. The investigation of how DRO could add value to the optimisation of terminal operations provides broad opportunities for future research. Finally, as pointed out by Song (2021), information sharing does require trust, commitment and collaboration among participated stakeholders.

5.3.2.3 Port decarbonisation and energy efficiency

The need for port decarbonisation provides fruitful research directions. GHG (greenhouse gas) emissions from international shipping has attracted IMO's attention for many years. In 2018, IMO published an initial GHG strategy to decarbonise shipping, where the long-term goal is to reduce carbon intensity by 70% in 2050 compared to 2008 levels (Rutherford and Comer, 2018). Although the IMO strategy did not explicitly include the emissions from port activities, there is a need for more regulatory and operational effort to port decarbonisation given that ports are pollution and emission concentration areas and many ports are located close to cities (Song, 2021). Meanwhile, to mitigate the negative environmental impacts in ports, stricter regulations have been adopted by authorities to minimise port pollution and target sustainable operations over the long term (Woo et al., 2018). However, in a comprehensive literature review on green ports and maritime logistics conducted by Davarzani et al. (2016), it is found that this research field is still in its early growth and expansion period.

Port emissions reduction is closely related to energy efficiency in ports. In a systematic literature review conducted by Iris and Jam (2019b), the energy efficiency measures in ports include three categories: operational strategies (e.g., operations optimisation, peak shaving,), technologies (e.g. cold-ironing, electrification of equipment, energy storage systems), and energy management (e.g., renewable energy, alternative fuels, smart energy management systems). In order to achieve energy efficiency and emission reductions without capital investment, many ports focus on operational optimisation, for example, through energy-aware planning of operations and reducing the idle in operations, etc. However, as pointed out in Iris and Lam (2019b), currently, the number of papers on energy-aware planning is limited, and there is a need to improve the integration of energy management and operational planning.

From the perspective of energy-aware operations optimisation, container handling in yards can be better managed by factoring energy consumption and emission-related objectives into the decision-making, as did e.g. in Sha et al. (2017), Iris and Lam (2021), and Karakas et al. (2021). For this purpose, new aspects need to be taken into consideration, including the container weight in fuel consumption (see Hussein and Petering, 2012), explicit tracking of the yard crane moving activities (trolley moving, gantry moving,

hoisting and lowering), crane idle time, etc. Moreover, from the perspective of peak shaving methods, marshalling operations in the yard can be better planned considering the balance of resources usage, as advised in Iris and Lam (2019b). Given that such operations are often preparatory work conducted before vessels or trucks arrival, which is usually not urgent, it can be beneficial to conduct such operations in non-peak periods in order to shift the peak workload (noting that the electricity price depends on the consumption level). It is noted that an attempt to determine the right time to remarshall outbound containers has received attention recently (see Kim et al., 2021), but they focus on the efficiency of the vessel loading operation rather than the energy efficiency.

5.3.2.4 Challenges under COVID-19 pandemic

The COVID-19 outbreak has seriously disrupted the container shipping supply chain, including the port and terminal logistics. As the world economy reopens following the lifting of lockdown at different countries, many ports have experienced a spike in container volumes as importers seek to replenish stock levels that dropped during the lockdown phase. The surging container volumes put pressure not only on a terminal's seaside operations (ships at anchor awaiting berths) but also on its yardside operations (yard congestion) and hinterland transport network (slow-moving of containers between the ports and the inland), leading to severe congestion to many ports. The situation of port congestion is challenged by a number of factors including shortages of manpower (e.g., terminal workers, truckers) and inland haulage capacity (e.g., trucks, railcars and chassis), lack of storage space, importers unwillingness to collecting containers, etc.

It is well acknowledged that reversing the congestion cannot be achieved overnight and it requires the joint effort and cooperation of the channel members in the container shipping supply chain. The work in the thesis does help to relieve port congestion because of the improved efficiency in import container retrieval through optimisation. Besides the prescriptive analytics methods, strategic-level proactive and reactive strategies are much required to mitigate the root causes of congestion and increase the port and supply chain resilience. Based on the author's knowledge, the ideas borrowed from existing literature and inspired by industry practices, the following potential solution methods are envisioned from four angles of goals (some are overlapping to some extent). Examinations of their practical viabilities are needed and may require the development of innovative research methods.

(i) Reduce container dwell time at terminal yards

- Impose incentive and penalty mechanisms on the storage pricing strategies of containers to encourage customers to collect inbound containers in time and deliver outbound containers at right time. In this regard, off-dock container yards or remote container yards can be largely used to control the inventory level of containers in the terminal yard (see e.g., Woo et al., 2016). Unlike studying the storage pricing from the perspective of competing relationships between terminal operators and remote container yard operators (e.g., Lee and Yu, 2012), collaborated and cooperated modes (e.g., Zhang et al., 2020) need to be investigated in order to achieve the incentive effects, whereas the profit-making goal as commonly targeted in the existing literature may not be primary in this context.
- Call for subsidy policies from the government to encourage customers to collect their containers from the terminal without delay.
- Restrict outbound container arrivals through a truck appointment system. A new appointment mechanism is needed to induce customers to deliver outbound containers to the terminals according to their priority of shipment.
- Skip the storage phase in the yard by direct transhipment. Direct transhipment is a new transhipment mode at container terminals, where arrival containers are directly loaded onto another transport mode instead of being unloaded to the container yard for temporary storage. Its concept is similar to the

cross-docking at a logistic distribution centre (Chiarello et al., 2018) where the arrival and departure of the carriers are synchronised to achieve low or zero inventory. Compared to the conventional indirect-transhipment mode, the direct-transhipment mode could save yard resources but require more restrictions on the berthing time of vessels due to the need for synchronising container arrivals and departures. The direct-transhipment mode has been taken into consideration for vessel-to-vessel transhipment (see Zeng et al., 2017; Monaco and Sammarra, 2018) and vessel-to-train transhipment (Yan et al., 2020), but it is still a relatively new research stream in the literature of terminal operations management. Facing the challenge of port congestion, direct transhipment can be propelled to carry hinterland transport. In this aspect, using barges and trains are considered to be more beneficial and viable compared to road trucks given their higher capacity and more controllable and predictable arrival times, implying easier coordination with deep-sea vessels. From the operations process perspective, the coordination of the schedule and operation plan (e.g., berthing plan, vessel stowage plan, container unloading and loading plan, train arrival plan) between deep-sea vessels and barges/trains is needed. From the commercial process perspective, the barrier of custom clearance constraints needs to be removed.

(ii) Expand terminal capacity

- Design higher and more compact rack-based yard storage systems, such as containers racks, ultra-high container warehouse, and container tower (see Gharehgozli et al., 2020), which can enlarge yard storage capacity substantially and eliminate the need for relocation. Whether such systems can become profitable in the long term need to be investigated.
- Build or lease new off-dock warehouses in proximity to ports as buffer capacity to hedge unprecedented container volumes. Demand forecasting may be needed to determine the leasing contract. Charging policies to customers may need to be established.
- Establish collaborations and integrate with the terminals/ports within the same port cluster to jointly hedge the risk of shutdowns in one or more terminals (e.g., a month-long shutdown at Yantian terminal due to Covid-19 outbreak). Revenue-sharing schemes need to be developed.

(iii) Release resources

- Release yard space in advance. Prepare sufficient storage space for the coming huge number of containers by transferring the already stacked containers in the terminal yard to remote container yards. To arrange efficient transferring, it requires that the shipping lines share advance shipment information with the terminal. Coordination with customers is also needed when selecting which containers should be transferred.
- Improve road haulage capacity. The capacity of road haulage can be improved if the turnaround time of trucks at ports can be reduced. In this regard, the measures in point (i) are helpful because a shorter dwell time implies higher efficiency and thus lower truck turnaround times. In addition, promoting a modal shift from road to barges and trains can reduce the pressure on the roads.

(iv) Reduce human intervention

- Accelerate the degree of port automation and digitalisation. This can reduce the reliance on manpower and thus avoid operation suspending caused by human factors. For example, the Yantian terminal in China was partially closed in May this year after a local Covid infection cluster involving port workers.

Finally, it is noted that many of the above solutions are also discussed in normal situations without disruptive events, but the Covid-19 outbreak is urging their development and implementation.

References

Azab, A., & Morita, H. (2021). The Block Relocation Problem with Appointment Scheduling. European Journal of Operational Research. https://doi.org/10.1016/j.ejor.2021.06.007

Boge, S., Goerigk, M., & Knust, S. (2020). Robust optimization for premarshalling with uncertain priority classes. European Journal of Operational Research, 287(1), 191-210.

Borjian, S., Manshadi, V., Barnhart, C., & Jaillet, P. (2013). Dynamic Stochastic Optimization of Reshuffles in Container Terminals. In Manufacturing and Service Operations Management (MSOM) conference.

Borjian, S., Manshadi, V. H., Barnhart, C., & Jaillet, P. (2015). Managing relocation and delay in container terminals with flexible service policies. arXiv preprint 1503.01535.

Carlo, H. J., Vis, I. F., & Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. European journal of operational research, 235(2), 412-430.

Caserta, M., Schwarze, S., & Voß, S. (2011a). Container Rehandling at Maritime Container Terminals. In J. W. Böse (Ed.), *Handbook of Terminal Planning*, Operations research/computer science interfaces series 49 (pp. 247-269). Springer, New York.

Caserta, M., Schwarze, S., & Voß, S. (2020). Container rehandling at maritime container terminals: A literature update. In J. W. Böse (Ed.), *Handbook of Terminal Planning* (2nd ed.), Operations Research/Computer Science Interfaces Series 64 (pp. 343-382). Springer, Cham.

Chang, Y., & Zhu, X. (2019). A Novel Two-Stage Heuristic for Solving Storage Space Allocation Problems in Rail–Water Intermodal Container Terminals. Symmetry, 11(10), 1229.

Chen, L., & Lu, Z. (2012). The storage location assignment problem for outbound containers in a maritime terminal. International Journal of Production Economics, 135(1), 73-80.

Choi, Y. S., & Kim, K. H. (2016). Optimizing Mixed Storage and Re-Marshalling Plans. In H. Kotzab et al. (Eds.), *Dynamics in Logistics*, Lecture Notes in Logistic (pp. 203-213). Springer, Cham.

Chiarello, A., Gaudioso, M., & Sammarra, M. (2018). Truck synchronization at single door cross-docking terminals. OR Spectrum, 40(2), 395-447.

Covic, F. (2017). Re-marshalling in automated container yards with terminal appointment systems. Flexible Services and Manufacturing Journal, 29(3), 433-503.

Covic, F. (2018). A literature review on container handling in yard blocks. In International Conference on Computational Logistics (pp. 139-167). Springer, Cham.

Covic, F. (2019). Container Handling in Automated Yard Blocks: An Integrative Approach Based on Time Information. Springer. https://doi.org/10.1007/978-3-030-05291-1

Dai, Q., & Yang, J. (2020). A Distributionally Robust Chance-Constrained Approach for Modeling Demand Uncertainty in Green Port-Hinterland Transportation Network Optimization. Symmetry, 12(9), 1492.

Davarzani, H., Fahimnia, B., Bell, M., & Sarkis, J. (2016). Greening ports and maritime logistics: A review. Transportation Research Part D: Transport and Environment, 48, 473-487.

Delage, E., & Ye, Y. (2010). Distributionally robust optimization under moment uncertainty with application to data-driven problems. Operations research, 58(3), 595-612.

Esfahani, P. M., & Kuhn, D. (2018). Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. Mathematical Programming, 171(1), 115-166.

Galle, V., Manshadi, V. H., Boroujeni, S. B., Barnhart, C., & Jaillet, P. (2018). The stochastic container relocation problem. Transportation Science, 52(5), 1035-1058.

Gharehgozli, A., & Zaerpour, N. (2018). Stacking outbound barge containers in an automated deep-sea

terminal. European Journal of Operational Research, 267(3), 977-995.

Gharehgozli, A., Zaerpour, N., & de Koster, R. (2020). Container terminal layout design: transition and future. Maritime Economics & Logistics, 22(4), 610-639.

Hakan Akyüz, M., & Lee, C. Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. Naval Research Logistics (NRL), 61(2), 101-118.

Heilig, L., Stahlbock, R., & Voß, S. (2020). From digitalization to data-driven decision making in container terminals. In J. W. Böse (Ed.), *Handbook of Terminal Planning* (2nd ed.), Operations Research/Computer Science Interfaces Series 64 (pp. 125-154). Springer, Cham.

Hussein, M., & Petering, M. E. (2012). Genetic algorithm-based simulation optimization of stacking algorithms for yard cranes to reduce fuel consumption at seaport container transshipment terminals. In 2012 IEEE Congress on Evolutionary Computation (pp. 1-8). IEEE.

Iris, Ç., & Lam, J. S. L. (2019a). Recoverable robustness in weekly berth and quay crane planning. Transportation Research Part B: Methodological, 122, 365-389.

Iris, Ç., & Lam, J. S. L. (2019b). A review of energy efficiency in ports: Operational strategies, technologies and energy management systems. Renewable and Sustainable Energy Reviews, 112, 170-182.

Iris, Ç., & Lam, J. S. L. (2021). Optimal energy management and operations planning in seaports with smart grid while harnessing renewable energy under uncertainty. Omega, 103, 102445.

Jovanovic, R., Tuba, M., & Voß, S. (2019). An efficient ant colony optimization algorithm for the blocks relocation problem. European Journal of Operational Research, 274(1), 78-90.

Karakas, S., Kirmizi, M., & Kocaoglu, B. (2021). Yard block assignment, internal truck operations, and berth allocation in container terminals: introducing carbon-footprint minimisation objectives. Maritime Economics & Logistics, 1-22.

Kim, K. H., & Yi, S. (2021). Utilizing information sources to reduce relocation of inbound containers. Maritime Economics & Logistics, 1-24.

Kim, K. H., Woo, Y. J., & Kim, J. G. (2021). Space reservation and remarshalling operations for outbound containers in marine terminals. Maritime Economics & Logistics, 23(1), 154-178.

Kourounioti, I., Polydoropoulou, A., & Tsiklidis, C. (2016). Development of models predicting dwell time of import containers in port container terminals–an Artificial Neural Networks application. Transportation Research Procedia, 14, 243-252.

Ku, D., & Arthanari, T. S. (2016). Container relocation problem with time windows for container departure. European Journal of Operational Research, 252(3), 1031-1039.

Lee, C. Y., & Yu, M. (2012). Inbound container storage price competition between the container terminal and a remote container yard. Flexible Services and Manufacturing Journal, 24(3), 320-348.

Liu, B., Zhang, Q., & Yuan, Z. (2021). Two-stage distributionally robust optimization for maritime inventory routing. Computers & Chemical Engineering, 149, 107307.

Luo, J., Wu, Y., Halldorsson, A., & Song, X. (2011). Storage and stacking logistics problems in container terminals. OR insight, 24(4), 256-275.

Monaco, M. F., & Sammarra, M. (2018). Skipping the Storage Phase in Container Transshipment Operations. In International Conference on Computational Logistics (pp. 207-221). Springer, Cham.

Moini, N., Boile, M., Theofanis, S., & Laventhal, W. (2012). Estimating the determinant factors of container dwell times at seaports. Maritime economics & logistics, 14(2), 162-177.

Rendl, A., & Prandtstetter, M. (2013). Constraint models for the container pre-marshaling problem. In: G. Katsirelos & C. Quimper (Eds.), Modref 2013: 12th International Workshop on Constraint Modelling

and Reformulation, pp 44–56.

Russell, S.J., & Norvig, P. (2016). Artificial intelligence: a Modern Approach (3rd ed.). Pearson Education, Limited.

Rutherford, D., & Comer, B. (2018). The International Maritime Organization's initial greenhouse gas strategy. TRID. Retrieved September 7, 2021, from https://trid.trb.org/view/1509482.

Sha, M., Zhang, T., Lan, Y., Zhou, X., Qin, T., Yu, D., & Chen, K. (2017). Scheduling optimization of yard cranes with minimal energy consumption at container terminals. Computers & Industrial Engineering, 113, 704-713.

Shang, C., & You, F. (2018). Distributionally robust optimization for planning and scheduling under uncertainty. Computers & Chemical Engineering, 110, 53-68.

Song, D. (2021). A Literature Review, Container Shipping Supply Chain: Planning Problems and Research Opportunities. Logistics, 5(2), 41.

Tanaka, S., & Mizuno, F. (2018). An exact algorithm for the unrestricted block relocation problem. Computers & Operations Research, 95, 12-31.

Tanaka, S., & Takii, K. (2015). A faster branch-and-bound algorithm for the block relocation problem. IEEE Transactions on Automation Science and Engineering, 13(1), 181-190.

Tang, L., Jiang, W., Liu, J., & Dong, Y. (2015). Research into container reshuffling and stacking problems in container terminal yards. IIE Transactions, 47(7), 751-766.

Tierney, K., & Voß, S. (2016). Solving the robust container pre-marshalling problem. In A. Paias et al. (Eds.), *Computational Logistics*. ICCL 2016. Lecture Notes in Computer Science, vol 9855 (pp. 131-145). Springer, Cham.

Ting, C. J., & Wu, K. C. (2017). Optimizing container relocation operations at container yards with beam search. Transportation Research Part E: Logistics and Transportation Review, 103, 17-31.

Tricoire, F., Scagnetti, J., & Beham, A. (2018). New insights on the block relocation problem. Computers & Operations Research, 89, 127-139.

van Asperen, E., Borgman, B., & Dekker, R. (2013). Evaluating impact of truck announcements on container stacking efficiency. Flexible Services and Manufacturing Journal, 25(4), 543-556.

Van Nguyen, T., Zhang, J., Zhou, L., Meng, M., & He, Y. (2020). A data-driven optimization of large-scale dry port location using the hybrid approach of data mining and complex network theory. Transportation Research Part E: Logistics and Transportation Review, 134, 101816.

Wang, Y., Xiao, Z., Huang, Y., Hao, Y., & Gu, T. (2017). Study of Continuous Berth Allocation Algorithm Based on Fairness Maximization. Journal of Engineering Science & Technology Review, 10(5).

Woo, Y. J., Song, J. H., & Kim, K. H. (2016). Pricing storage of outbound containers in container terminals. Flexible Services and Manufacturing Journal, 28(4), 644-668.

Woo, J. K., Moon, D. S., & Lam, J. S. L. (2018). The impact of environmental policy on ports and the associated economic opportunities. Transportation Research Part A: Policy and Practice, 110, 234-242.

Yan, B., Zhu, X., Lee, D. H., Jin, J. G., & Wang, L. (2020). Transshipment operations optimization of sea-rail intermodal container in seaport rail terminals. Computers & Industrial Engineering, 141, 106296.

Zeng, Q., Feng, Y., & Chen, Z. (2017). Optimizing berth allocation and storage space in direct transshipment operations at container terminals. Maritime Economics & Logistics, 19(3), 474-503.

Zeng, Q., Feng, Y., & Yang, Z. (2019). Integrated optimization of pickup sequence and container rehandling based on partial truck arrival information. Computers & Industrial Engineering, 127, 366-382.

Zhang, C., Guan, H., Yuan, Y., Chen, W., & Wu, T. (2020). Machine learning-driven algorithms for the container relocation problem. Transportation Research Part B: Methodological, 139, 102-131.

Zhang, Q., Yang, S., Zeng, Q., & Yu, T. (2020). Storage pricing model of container yards under fluctuating demand. Applied Economics, 52(39), 4223-4235.

Zhao, W., & Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. Transportation Research Part E: Logistics and Transportation Review, 46(3), 327-343.

Zweers, B. G., Bhulai, S., & van der Mei, R. D. (2020). Optimizing pre-processing and relocation moves in the stochastic container relocation problem. European Journal of Operational Research, 283(3), 954-971.