Minimizing Robot Digging Times to Retrieve Bins in Robotic-Based Compact Storage and Retrieval Systems

by

Anni Yue

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, we study storage policies for a robotic-based compact storage and retrieval system (RCS/RS), which provides high-density storage in distribution center and warehouse applications. In the system, items are stored in bins, and the bins are organized inside a three-dimensional grid. Robots move on top of the grid to retrieve and deliver bins. To retrieve a bin, a robot removes all the bins above one by one with its gripper, called bin digging. The closer the target bin is to the top of the grid, the less digging the robot needs to do to retrieve the bin, and the shorter the waiting time at the workstation.

In this thesis, we propose a policy to optimally arrange the bins in the grid while processing bin requests so that the most frequently accessed bins remain near the top of the grid. This improves the performance of the system and makes it responsive to changes in the bin demand. Our solution approach identifies the optimal bin arrangement in the storage facility, initiates a transition to this optimal setup, and subsequently ensures the ongoing maintenance of this arrangement for optimal performance.

We perform extensive simulations on a custom-built discrete event model of the system. Our simulation results show that under the proposed policy more than half of the bins requested are located on top of the grid, reducing bin digging compared to existing policies. Compared to existing approaches, the proposed policy results in a significant reduction in the retrieval time of the requested bins (by at least 30%) and the number of bin requests that exceed certain time thresholds (by more than 50%). Our simulation results also illustrate that the proposed bin arrangement and policy effectively reduce the working time of robots by at least 20% compared to existing policies.

Acknowledgements

Dedication

This thesis is dedicated to my parents and all other family members.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Warehousing and material handling systems are currently adapting to changes in customer purchasing behavior and modern distribution strategies. Online shopping, same- or next-day delivery, store pick-up, metro e-commerce centers (MECs), micro-fulfillment centers (MFCs), and dark stores require warehouses to have high efficiency, high reliability, high storage density, high energy efficiency, and low cost, which overlap with the well-known objectives of automated storage and retrieval systems (AS/RS). Among various types of AS/RS, robot-based compact storage and retrieval systems (RCS/RS) stand out as a great solution for storing small or miniature items in limited space, providing high throughput, and maintaining order-picking accuracy.

RCS/RS, such as the systems of AutoStore [1] and Ocado [44], store small items in standardized bins organized in a three-dimensional grid (see Fig. 1.1). Robots work on top of the grid to retrieve the bin containing the requested item and deliver the bin to the workstation (goods-to-person or GTP), where a worker (human or machine) completes the order-picking or storage-replenishing tasks. The bins returned from workstations are placed on top of the grid, which makes bins with high-demand items move to the top of the grid and bins with low-demand items sink to the bottom. AutoStore [2] has successfully deployed its system for companies including PUMA and Best Buy in the United States, MS Direct in Switzerland, Elotec and XXL in Norway, Siemens in Germany, and Decathlon in Canada. An online survey [20] of 64 participants (AutoStore users) provides the system design parameters, discusses the advantages and disadvantages of the system, describes the factors that need to be considered in each phase of the system (planning, realization and operation), etc.

Existing research on RCS/RS has evaluated storage policies with different types of
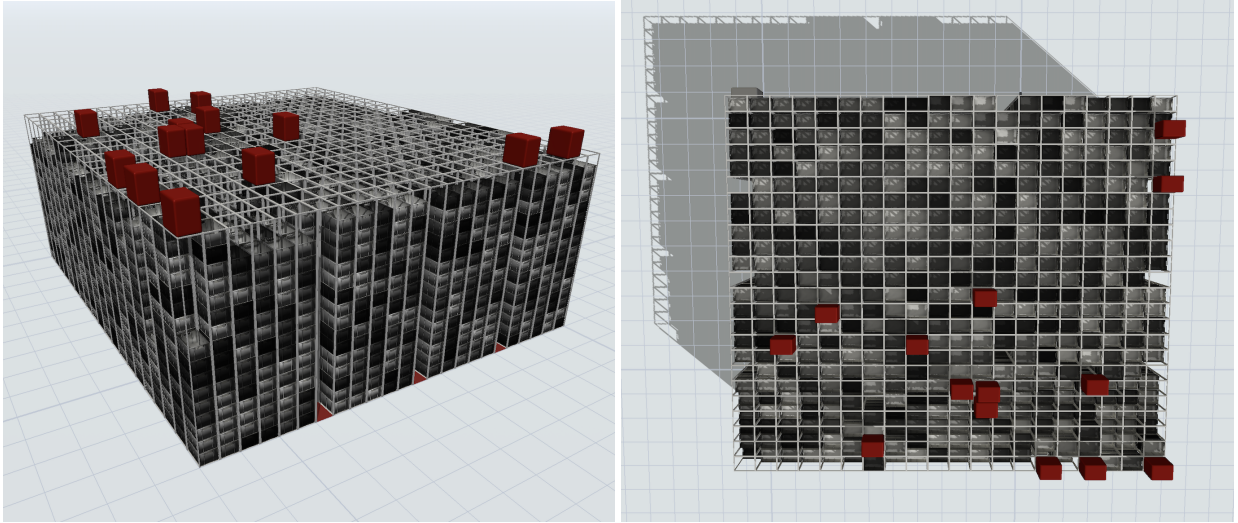
(a) Oblique view   (b) Top view

Figure 1.1: Robotic-Based Compact Storage and Retrieval System

storage stack and reshuffling methods for a given inventory and system configuration [66], and has investigated policies to add new bins to the existing system [7]. However, a key challenge that remains unaddressed is how the system should respond when there is a sudden change in the demand for bins, i.e., how often each bin is requested. Such changes can occur due to seasonal turnover of products or promotional activities and sales. Our objective is to ensure high throughput of the system without suspending order picking. To achieve this goal, we study the optimal arrangement of bins in the grid and propose methods that incrementally transition an initial arrangement to an optimal one while processing bin requests.

1.1 Thesis Contributions

The key contributions of this thesis are three-fold:

1. Given the demand level of each bin, we derive the optimal bin arrangement within the grid, which guarantees the minimum expected time to retrieve one bin from the system.

2. On the basis of the optimal bin arrangement, we build a set of target bin arrangements

within the grid, which reduces the time to retrieve a sequence of bins from the system (with returns).

3. We propose a policy that selects storage locations in the grid for bins returned from workstations, which asymptotically transforms any given initial bin arrangement into one that is in the set of target bin arrangements in the grid.

We build discrete event simulation models to validate the proposed approach and compare it with two existing methods [66]. The simulation results show that our approach outperforms these existing methods in terms of less digging work, less time to retrieve the requested bins, and less work time for the robots to complete the same amount of bin requests.

1.2   Thesis Outline

Chapter 2 reviews the relevant research literature that relates to automated storage and retrieval systems. In Chapter 3, we present a detailed description of robotic-based compact storage and retrieval systems (Section 3.1), and we formalize the system, explain the notation used in the thesis, and define bin grid configuration problems (Section 3.2). In Chapter 4, we derive optimal configurations of the bins in the grid (Section 4.1) and propose a policy (Section 4.2) to minimize the retrieval time of the bins. In Chapter 5, we introduce our simulation environment, evaluate the performance of the proposed policy, and compare our solution with two existing policies. Finally, we draw conclusions and discuss the possibilities of future research in Chapter 6.

Chapter 2

Literature Review

In this section, we start with a cursory history of the development of the automated storage and retrieval systems (AS/RS) in Section 2.1. In Section 2.2, we introduce some widely applied AS/RSs and talk about their advantages and drawbacks. Following that, we present extensive existing studies on robotic mobile fulfillment systems (RMFS) in Section 2.3 and autonomous vehicle-based storage and retrieval systems (AVS/RS) in Section 2.4. In Section 2.5, we go through the studies on the robotic-based compact storage and retrieval system (RCS/RS) that we are able to find, and highlight the research gap in RCS/RS.

2.1 The Development of AS/RS

The world of warehousing and logistics started seeing automation in the early 1950s in a German company Demag [28]. Over the past few decades, with the rapid development of technology, the material handling industry has continued to develop AS/RS as more efficient, intelligent, and cost-effective. Comprehensive surveys [47] and [19] introduce and conclude the developments and characteristics of a variety of AS/RS. Their work provides valuable information on the advances, features, and functionalities of different types of AS/RS. In [45], authors complete a systematic review of the rapid growth of last-mile logistics and provide requirements and directions for further research in the field of material handling. Meanwhile, a variety of optimization methods have been developed for order selection planning to deal with different types of AS/RS [56].

AS/RS were originally designed to handle bulky cargoes [22]. Today, the systems have been improved and developed in variant subdivisions, allowing them to effectively manage

4

storage units of different sizes at different stages of the logistics process. In [3], the authors review the development of modern AS/RS and emphasize the trend of using robotic technology in distribution centers. Their work highlights autonomous vehicle-based storage and retrieval systems (AVS/RS, which are aisle-based shuttle systems), robotic-based compact storage and retrieval systems (RCS/RS, which are grid-based shuttle systems), and robotic mobile fulfillment systems (RMFS) in new robotized warehouses and identifies gaps for further research.

2.2   Types of AS/RS

Depending on how a system delivers the storage item, we classify AS/RS into three categories:

Tray-based: the tray storing the required item is moved to the operator.

Container-based: the container storing the required item is delivered to the operator by robots (e.g., shuttles).

Rack-based: the rack storing the required item is delivered to the operator by automated guided vehicles (AGVs) or autonomous mobile robots (AMRs).

Carousel modules and vertical lift models are shelf-based AS/RS. In a carousel module, shelves are linked together to form a closed loop that rotates vertically (see Fig. 2.1a) or horizontally to bring the appropriate shelf containing the required item to the operator (one at a time). The carousel module has been there for decades and has been extensively studied [6], [36], and [46]. Carousels are typically used to store small and medium-sized items. However, due to their low throughput and potential downtime risks, many companies have decided to move away from carousels in recent years.

A vertical lift module (VLM, see Fig. 2.1b) is another shelf-based AS/RS that consists of an inserter/extractor in the center of two shelves to retrieve or store the shelf containing the required item. The throughput of the system is improved compared to carousel modules, where throughput models have been developed to analyze the performance of different VLMs., e.g., single pod in [41] and dual-tray in [12]. In addition, a recent study [43] proposes optimal order batching models that effectively reduce order fulfillment time. This system is particularly suitable for long and narrow inventory spaces. VLM has the same drawbacks as carousel modules, the throughput is limited because it can only work with one tray at a time.
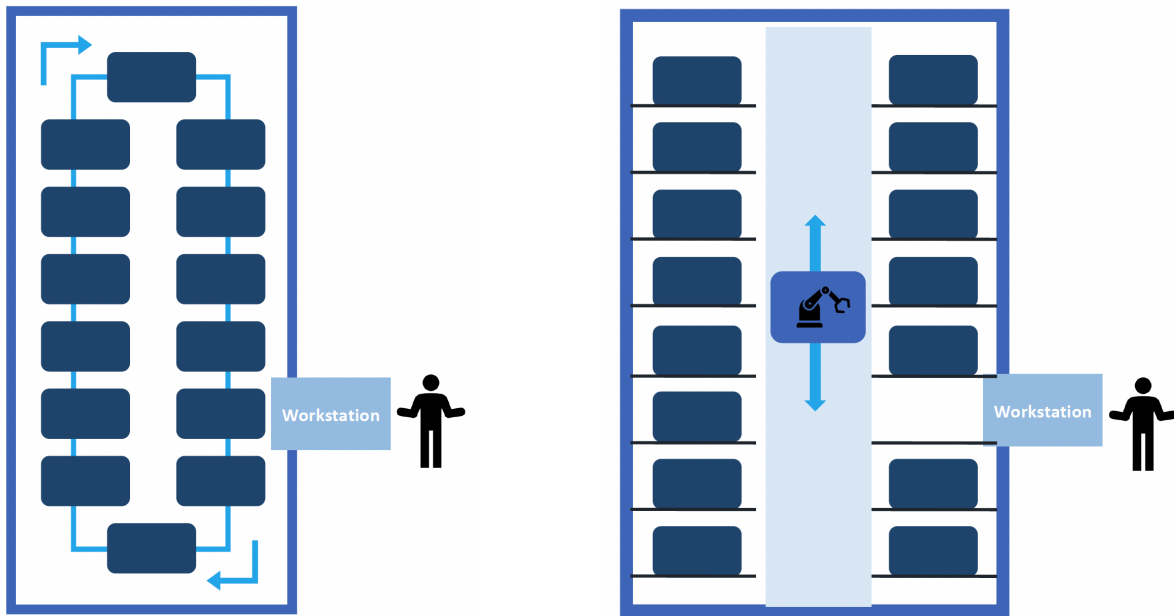
(a) Vertical Carousel Module

(b) Vertical Lift Module

Figure 2.1: Shelf-Based AS/RS
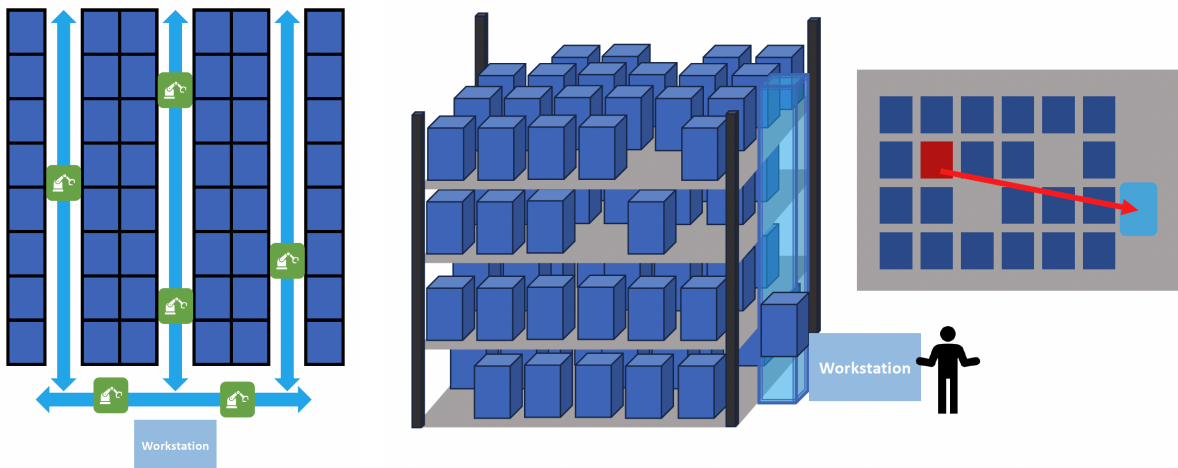
(a) AVS/RS

(b) Live-Cube Storage System with Lift

Figure 2.2: Bin-Based AS/RS
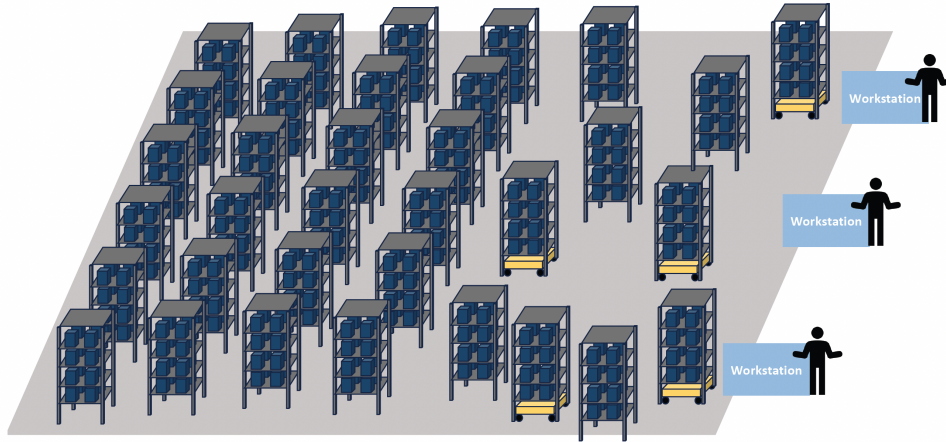
Figure 2.3: Robotic Mobile Fulfillment System

Autonomous vehicle-based storage and retrieval systems (AVS/RS) were introduced in the 1990s to address the productivity limitations of shelf-based AS/RS. AVS/RS use shuttles that travel horizontally (x and y directions) at any level of racks or shelves and lifts that move shuttles vertically between different shelves (see Fig. 2.2a). We present extensive studies on AVS/RS in Section 2.4. The retrieval capacity of such a system can be increased by adding additional shuttles. However, a higher number of shuttles does not guarantee a higher retrieval capacity due to the blocking effect [50]. The main advantage of AVS/RS is its high speed, which makes it widely used in warehousing that requires high throughput. However, high speed requires more energy. In addition, strict initial design and complex integration are required before installation, leading to a long lead time and a high initial investment. Similar to shelf-based AS/RS, AVS/RS are subject to potential downtime risks, i.e. any malfunction or failure of a shuttle device can lead to a shutdown of the entire system.

To compress the required storage space, live-cube storage systems with lift (refer to Fig. 2.2b [63]) and robotic-based compact storage and retrieval systems (RCS/RS, see Fig. 1.1) have been developed. We review related literature on this system in Section 2.5. In Section 3.1, we provide essential details about the specific RCS/RS studied in this thesis. Compared to AVS/RS, RCS/RS has the following main advantages: 1) avoids downtime for the entire system; 2) provides higher storage density; 3) allows for more flexible layouts; 4) allows for easier expansion; 5) requires less initial investment; 6) saves energy.

AVS/RS, live-cube storage systems with lift, and RCS/RS use robots (and lifts) to

deliver containers to workstations, and are bin-based storage and retrieval systems. In contrast to AVS/RS and RCS/RS, robotic mobile fulfillment systems (RMFS) use automated guided vehicles (AGVs) or autonomous mobile robots (AMRs) to move racks that store containers to workstations (see Fig. 2.3). RMFS is a good storage solution for clients such as online retailers, where each order contains only a few items with internal connections (or similarities). The system stores items that are usually ordered together on the same rack, and thus one delivery can retrieve multiple required items, improving the picking speed. In [10], the authors compare the performance of AVS/RS (a miniload system) and RMFS (Kiva system) in simulation and discuss their benefits and limitations. In Section 2.3, we present the literature on RMFS.

2.3   Robotic Mobile Fulfillment Systems

Kiva Systems [23] is a typical RMFS, which is used by clients such as online retailers, where each order contains only a few items with internal connections (e.g., usually ordered together). The system can store such items on the same rack, and thus one delivery can retrieve multiple required items, improving the picking speed. A considerable amount of work has been done to optimize the assignment of robots to workstations [65], storage racks [59], and storage zones [51] within the facility. To improve system performance, models are built to suggest rack layout in the pre-design stage [57] and to assign and sequence orders and racks to workstations [55]. When investigating RCS/RS, we can refer to the above work, since those assignments also happen in RCS/RS. However, RMFS do not fully utilize the vertical space (limited by the height of the rack) and the horizontal space (reduced by floor space for robot movements) in a warehouse, resulting in a lower storage density compared to RCS/RS.

2.4   Autonomous Vehicle Storage and Retrieval Systems

RCS/RS share similarities with AVS/RS, and they face similar challenges. Both AVS/RS and RCS/RS use robots to retrieve and move containers (e.g., bins). The horizontal motions of the robots in both systems are very similar. The robots travel in x and y directions on fixed paths or tracks. Generally, there are multiple robots that work collaboratively in a system to improve throughput. Therefore, both systems require dynamic routing, congestion management, and collision avoidance to improve their reliability and safety within

Table 2.1: Overview of Literature on Design Parameters of AVS/RS

Reference	Design Parameter(s)	Method
[37]	The number of storage columns, tiers, lifts, and vehicles	Markov chain
[17]	Vehicle dwelling point	Decision tree
[31]	Class-based storage policies	Queuing network
[15]	Rack configuration	Simulation-based regression analysis
[13]	Dwell point policy, scheduling rule, input/output (I/O) locations, and interleaving rule	Simulation-based experiment (DOE)
[49]	Configuration of aisles and columns, allocation of resources to zones, and vehicle assignment rules	Queuing network and decomposition-based approach
[40]	Rack configuration, i.e. tier-captive versus tier-to-tier	Simulation-based experiment (DOE)
[48]	Vehicle dwell-point and location of cross-aisles	Queuing network

compact and densely packed structures. The two systems differ in the vertical movements of the robots in the z direction. AVS/RS store bins on shelves and use lifts to move robots between layers, thus robots are able to move on any layers. RCS/RS remove shelves and stack bins on top of each other. To retrieve a bin, a robot needs to use its gripper to remove the bins above the target bin one by one. Due to the large difference, some of the research findings and conclusions about AVS/RS are inappropriate or inapplicable to RCS/RS.

Malmborg proposes Markov chain models to analyze system performance [37]. They carefully explain the design parameters of the system (e.g., the number of storage columns, tiers, lifts and vehicles) and state the evaluators to estimate the performance of the system (e.g., order picking cycle time and vehicle utilization). Subsequent studies have furthered our understanding of AVS/RS by providing more detailed and comprehensive analyses. In the following, we provide a concise overview from two key perspectives: design factors and performance evaluators.

To design an AVS/RS that best fits the operation requirements, a designer should consider design parameters such as storage policies [37], scheduling rules, input/output (I/O) locations for the bins entering and exiting the storage facility, interleaving rules [13], rack configuration [15, 40], dwell point policies, location of cross-aisles [48], number of tires, type of shuttles [52], and system types, i.e., vertical and horizontal [4]. Table 2.1 present a summary of related work.

To effectively assess the performance of an AVS/RS for various purposes, there are

Table 2.2: Overview of Literature on Models for Performance Analysis of AVS/RS

Reference	Model and/or Key Approach	Evaluators to Estimate the Performance
[37]	State equation model	Order picking cycle time and vehicle utilization
[38]	State equation model (extended)	Storage and retrieval cycle times, system utilization, and throughput capacity
[30]	Nested queuing model and queuing approximations	Transaction service times, transaction waiting time, vehicle utilization
[18]	Network queuing model using opportunistic interleaving	Resource utilization, and costs and space requirements
[64]	Dynamically adjust approximation strategies	Transaction waiting times
[27]	Open queuing network (OQN) approach and manufacturing system performance analyser (MPA)	Performance of a system with tier-captive configuration
[39]	Open queuing network approach	Transaction cycle time and waiting times
[14]	Single-class, multiple-server, Semi-Open Queueing Network Model (SOQN)	Throughput of each station, the mean number of jobs each station, mean residence time, and throughput of the entire network
[35]	Analytical travel time model using the probability theory	Travel (cycle) time
[33]	Pragmetric simulation model	Cycle times and throughput performances

a range of performance evaluators to choose from, including order picking cycle time, vehicle utilization [37], storage and retrieval cycle times, system utilization, throughput capacity [38], transaction service times, transaction waiting time [30], resource utilization, costs and space requirements [18], transaction cycle time [39, 35], throughput of each station, mean number of jobs at each station, mean residence time, and throughput of the entire network [14, 33]. Table 2.2 present a summary of related work.

2.5 Robotic-Based Compact Storage and Retrieval Systems

The first AutoStore system appeared in the early 21st century [16]. However, there is limited literature directly related to the system. Most research focused on RCS/RS has been published in the past five years. We see that RCS/RS has garnered increasing attention recently and is one of the main research directions in the future [3].

In [66], an analytical model using reduced semi-open queuing networks (SOQNs) is built to predict the system throughput time, the expected waiting time of orders for robots and

robots for workstations, and the utilization of robots and pickers. They validate the models through simulations, where they compare two storage policies (dedicated and shared), two storage stacks (random and zoned), and two reshuffling methods (immediate and delayed). They use real case data to verify analytic models for the shared storage policy with the random storage stack and immediate reshuffling. They also suggest the optimal length-to-width dimension for each strategy. However, their models have a number of simplifications (e.g., when moving on top of the grid, robots go along with the shortest path without stopping and can cross obstacles), and they do not consider the new bins being added from outside the grid. In [7], the authors propose two input storage policies to add new storage items to the system. They examine their policies in a discrete event simulation environment, where robots use the A* algorithm [24] to plan the path of robots on top of the grid.

The performance of RCS/RS, similar to that of AVS/RS, is based on various design parameters. These parameters include the dimension of the grid, the number of robots and workstations, the order structure, and the storage policy. Due to the complex interactions between these factors, the impact of each individual factor needs to be accurately analyzed through simulation studies. The impact of the number of robots in the system while keeping other factors constant is examined in simulation [21]. The authors find that the number of robots should align with the size of the grid and the number of workstations, as redundant robots can hinder system performance. Additionally, they emphasize that the distribution of bin requests plays a crucial role in determining the suitability of the system. In [54] a discrete event simulation model is developed to analyze the impact of additional design parameters on system performance. The simulation results reveal that 1) higher or fuller stacks result in lower throughput, 2) the location of the requested bin within a stack significantly affects throughput, and 3) the optimal number of robots for a system is determined with great precision by means of simulations.

Some studies explore the broader aspects of RCS/RS operations (i.e., upstream and downstream assignments), including order rearrangement before entering the system, order picking at workstations, and order packing after leaving the system. In [53], the author redesign the AutoStore order processing line to improve productivity (the number of orders per man hour). The proposed improvements include the balance of workload between offline and online workstations, the minimization of order travel distances, the implementation of a cellular layout for similar process procedures, and the integration of automation into the packing process. To effectively minimize the total number of processed bins, a rollout heuristic algorithm was developed to determine the optimal-order processing sequence [29]. In reviewing synchronization difficulties in warehouses with parts-to-picker, [9] conclude that selecting an appropriate workstation configuration can significantly improve through-

put efficiency. At the workstation, AI-picking solutions designed for items stored in small bins [62] and robot arms replace humans to improve picking efficiency and accuracy. With their instance segmentation method, robust and precise masks are generated for storage goods, and they showed that the proposed method outperforms Mask R-CNN [25] for unseen scenarios and outperforms UCN [60] and UOAIS [5] for unseen objects.

Both the studies focused on the system itself and the broader studies related to the system are providing us with an increasingly complete and clear picture of RCS/RS. However, a situation that requires further investigation is how the system should respond to changes in the frequency of requests for bins. In this thesis, we propose a solution for this situation that ensures high throughput and small retrieval times of bins but does not interrupt normal order-picking.

Chapter 3

Preliminaries

3.1 System Description

Section 3.1.1 introduces key components of the specific RCS/RS studied in this thesis. Section 3.1.2 explains the basic operations and the workflow of the system.

3.1.1 Robotic-Based Compact Storage and Retrieval System

The robotic-based compact storage and retrieval system provides a superior level of storage density by eliminating the wasted space of traditional shelving. In the rest of the thesis, we use *the system* to refer to RCS/RS.

Figure 3.1 presents the essential components of the system. All small inventory items are stored in standardized *bins*. The bins are stacked from bottom to top to form a *bin column* (only the bins), and each bin column is organized in a *stack* (the steel structure). The *stack* is a modular vertical frame that maintains only the vertical alignment of the bins, but does not support each bin like a shelf. Each bin space in the stack is a *cell* (normal), and there is a *temporary cell* above each stack. Depending on the shape of the warehouse footprint, the stacks are adjacently aligned to form a *grid* with a flat top.

The grid holds all the components together. At the top of the grid, *rails* are constructed in x and y directions and allow robots to travel to any stack (we call this robot motion the *delivery motion* in the rest of the thesis). The robot accesses a target bin from the top of the stack by removing, if there is any, the bins above the target bin one by one using its gripper (we call this robot motion the *gripper motion* in the rest of the thesis). The dug-up
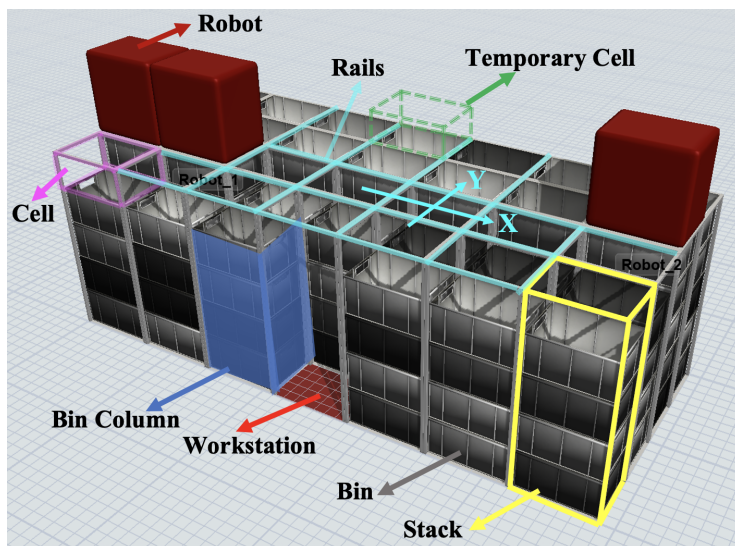
Figure 3.1: Components of robotic-based compact storage and retrieval system

bins are placed on top of nearby stacks that have empty cells (normal and temporary). With the gripper motion and the delivery motion, any robot can access any bin in the grid and deliver the bin to the *workstation*. At the workstation, a worker or robot processes the fulfillment and/or replenishment work.

In this thesis, we study only the system and do not consider the impact of the warehouse management system (WMS). While a WMS enhances the system performance by reordering bin requests before they enter the system, it is important to note that the impact of this optimization varies from case to case. Different WMS use different logic, leading to varying effects on system performance. To isolate this impact, we have the following assumption:

Assumption 1 *The system processes incoming bin requests on a first-come-first-served basis.*

3.1.2 The Working Process

A free robot with no assigned task waits at the top of a stack or workstation where it finished its last task (*dwelling point*) and enters a FIFO queue immediately after finishing its last task. Each time a workstation receives a *bin request*, the system processes the bin

14
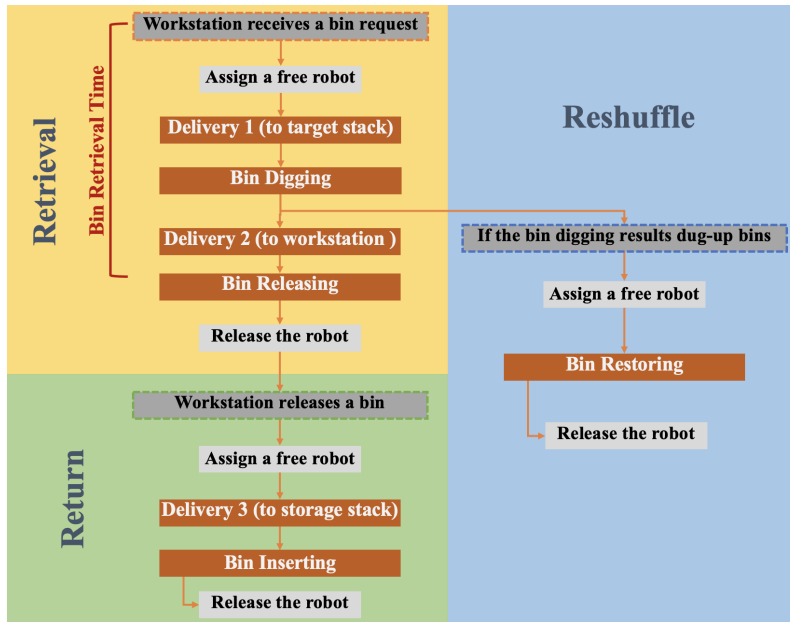
Figure 3.2: Flowchart of the system

request following the workflow presented in Fig. 3.2. Below is a detailed description of each task:

Delivery 1 (to target stack): A free robot moves from its dwelling point to the *target stack* that stores the *target bin*.

Bin Digging: The same robot (in Delivery 1) digs up bins above the target bin one by one, if there are any, and places the dug-up bins on top of nearby stacks (those stacks and dug-up bins are then *blocked*). The robot retrieves the target bin and brings it to the top of the target stack.

Delivery 2 (to workstation): The same robot (in Delivery 1) delivers the target bin to the assigned workstation via the shortest route.

Bin Releasing: The same robot (in Delivery 1) drops the target bin at the workstation.

Bin Restoring: If the bin request induces the dug-up bins in Bin Digging, a free robot leaves its dwelling point and restores the previous dug-up bins to the target stack in the same order. (After restoring the previous dug-up bins from the nearby stacks, these blocked bins and stacks are *freed*.)

15
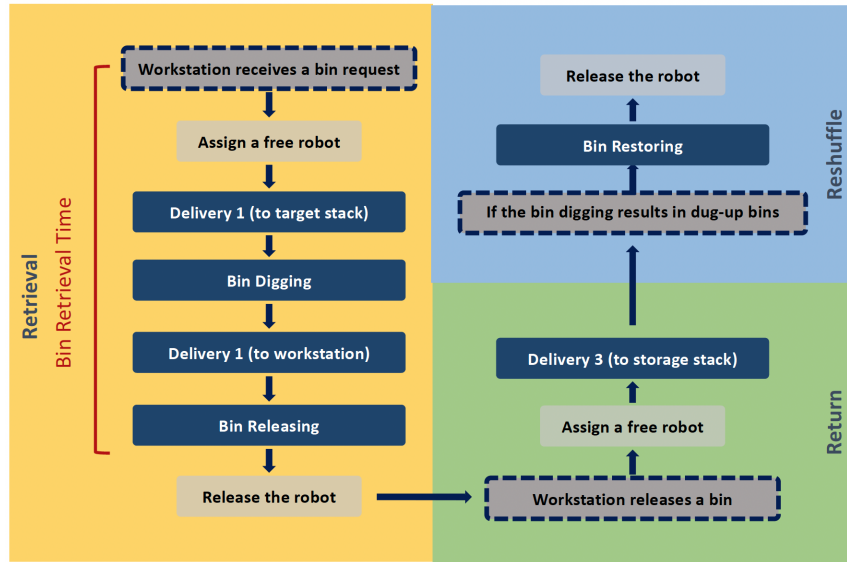
Figure 3.3: Flowchart of the system - Delayed Reshuffling

Delivery 3 (to storage stack): After finishing the work with the target bin at the workstation, a free robot moves from its dwelling point to the workstation, collects the target bin, and delivers the bin to the storage stack of the grid.

Bin Inserting: The same robot (in Delivery 3) drops the target bin on top of the storage stack.

In this mechanism, bins with high demand move to the top of the grid, while bins with low demand sink to the bottom of the grid.

3.1.3 Existing Working Processes

In [66], the authors describe existing storage and reshuffling policies in detail. Depending on the product stored in each stack, they define shared (multiple products stored in each stack) and dedicated (the same products stored in each stack) **storage policies**. On the basis of product demand, they consider random and zoned (all products are classified into three groups by their turnover rate, and each group of products is stored in a predefined location in the grid) **storage stacks**. They also define two **reshuffling methods** for the process of restoring dug-up bins (see Fig. 3.3 presenting the delayed reshuffling and Fig. 3.4 illustrating the immediate reshuffling).

16
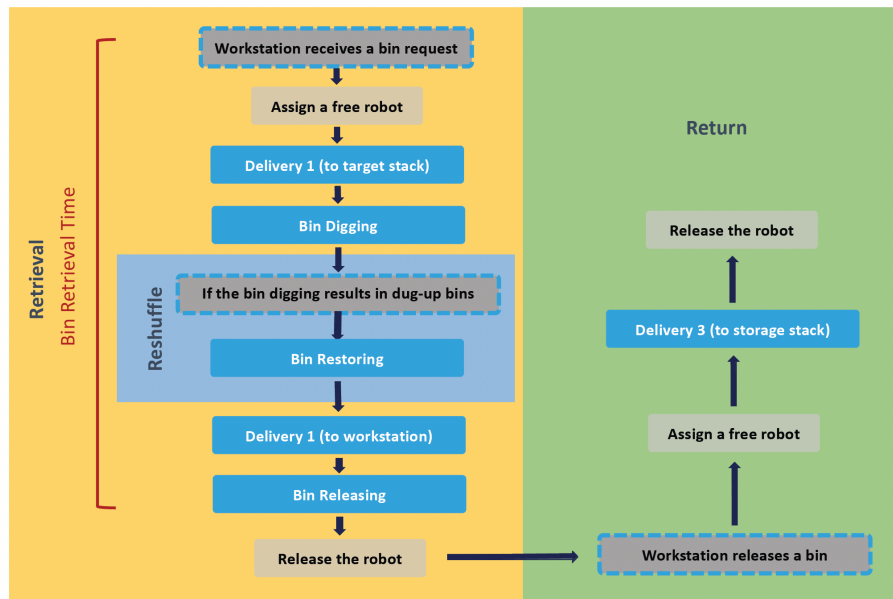
Figure 3.4: Flowchart of the system - Immediate Reshuffling

Combining storage policies, storage stacks, and reshuffling methods, they investigate six combinations: 1) dedicated-zoned, 2) dedicated-random, 3) shared-zoned-delayed, 4) shared-zoned-immediate, 5) shared-random-delayed, and 6) shared-random-immediate. In Chapter 5, we compare our proposed solution with their shared-random-delayed and shared-random-immediate methods. For convenience, we refer to the two existing solutions as the delayed policy and the immediate policy, respectively.

3.2 Problem Formulation

In this section, we first explain and list the essential notations to facilitate the system in Section 3.2.1. Then, we introduce the bin retrieval cost and the expected bin retrieval cost in Section 3.2.2. Ultimately, we define the optimal bin grid configuration problem and the bin rearrangement problem in Section 3.2.3.

Table 3.1: Notations for System Analysis

Notation	Description
$\mathbf{B}$	A feasible BGC, which is an $H \times M$ matrix.
$\mathbb{B}$	Set of all optimal BGCs.
$\hat{\mathbb{B}}$	Set of equivalent optimal BGCs, $\hat{\mathbb{B}} = \cup_{\mathbf{B} \in \mathbb{B}} [\mathbf{B}]$.
$\mathbf{B}^k$	BGC after returning the bin of the kth bin request, $k \in \{0, 1, 2, \dots\}$, where $\mathbf{B}^0$ represents the initial BGC.
$[\mathbf{B}]$	Equivalent class of $\mathbf{B}$.
$[\mathbf{B}]_\varepsilon$	Quasi-equivalent class of $\mathbf{B}$ with respect to ε.
$\mathbf{B}_{l,m}$	The ID of the bin in the cell located in stack m and layer l.
H	Grid height, by the number of cells. Also, the maximum fill level of a stack.
$\overline{H}$	The minimum fill level of a system, $\overline{H} = H(1 - \tau)$.
$\boldsymbol{h}$	Fill level vector of the grid.
h_c	Fill level of occupied stacks within the grid.
h_e	Empty level of occupied stacks within the grid, $h_e = H - h_c$.
l	Layer in the grid, $l \in [\![H]\!]$.
M	Number of the stacks in the grid.
$[\![M]\!]$, m	Set includes all stack IDs, $m \in [\![M]\!]$.
m_f	Number of occupied stacks, $m_f \leq M$.
N	Number of the bins stored in the grid, $N \leq H \cdot M$.
$[\![N]\!]$, n	Set includes all bin IDs, $n \in [\![N]\!]$.
$\Omega(\mathbf{B})$	Function computes the fill level of a BGC $\mathbf{B}$, $\Omega(\mathbf{B}) = h_c$.
p_n	Popularity of bin n, $\sum_{n \in [\![N]\!]} p_n = 1$.
$p_{\mathbf{B}_{l,m}}$	Popularity of the bin stored in stack m and layer l within the grid, $\sum_{l \in [\![H]\!]} \sum_{m \in [\![M]\!]} p_{\mathbf{B}_{l,m}} = 1$.
$\pi_l(\mathbf{B})$	Function computes the probability that the bin is in layer l within a BGC $\mathbf{B}$, $\pi_l(\mathbf{B}) = \sum_{m \in [\![M]\!]} p_{\mathbf{B}_{l,m}}$, $\sum_{l \in [\![H]\!]} \pi_l(\mathbf{B}) = 1$.
Φ	Set of all feasible BGCs.
$\boldsymbol{\sigma}$	Sequence of bin requests $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \dots, \sigma_k, \dots]$, where σ_k represents the bin ID of the kth requested bin.
τ	Percentage of the total cells that were put aside for expansion in the future.
ε	Percentage of popular bins out of all bins.
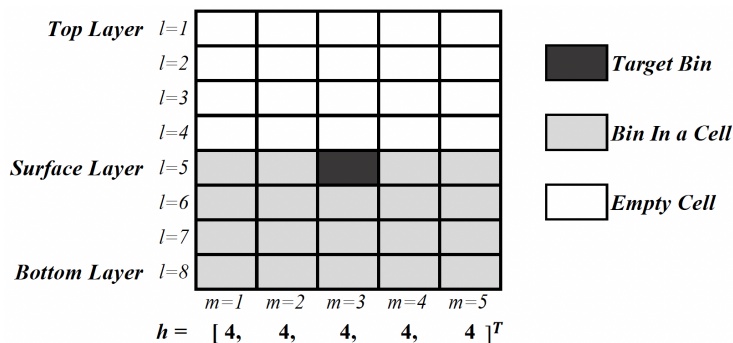
Figure 3.5: Notation of the model - a system consists five stacks (M = 5), has the height of eight (H = 8), offers forty cells available for bin storage (*storage capacity* = 40), and stores twenty bins (N = 20). Each stack has a fill level of four (h_c = 4) and an empty level of four (h_e = 4).

3.2.1 Storage System Model

To improve the performance of the system, we focus on the gripper motion and model the system as follows. Table 3.1 summarizes essential notations of the model (and additional notations we will define and use in the subsequent sections and chapters) and Fig. 3.5 presents an example to help the reader understand the notations. In Section 5.1, we have additional design parameters to build a simulation model that mimics reality. In the thesis, given $X \in \mathbb{N}^+$, we use $[\![X]\!]$ to denote the set of integers ranging from 1 to X, i.e., $[\![X]\!] = \{1, 2, \ldots, X\}$.

Consider a grid consisting of M stacks, and $m \in [\![M]\!]$ represents the stack ID. Each stack offers H cells (i.e., a stack can store up to H bins), and we define H as the *stack height*. We use the term *layer*, $l \in [\![H]\!]$, to locate a cell in a stack, where the top cell of a stack is in layer 1 (*top layer*) and the bottom cell of a stack is in layer H (*bottom layer*). The *surface layer* is the layer where the topmost bin is located. The *storage capacity* of the system, $M \cdot H$, is the total number of cells available for bin storage in the grid.

Within storage capacity, the grid stores N bins (i.e., $N \le M \cdot H$), and $n \in [\![N]\!]$ represents the *bin ID*. The *popularity* of bin n, $0 \le p_n \le 1$, indicates the demand of bin n, i.e., the frequency of bin n being requested. Without losing generality, we normalize the bin popularity and ensure $\sum_{n \in [\![N]\!]} p_n = 1$. Note that we assign the bin ID according to the bin popularity, where the bin with a higher bin popularity is assigned with a smaller bin ID, i.e., $\forall i, j \in [\![N]\!]$ and $i < j$, we have $p_i \ge p_j$. In practice, bin IDs are named according to the management preference.
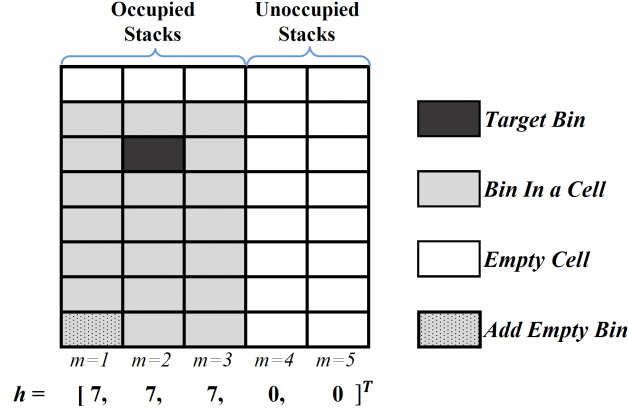
Figure 3.6: Uniformly Occupied Stacks - a BGC that has a fill level of seven ($h_c = 7$), occupies three stacks ($m_f = 3$), and includes one empty bin.

We use the term *fill level*, h_m, to represent the number of bins stored in stack m, where $0 \leq h_m \leq H$. The system has a *fill level vector* $\boldsymbol{h}$, including fill levels of all stacks in the grid, which is defined as $\boldsymbol{h} = [h_1, h_2, \ldots, h_M]^\top$.

Existing work [66] considered the percentage (τ) of the total cells that were put aside for potential expansion in the future, and set the number of cells for bin storage in each stack as an integer $\overline{H} = H(1 - \tau)$. Under their shared storage policy coupled with random storage stacks, $\boldsymbol{h} = [\overline{H}, \overline{H}, \ldots, \overline{H}]^\top$ is the only possible value for a system (see Fig. 3.5).

In our system, we remove the limitation of using all stacks and set the fill level to an integer h_c for occupied stacks, where $\overline{H} \leq h_c \leq H$. We use the term *empty level*, h_e, to represent the number of empty cells in occupied stacks, where $h_e = H - h_c$. When processing bin requests, the initially occupied stacks need to maintain the fill level. For the initially unoccupied stacks, they remain empty all the time. Since our objective is to improve the performance of the system by reducing digging work, and here we consider only the gripper motion in our analysis, the positions of the occupied stacks in the grid do not affect the analysis. Therefore, we have the following remark:

Remark 3.2.1 (Uniformly Occupied Stacks) *(See Fig. 3.6.) The number of occupied stacks is $m_f = \lceil N/h_c \rceil$. For the case when N is not devisable by h_c, we add $m_f \cdot h_c - N$ empty bins. We assign the newly added empty bins the bin ID as $N + 1, \ldots, m_f \cdot h_c$ and set their popularity at 0. Finally, we update N to $m_f \cdot h_c$. Consequently, $\mathbf{h} = [h_1, \ldots, h_{m_f}, h_{m_f+1}, \ldots, h_M]^\top = [h_c, \ldots, h_c, 0, \ldots, 0]^\top$.*

The uniformly occupied stacks structure reduces the time to dig up some bins, but may

need to dig up more bins above the target bin, which creates a trade-off between the average digging time and the number of bins that need to be dug out.

To describe the arrangement of the bins in the grid, we introduce the following terms:

Definition 3.2.1 (Bin Grid Configuration (BGC)) *A bin grid configuration is an $H \times M$ matrix $\mathbf{B}$ where each entry $\mathbf{B}_{l,m}$ gives the ID of the bin in stack m and layer l within the grid. If a cell is empty, then $\mathbf{B}_{l,m}$ is set to 0.*

We use $p_{\mathbf{B}_{l,m}}$ to represent the popularity of the bin stored in the cell of stack m and layer l, where $p_0 = 0$. With all the bins in the grid, $\sum_{l \in [\![H]\!]} \sum_{m \in [\![M]\!]} p_{\mathbf{B}_{l,m}} = 1$.

Definition 3.2.2 (The Set of All Feasible BGCs) *Let $\Phi \subset N^{H \times M}$ be the set of all feasible bin grid configurations. This set contains all matrices $\mathbf{B}$ of the following form:*

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{h_e \times m_f} & \mathbf{0}_{h_e \times (M - m_f)} \\ \mathbf{B}'_{h_c \times m_f} & \mathbf{0}_{h_c \times (M - m_f)} \end{bmatrix}, \tag{3.1}$$

where $m_f = \lceil N/h_C \rceil$ is the number of occupied stacks and $h_e + h_c = H$. Each nonzero entry of $\mathbf{B}'$ is a unique value of $[\![N]\!]$.

Definition 3.2.3 (The Equivalent Class of a BGC) *In the set of all feasible BGCs Φ, we define the equivalence relation $\sim$ as follows: given $\overline{\mathbf{B}} \in \Phi$ and $\mathbf{B} \in \Phi$, then $\overline{\mathbf{B}} \sim \mathbf{B}$ if and only if for each column $i \in \{1, \ldots, m_f\}$ the entries in $\overline{\mathbf{B}}$ and $\mathbf{B}$ are the same (with possibly different ordering). We let $[\overline{\mathbf{B}}] := \{\mathbf{B} \in \Phi : \overline{\mathbf{B}} \sim \mathbf{B}\}$ denote the equivalence class to which $\overline{\mathbf{B}}$ belongs.*

With all the definitions and notation introduced in this section, we can dive into the system performance analysis. Next, we set the evaluation variables and define the problems studied in this thesis.

3.2.2 Costs Associated with Bin Retrieval

In an H height grid, to retrieve a bin from layer l in a target stack m, a robot needs to dig up bins above the target bin one by one and place them on top of the nearby occupied stack(s) $k \in [\![M]\!]/\{m\}$ (see Fig. 3.7). Focusing on the gripper motion and excluding the delivery motion, we decompose the bin retrieval time, C_r, into two parts:
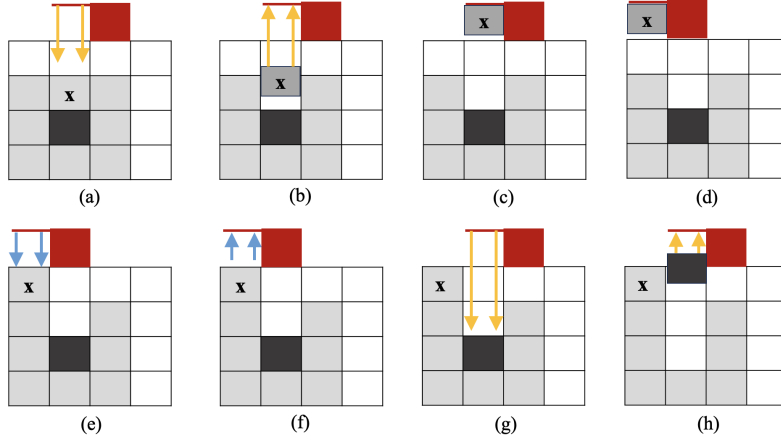
Figure 3.7: Example illustration of the bin retrieval process: grey rectangles represent cells with bins, where the one with a darker colour represents the target bin and the one with an "**X**" mark represents the bin above the target bin; white rectangles represent empty cells; and, the red symbol stands for the robot where the arrows indicate the gripper motion and direction.

1. $C_{r,1}(l, H, h_m)$: the cost of digging up bins from the target stack, i.e., the cost incurred in the target stack (represented by yellow arrows in Fig. 3.7).

2. $C_{r,2}(l, H, h_m, h_k)$: the cost of placing the bins removed above the target bin on top of the nearby stacks, i.e., the costs incurred outside of the target stack (represented by blue arrows in Fig. 3.7).

In addition, all occupied stacks have the same fill level, $h_m = h_k = h_c$. As a result, the bin retrieval cost is computed by adding the two parts:

$$C_r(l, H, h_c) = C_{r,1}(l, H, h_c) + C_{r,2}(l, H, h_c). \tag{3.2}$$

The cost also depends on some other parameters, such as the dimension of the grid (the height of the bin) and the moving speed of the robot gripper. To simplify the analysis, we apply a few approximations when deriving the cost functions in Section 4.1.1.

With a given BGC **B**, we can compute the expected bin retrieval cost for a single bin request $E_1[C_r]$, which is the weighted average of the bin retrieval cost of each bin in the

given BGC, by the following equation

$$
\begin{aligned}
E_1[C_r] &= \sum_{l \in \llbracket H \rrbracket} \sum_{m \in \llbracket M \rrbracket} C_r(l, H, h_c) \cdot p_{\mathbf{B}_{l,m}} \\
&= \sum_{l \in \llbracket H \rrbracket} \left(C_r(l, H, h_c) \cdot \sum_{m \in \llbracket M \rrbracket} p_{\mathbf{B}_{l,m}} \right) \\
&= \sum_{l \in \llbracket H \rrbracket} C_r\Big(l, H, \Omega(\mathbf{B})\Big) \cdot \pi_l(\mathbf{B}),
\end{aligned}
\tag{3.3}
$$

where $\pi_l(\mathbf{B}) = \sum_{m \in \llbracket M \rrbracket} p_{\mathbf{B}_{l,m}}$ is the probability that the target bin is in layer l within the BGC $\mathbf{B}$. Note that $\sum_{l \in \llbracket H \rrbracket} \pi_l(\mathbf{B}) = 1$. Since the fill level h_c is a property of a BGC $\mathbf{B}$, we define a function $\Omega(\mathbf{B}) = h_c$ that computes the fill level of the BGC.

3.2.3   Bin Grid Configuration Problem

In this thesis, our objective is to reduce the bin retrieval time (see Fig. 3.2) by reducing the *digging workload* of the robots, which depends on the number of bins above the target bin and the **digging depth** of the target bin. See Fig. 3.6 where the target bin has a digging depth of three and the number of bins above the target bin is one.

Definition 3.2.4 (Digging Depth) *The digging depth of a target bin is given by the layer it occupies within the grid.*

First, we consider requesting only one bin from the system.

Problem 1 (Optimal Bin Grid Configuration Problem) *Given a grid consisting of M stacks, where each stack stores up to H bins, and a total of $N(\le M \cdot H)$ bins to be stored in the grid, find the BGC that minimizes the expected bin retrieval cost for a single bin request, i.e.,*

$$
\begin{aligned}
\min_{\mathbf{B} \in \Phi} \quad & E_1[C_r] = \sum_{l \in \llbracket H \rrbracket} C_r\Big(l, H, \Omega(\mathbf{B})\Big) \cdot \pi_l(\mathbf{B}) \\
subject\ to \quad & \sum_{l \in \llbracket H \rrbracket} \pi_l(\mathbf{B}) = 1, \\
& \overline{H} \le \Omega(\mathbf{B}) \le H.
\end{aligned}
\tag{3.4}
$$

We call the solution to Eq. (3.4) an optimal BGC*, which is not unique to a system with multiple occupied stacks. Thus, $\mathbb{B} \subset \Phi$, is a set that contains all the solutions to Eq. (3.4).*

If the target bin is not in the surface layer, after returning the bin and placing it on top of a storage stack, the optimal BGC determined by Problem 1 is not maintained, which means that the optimal BGC is not sustainable while fulfilling a series of bin requests. To address this issue, we seek a set of more flexible BGCs that 1) deviate minimally from optimal BGCs and 2) are transformable and maintainable while processing a series of bin requests. Using the equivalence class relation of a BGC (recall Definition 3.2.3 at the end of Section 3.2.1), we define *the set of equivalent optimal BGCs*:

Definition 3.2.5 (The Set of Equivalent Optimal BGCs) *The set of equivalent optimal BGCs $\hat{\mathbb{B}} \subset \Phi$ is a set containing all BGCs that the system aims to transform into, $\hat{\mathbb{B}} = \cup_{\mathbf{B} \in \mathbb{B}}[\mathbf{B}]$.*

Next, we design a policy that transforms any BGC into a BGC that belongs to the set of equivalent optimal BGCs while processing a series of bin requests $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_k, \ldots]$. We use σ_k to denote the bin ID of the kth requested bin. With a slight abuse of notation, we denote the BGC after returning the kth requested bin by $\mathbf{B}^k$ where $\mathbf{B}^0$ is the initial BGC. We have two possible initial situations, $\mathbf{B}^0 \in \hat{\mathbb{B}}$ or $\mathbf{B}^0 \notin \hat{\mathbb{B}}$. To regulate and govern this transformation process, we address and solve the following problem.

Problem 2 (Bin Rearrangement Problem) *Given an initial BGC $\mathbf{B}^0 \notin \hat{\mathbb{B}}$ and a sequence of bin requests $\boldsymbol{\sigma}$, find a policy $\Gamma : \Phi \times [\![N]\!] \to \Phi$ such that*

$$\mathbf{B}^{k+1} = \Gamma(\mathbf{B}^k, \sigma_{k+1}) \quad with \quad k \in \{0, 1, \ldots\}, \tag{3.5}$$

where

1. *there existing a λ such that the policy transforms $\mathbf{B}^0$ into a $\mathbf{B}^\lambda \in \hat{\mathbb{B}}$ in λ bin requests where $\forall k < \lambda$ we have $\mathbf{B}^k \notin \hat{\mathbb{B}}$;*

2. *for every $k \geq \lambda$ the policy ensures that $\mathbf{B}^k \in [\mathbf{B}^\lambda] \subset \hat{\mathbb{B}}$, i.e., $[\mathbf{B}^\lambda]$ and $\hat{\mathbb{B}}$ are each positively invariant under $\Gamma(\cdot)$.*

Because we do not want the rearrangement to disturb the general bin requests or lead to additional robot work, we proceed with the rearrangement when returning the bin from the workstation by selecting an appropriate storage stack. Our strategy to solve Problem 2 is to gradually adjust the bins in each stack and bring the BGC closer to an equivalent optimal BGC (we provide the refined definition of the distance in Section 4.2.2 before we measure it), and the bin retrieval time on average decreases as a result. We solve Problem 1 in Section 4.1 and Problem 2 in Section 4.2.

24

Chapter 4

Solution Method

4.1 The Optimal Bin Grid Configuration

We first derive the equations to compute the bin retrieval cost in Section 4.1.1. Then, we present the solution to Problem 1 in Section 4.1.2.

4.1.1 Cost Function and Approximation

To simplify the calculation, we set the cost of a robot raising or lowering its gripper by one cell height at 1. We also neglect the cost of a robot to load and unload a bin with its gripper, and the cost is the same whether the gripper is carrying a bin or not regardless of the weight of the bin. As a result, the cost is l when a robot:

1. releases its gripper from the top of the grid to a cell in layer l (e.g., see Fig. 3.7(g), where the target bin is located in layer 3, and the robot releases the gripper to the target bin via the yellow down arrow at a cost of 3), or,

2. lifts its gripper from a cell in layer l to the top of the grid (e.g., see Fig. 3.7(b), where the bin x is located in layer 2, and the robot lifts the gripper with the bin to the top of the grid via the yellow up arrow at a cost of 2).

In what follows, we first derive the expressions of $C_{r,1}(l, H, h_c)$ and $C_{r,2}(l, H, h_c)$, respectively. Then, we sum the two results to arrive at the expression for $C_r(l, H, h_c)$. Before

starting the derivations, we specify that the surface layer is located in the layer $h_e + 1$ of the grid. Recall that $h_e = H - h_c$ is the number of empty cells in each occupied stack.

The function $C_{r,1}(l, H, h_c)$ computes the cost of digging up the bins above the target bin and the target bin in the target stack. The cost of removing the topmost bin (in the surface layer $h_e + 1$) from the stack includes releasing the gripper to reach the bin and raising the gripper to bring the bin to the top of the grid, which is $2(h_e + 1)$. Then we add the cost of removing the next topmost bins from the target stack, one at a time, until we get the target bin (in layer l) at the top of the grid. By the summation formula of the arithmetic series, the cost of digging up bins from the target stack is computed by the following equation

$$
\begin{aligned}
C_{r,1}(l, H, h_c) &\equiv C_{r,1}(l, h_e) \\
&= \underbrace{2(h_e + 1)}_{\text{the } 1^{st} \text{ topmost bin}} + \underbrace{2(h_e + 2)}_{\text{the } 2^{nd} \text{ topmost bin}} + \cdots + \underbrace{2(l)}_{\text{the target bin}} \\
&\qquad\qquad \underbrace{}_{\text{where } [h_e+1, h_e+2, \ldots, l] \text{ is an arithmetic series}} \\
&= 2 \underbrace{\frac{\left((h_e + 1) + l\right)\left(l - (h_e + 1) + 1\right)}{2}}_{\text{by the summation formula of the arithmetic series}} \\
&= l^2 + l - h_e^2 - h_e,
\end{aligned}
\tag{4.1}
$$

which is a monotonically increasing function with respect to l.

The function $C_{r,2}(l, H, h_c)$ computes the cost of placing the dug-up bins above the target bin on top of the nearby stack(s). Since we consider a single bin request, every occupied stack, excluding the target stack, has $h_e + 1$ empty cells (the 1 is the temporary cell above a stack) to temporarily store the dug-up bins. The costs of filling these empty cells from the bottom to the top with dug-up bins are $[2(h_e), 2(h_e - 1), \ldots, 2(0)]$, we call it the *C sequence*. We choose to place the $l - h_e - 1$ dug-up bins to fill the closest stack (Manhattan distance), then the next closest stack, etc. Therefore, the cost of placing each of the dug-up bins is a sequence that is formed by cyclically repeating the elements of the C sequence, i.e., $[2(h_e), 2(h_e - 1), \ldots, 2(0), 2(h_e), \ldots]$, and we call it the *D sequence*.

Example 1 (Computation of $C_{r,2}(l, H, h_c) \equiv C_{r,2}(l, h_e)$) *Figure 4.1a presents the BGC before bin retrieval begins. Accordingly, C sequence is $[4, 2, 0]$, where $4 = 2(h_e) = 2(2)$ and D sequence is $[4, 2, 0, 4, 2, \ldots]$. We need to temporarily store the 5 bins, where $5 = l - h_e - 1 =$*
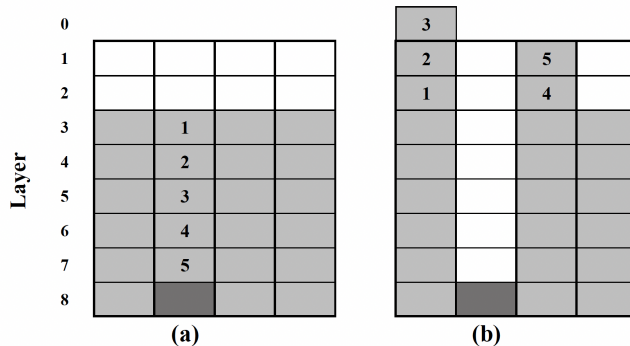
Figure 4.1: Placement of the dug-up bins - A system: $H = 8$, $h_c = 6$, $h_e = 2$, target bin (dark gray rectangle) is located in $l = 8$. Light gray rectangles represent bins and white rectangles are empty cells.

$8 - 2 - 1$. *Figure 4.1b presents the arrangement of the bins after removing and temporarily storing all the bins above the target bin. Consequently, $C_{r,2}(l = 8, h_e = 2) = 4+2+0+4+2 = 12$.*

The result of $C_{r,2}(l, h_e)$ is the sum of the first $l - h_e - 1$ elements of the D sequence. Since each element of the D sequence is a non-negative integer, $C_{r,2}(l, h_e)$ is monotonically non-decreasing with respect to l.

To quickly compute the value of $C_{r,2}(l, h_e)$, we store all possible results in a lookup table (see Table 4.1), and define a function $T(h_e, l)$ that computes the values stored in rows named h_e and columns named l, i.e.,

$$C_{r,2}(l, H, h_c) \equiv C_{r,2}(l, h_e) = T(h_e, l). \tag{4.2}$$

Finally, we compute the bin retrieval cost function by summing Eqs. (4.1) and (4.2) as

$$C_r(l, H, h_c) \equiv C_r(l, h_e) = l^2 + l - h_e^2 - h_e + T(h_e, l), \tag{4.3}$$

which is a monotonically increasing function with respect to l.

Given the optimal BGC with the empty level h_e for each occupied stack, for all $l \leq h_e$, we have $\pi_l(\mathbf{B}) = 0$. Thus, the objective function of Problem 1 is rewritten as

$$E_1[C_r] = \sum_{l=h_e+1}^{H} C_r(l, h_e) \cdot \pi_l(\mathbf{B}). \tag{4.4}$$

27

Table 4.1: LUT of $C_{r,2}(l, h_e)$, where symbol - means the value does not exist.

h_e																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	17	19	20	21	22
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	-	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14	16	16	18	18	20	20
2	-	-	0	4	6	6	10	12	12	16	18	18	22	24	24	28	30	30	34	36	36	40
3	-	-	-	0	6	10	12	12	18	22	24	24	30	34	36	36	42	46	48	48	54	58
4	-	-	-	-	0	8	14	18	20	20	28	34	38	40	40	48	54	58	60	60	68	74
5	-	-	-	-	-	0	10	18	24	28	30	30	40	48	54	58	60	60	70	78	84	88
6	-	-	-	-	-	-	0	12	22	30	36	40	42	42	54	64	72	78	82	84	84	96
7	-	-	-	-	-	-	-	0	14	26	36	44	50	54	56	56	70	82	92	100	106	110
8	-	-	-	-	-	-	-	-	0	16	30	42	52	60	66	70	72	72	88	102	114	124
9	-	-	-	-	-	-	-	-	-	0	18	34	48	60	70	78	84	88	90	90	108	124
10	-	-	-	-	-	-	-	-	-	-	0	20	38	54	68	80	90	98	104	108	110	110
11	-	-	-	-	-	-	-	-	-	-	-	0	22	42	60	76	90	102	112	120	126	130

4.1.2 The Optimal Bin Grid Configuration

We find the optimal BGCs that solve the Problem 1 have the following property.

Proposition 1 (Property of Optimal BGCs) *In an optimal BGC, $\forall l_a, l_b \in \{h_e+1, \ldots, H\}$, where $h_e + 1$ is the surface layer of the BGC, and $l_a < l_b$, then the popularity of any bin stored in a cell in layer a must be greater than or equal to the popularity of any bin stored in a cell in layer b, i.e., $p_{\mathbf{B}_{l_a,m}} \geq p_{\mathbf{B}_{l_b,m}}$.*

Proof:[By Contradiction.] We first prove that the most popular m_f bins are on the surface layer, where m_f is the number of occupied stacks. Given a $\mathbf{B}^*$ that has the property of optimal BGCs, we denote its expected cost as $E_1^*[C_r]$. In $\mathbf{B}^*$, layer $h_e + 1$ is the surface layer that stores the m_f most popular bins, and so on for deeper layers.

Assume another $\mathbf{B}$ that achieves the minimum expected bin retrieval cost for a single bin request, but conflicts with the property of optimal BGCs. We denote its expected cost as $E_1[C_r]$. The layer $h_e + 1$ and layer $q \in \{h_e + 2, \ldots, H\}$ of $\mathbf{B}$ are different from that of $\mathbf{B}^*$ (see Fig. 4.2) and given $p_\alpha > p_\beta$, i.e.,

$$\begin{cases} \pi_l(\mathbf{B}) \neq \pi_l(\mathbf{B}^*), \ \forall l \in \{h_e + 1, q\}, \\ \pi_l(\mathbf{B}) = \pi_l(\mathbf{B}^*), \ \forall l \in \{h_e + 1, \ldots, H\}/\{h_e + 1, q\}. \end{cases} \tag{4.5}$$
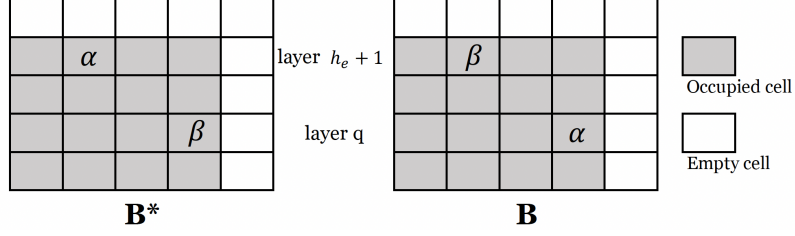
Figure 4.2: For the proof of the optimal BGC

We calculate the difference in expected costs for the two BGCs as follows:

$$E_1^*[C_r] - E_1[C_r]$$

$$= \sum_{l=h_e+1}^{H} C_r(l, h_e) \cdot \pi_l(\mathbf{B}^*) - \sum_{l=h_e+1}^{H} C_r(l, h_e) \cdot \pi_l(\mathbf{B})$$

$$= \sum_{l=h_e+1}^{H} C_r(l, h_e) \cdot \left(\pi_l(\mathbf{B}^*) - \pi_l(\mathbf{B})\right)$$

$$= C_r(h_e + 1, h_e) \cdot \left(\pi_{h_e+1}(\mathbf{B}^*) - \pi_{h_e+1}(\mathbf{B})\right) + C_r(q, h_e) \cdot \left(\pi_q(\mathbf{B}^*) - \pi_q(\mathbf{B})\right) \qquad (4.6)$$

$$= C_r(h_e + 1, h_e) \cdot (p_\alpha - p_\beta) + C_r(q, h_e) \cdot (p_\beta - p_\alpha)$$

$$= \underbrace{\left(p_\alpha - p_\beta\right)}_{[> 0,\ \text{since}\ p_\alpha > p_\beta.]} \cdot \underbrace{\left(C_r(h_e + 1, h_e) - C_r(q, h_e)\right)}_{[< 0,\ \text{since}\ C_r\ \text{is an mono increasing equation.}]}$$

$$< 0$$

$$\Rightarrow E_1^*[C_r] < E_1[C_r],$$

which conflicts with our assumption. Thus, in the optimal BGC, the most popular m_f bins must be in the surface layer. We use this process to gradually verify bins in deeper layers. □

We define the expected bin retrieval cost for a single bin request corresponding to a given empty level h_e as $E_1[C_r]_{h_e}$. With the property of optimal BGC and Eq. (4.4), we calculate the minimum expected bin retrieval cost $E_1[C_r]_{h_e}$ for a single bin request for all possible empty levels $\forall h_e \in \{0, \ldots, H - \overline{H}\}$. The optimal empty level h_e^* is the h_e that corresponds to the minimum $E_1[C_r]_{h_e}$. Knowing the optimal empty level and the property of the optimal BGC, we solve Problem 1.

However, the optimal BGC is changed and is not optimal if the requested bin is not in the surface layer. As a result, the optimal BGC cannot be maintained while serving a

series of bin requests at all times. On the basis of the optimal BGC, we propose a set of more flexible BGCs that can be maintained while serving successive bin requests in the next section.

4.2 The Layer Group Complete Policy

In Section 4.2.1, we present a method that builds the set of equivalent optimal BGCs. In Section 4.2.2, we propose a policy that solves Problem 2.

4.2.1 The Equivalent Optimal BGCs

Earlier in Section 3.2.3, we defined the set of equivalent optimal BGCs $\hat{\mathbb{B}}$ (Definition 3.2.5) as a set that includes the equivalent classes of all optimal BGCs. By the defined equivalence relation (Definition 3.2.3), a BGC belonging to the set of equivalent optimal BGCs should have the same set of bins in each stack of an optimal BGC. Hence, we construct a such BGC $\hat{\mathbf{B}} \in \hat{\mathbb{B}}$ by deforming an optimal BGC $\mathbf{B}^*$ in the following steps:

1. Based on the $\mathbf{B}^*$, we set up *layer groups* with each layer group including the bins in the layer, $l \in \{h_e + 1, h_e + 2, \ldots, h_e + h_c\}$, and we name the layer groups $1, 2, \ldots, h_c$ correspondingly.

2. Each stack in $\hat{\mathbf{B}}$ contains one bin from each layer group (no specific order required). We call such a stack a *layer-complete* stack.

Lemma 1 (Equivalent Optimal BGC Verification) *If all the occupied stacks of a BGC are layer-complete, then the BGC belongs to the set of equivalent optimal BGCs.*

Proof:[Direct Proof.] Assume that a BGC $\mathbf{B}$ with each occupied stack is layer-complete. We rearrange each stack by stacking the bins in ascending popularity order from bottom to top to form a new BGC $\overline{\mathbf{B}}$, where the resulting $\overline{\mathbf{B}}$ is an optimal BGC (i.e., $\overline{\mathbf{B}} \in \mathbb{B}$). According to the equivalence relation (Definition 3.2.3), $\mathbf{B} \in [\overline{\mathbf{B}}]$. Since $\overline{\mathbf{B}} \in \mathbb{B}$ and by the definition of the set of equivalent optimal BGCs (Definition 3.2.5), $[\overline{\mathbf{B}}] \subset \hat{\mathbb{B}}$. Therefore, $\mathbf{B} \in \hat{\mathbb{B}}$. □
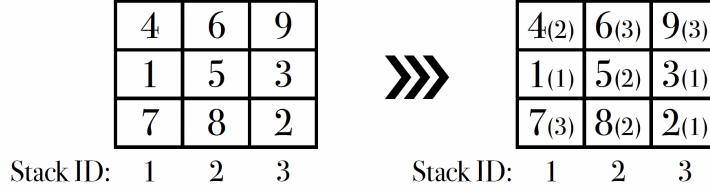
Figure 4.3: Identifying layer-completed stack

Remark 4.2.1 (Repeated Popularity) *If many bins have the same popularity, these bins may be suitable for multiple layers in an optimal BGC. However, when verifying whether a stack is layer-complete or not, a bin should be classified into exactly one layer group.*

Example 2 (Repeated Popularity) *A grid consists of three stacks ($M = 3$), has the height of three cells ($H = 3$), and stores nine bins ($N = 9$, with bin popularity: $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.06$, and $p_4 = \cdots = p_9 = 0.04$). Bins 1,2 and 3 belong to layer group 1, and the rest of the bins belong to layer groups 2 and 3. Figure 4.3 (left) presents a BGC of a system, and the number in each rectangle is the bin ID.*

We need to classify each bin into one layer group and verify whether each stack is layer-complete. Then, we can conclude whether the BGC is in the set of equivalent optimal BGCs or not.

A non-unique result is given in Fig. 4.3 (right), and the number in parentheses indicates the layer group into which the bin is classified. Only stack 1 is layer-complete, while stacks 2 and 3 are not. Thus, the BGC is not in the set of equivalent optimal BGCs.

In the following section, we propose a policy that transforms such a BGC $\mathbf{B} \notin \hat{\mathbb{B}}$ into a BGC $\hat{\mathbf{B}} \in \hat{\mathbb{B}}$ while processing a series of bin requests.

4.2.2 The Layer Complete Policy (LCP)

To transform a BGC that is not in the set of equivalent optimal BGCs into a BGC that is in the set of equivalent optimal BGCs while processing a series of bin requests, we work on selecting appropriate storage stacks for the bins that returned from the workstations. Our policy $\Gamma(\cdot)$ takes the current BGC and bin request as input, includes five possible cases when selecting the storage stack, and outputs the BGC after returning the requested bin back to the grid.
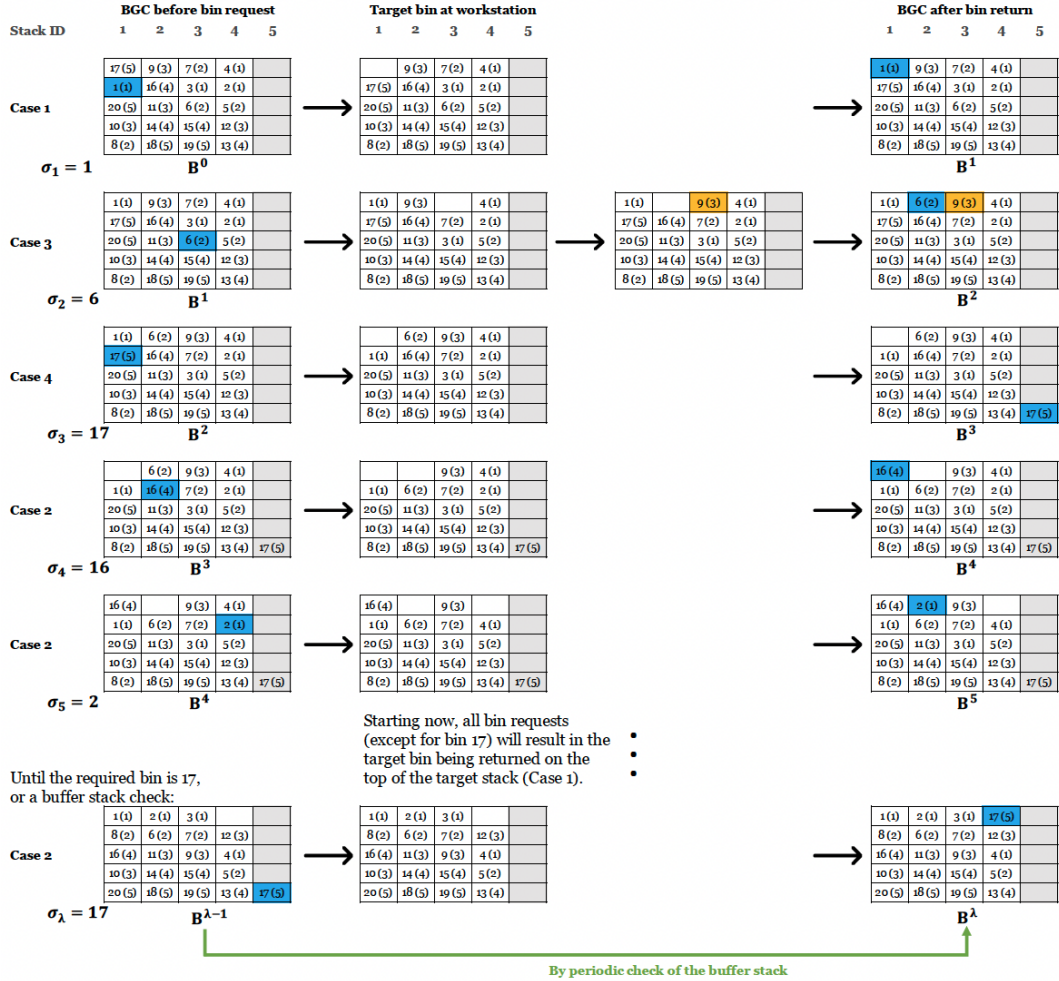
Figure 4.4: A grid consists of four occupied stacks (white) and a buffer stack (gray), has a height of five cells, and stores twenty bins in total. In each small rectangle, the number outside parentheses is the bin ID, and the number inside parentheses is the layer group of the bin. Each figure in the left column is the BGC (together with the request bin are the inputs of $\Gamma(\cdot)$) before a bin retrieval starts, and the blue rectangle in the BGC is the target bin. The figures in the middle are BGCs after removing the target bin from the grid. Each figure in the right column is the BGC after returning the target bin to the grid (i.e., the output of $\Gamma(\cdot)$), and the orange rectangle is the swap bin (in case 3). Black arrows indicate the inputs and outputs of the policy. Green arrows indicate the change due to the periodic check of the buffer stack. Step by step, the proposed policy transforms a random BGC $\mathbf{B^0}$ (on top left) into a BGC $\mathbf{B^\lambda}$ (on bottom right) that is in the equivalent optimal BGCs.

32

In the policy, we need to set aside an empty stack in the grid and name the stack the *buffer stack*. The cases are checked sequentially and described as follows (see Fig. 4.4 for examples):

Case 1: If the target bin is the only bin from layer group x in the target stack, then the target bin is placed on top of the target stack.

Case 2: If there is another stack m that has no bin from layer group x and has empty cells, then the target bin is placed on top of stack m.

Case 3: If there is another stack m that no bin from layer group x and has more than one bin (call them the *swap bins*) from layer group y, and the target stack has no bin from layer group y, then (1) the uppermost swap bin is taken out from stack m and placed on top of the target stack, after, (2) the target bin is placed on top of stack m.

Case 4: If an empty cell is available in the buffer stack, then the target bin is placed on top of the buffer stack.

Case 5: The target bin is placed on top of the occupied stack that has the maximum number of empty cells.

Additionally, the buffer stack is periodically checked, where robots restore the bins from the buffer stack to occupied stacks with empty cells that need a bin from a matching layer group.

Before assessing the effectiveness of the LCP, we clarify two distances:

Definition 4.2.1 (Distance between Two Stacks) *Given a stack m, define a multiset U_m that contains the layer groups of the bins in the stack. The distance between two stacks with stack ID i and j is defined as $D_s(U_i, U_j) = |U_i - U_j| + |U_j - U_i|$, where $|U_i - U_j|$ is the number of elements in U_i that are left over after the elements in U_j have been removed from U_i.*

Example 3 (Distance between Two Stacks) *Given stack i has $U_i = \{1, 1, 1, 2, 5\}$ and the layer-complete stack has $U_* = \{1, 2, 3, 4, 5\}$. Then, the distance between the stack i and the layer-complete stack is $4 = |\{1, 1\}| + |\{3, 4\}|$, where $|\{1, 1\}|$ represents the number of abundant bins with repeat layer group(s) and $|\{3, 4\}|$ represents the total number of bins for the absent layer group(s) in stack i.*

Definition 4.2.2 (Distance between Two BGCs) *Given two BGCs* $\mathbf{B}$ *and* $\overline{\mathbf{B}}$, *the distance between them is defined as the sum of the distances of the corresponding stacks of* $\mathbf{B}$ *and* $\overline{\mathbf{B}}$, *i.e., with a slight abuse of notation, we denote* $U_{m,\mathbf{B}}$ *as the multiset that contains the layer groups of the bins in stack* m *of* $\mathbf{B}$, *then* $D(\mathbf{B},\overline{\mathbf{B}}) = \sum_{m\in[\![M]\!]} D_s(U_{m,\mathbf{B}}, U_{m,\overline{\mathbf{B}}})$.

Given the policy $\Gamma(\cdot)$ (refer to Eq. (3.5)), an initial BGC $\mathbf{B}^0$, a series of bin requests $\boldsymbol{\sigma}$, and the corresponding equivalent optimal BGC $\hat{\mathbf{B}}$, we define *the sequence of distances,* $d_k = D(B^k, \hat{\mathbf{B}})$, representing the distance between the *resulting BGC* after returning the kth requested bin and the corresponding equivalent optimal BGC.

Proposition 2 (The Effectiveness of LCP) *Given any initial BGC and a series of bin requests, under the layer complete policy (LCP), the sequence of distances* d_k *is monotonically decreasing with* k.

Proof:[Direct Proof.] First, we assess each possible change that may happen in a stack by evaluating the impact on the distance between the affected stack and a layer-complete stack:

- Removal of a duplicate layer bin (RmD): if an occupied stack contains more than one bin from a layer group and one such bin is removed from the stack, the distance decreases by 1;

- Insertion of an absence layer bin (InA): if an occupied stack does not contain a bin from a layer group and such a bin is placed on top of the stack, the distance decreases by 1;

- Insertion of a duplicate layer bin (InD): if an occupied stack contains one (or more) bin(s) from a layer group and another such bin is placed on top of the stack, the distance increases by 1;

- Free insertion 1 (FIn1): if an occupied stack contains only one bin from a layer group, and the bin is taken out and then returned to the same stack, then the distance remains the current value; and

- Free insertion 2 (FIn2): if a bin is placed on top of the buffer stack, the distance remains unchanged.

Next, we derive for each policy case the corresponding change in the distance between the resulting BGC and an equivalent optimal BGC:

- Case 1: 0, due to one FIn1;

- Case 2: −2, due to one RmD and one InA;

- Case 3: −4, due to two RmDs and two InAs;

- Case 4: −1, due to one RmD and one FIn2;

- Case 5: 0, due to one RmD and one InD; and

- Periodic check of buffer stack: ≤ 0, by no or a few InAs.

Since the change in distance is non-positive for each case of the policy and the periodic check, the proposed policy will never increase the distance between the resulting BGC and the equivalent optimal BGC. In other words, the policy gradually adjusts the bins in occupied stacks that are not layer-complete, decreasing or maintaining the distance between any stack and a layer-complete stack, and thus the distance between the resulting BGC and an equivalent optimal BGC. □

To reduce the distance between $\overline{\mathbf{B}} \notin \hat{\mathbb{B}}$ and $\hat{\mathbf{B}} \in \hat{\mathbb{B}}$, the policy must trigger one of **Cases** 2, 3, or 4. In **Case** 2, one bin request rearranges one bin that needs to be reorganized (calling such a bin *out-of-place* bin). In **Case** 3, one bin request rearranges two out-of-place bins. In **Case** 4, one out-of-place bin is moved to the buffer stack, and we assume that the periodic check of the buffer stack moves the bin to its right place, thus one bin request adjusts one out-of-place bin. Therefore, each out-of-place bin must be requested at least once to make it *in-place*. Note that the series of bin requests $\boldsymbol{\sigma}$ depends on the demand distributions of the storage items, i.e., the bin popularity p_n.

Claim 1 *Suppose all out-of-place bins have a popularity greater than zero, i.e., $p_n > 0$ where n is the ID of an out-of-place bin. Then, the LCP can transform any initial BGC $\overline{\mathbf{B}} \notin \hat{\mathbb{B}}$ into an equivalent optimal BGC $\hat{\mathbf{B}}$, and the expected number of bin requests to complete the transformation is finite. Thus, the LCP guarantees that any initial BGC converges to an equivalent optimal BGC.*

Proof:[Direct Proof] An initial BGC $\overline{\mathbf{B}} \notin \hat{\mathbb{B}}$ has a total number of x out-of-place bins. We define a vector $\mathbf{p}$ that stores the popularity of out-of-place bins, where $\mathbf{p}(i)$ represents the ith entry of $\mathbf{p}$ and $\mathbf{p}(i) > 0$. We need to retrieve each of the out-of-place bins at least once to transform a $\overline{\mathbf{B}}$ into a $\hat{\mathbf{B}} \in \hat{\mathbb{B}}$, which is a coupon collector problem (CCP) with different probabilities.

Depending on the probability summation of all in-place bins, we have two possible CCP setups:

Setup 1: if $1 - \sum_{i=1}^{x} \mathbf{p}(i) > 0$, then we assume there are $x + 1$ types of coupons, and that each day a collector randomly gets a coupon with probability p'_i corresponding to the ith coupon $(1 \le i \le x + 1)$, where

$$p'_i = \begin{cases} \mathbf{p}(i) & \text{if } 1 \le i \le x \\ 1 - \sum_{j=1}^{x} \mathbf{p}(j) & \text{if } i = x + 1 \end{cases} . \qquad (4.7)$$

Setup 2: if $1 - \sum_{i=1}^{x} \mathbf{p}(i) = 0$, then we assume there are x types of coupons, and that each day a collector randomly gets a coupon with probability p'_i corresponding to the ith coupon $(1 \le i \le x)$, where $p'_i = \mathbf{p}(i)$.

Note that in both setups, $p'_i > 0$ for all i. Then, we submit the values in the equation that computes the expected time (C_j, the number of bin requests in our problem) for a full collection [8] as

$$E[C_j] = \sum_{q=0}^{y=x \text{ or } x-1} (-1)^{y-q} \sum_{|J|=q} \frac{1}{1 - P_J} \quad \text{with} \quad P_J = \sum_{j \in J} p'_j. \qquad (4.8)$$

Since every $p'_i > 0$, we have all $0 < P_J < 1$, which result in $1/(1 - P_J) < \infty$. Thus, $E[C_j]$ is finite, which is the expected time to collect all the $x + 1$ or x coupons. Therefore, with the given condition, the excepted number of bin requests to transform $\overline{\mathbf{B}}$ into $\hat{\mathbf{B}}$ is finite, where the number is smaller than (with Setup 1) or equal to (with Setup 2) $E[C_j]$. $\qquad \square$

The above proof also shows that the requirement that all out-of-place bins have positive popularity is also necessary. If we have any out-of-place bin(s) that has zero popularity, then one of a few p'_i is(are) equal to zero. Therefore, there must exist one (or more) $P_J = 1$ and result in $1/(1 - P_J) \to \infty$. Thus, $E[C_j]$ is infinite. Therefore, the LCP does not guarantee the transformation from any BGC into an equivalent optimal BGC in a finite number of bin requests.

In fact, *popular* bins with higher polarities are requested most of the time, and the rest of the bins (*unpopular*) are requested occasionally. The 80/20 rule - the most popular 20% of bins are retrieved for 80% of order fulfillment - is widely applied in warehouse management [42]. If we focus on the percentage of popular bins out of all bins (ε) for most of the order fulfillment, we only need to ensure that a BGC can be transformed into a *quasi-equivalent optimal BGC* (see Approximation 4.2.1).

Approximation 4.2.1 (Quasi-Equivalent Optimal BGC) *A quasi-equivalent optimal BGC $\tilde{\mathbf{B}}$ has each occupied stack containing one bin from the layer groups $1, \ldots, \lceil h_c \cdot \varepsilon \rceil$.*

36

Consequently, each popular bin has positive popularity and the sum of popularities of the unpopular bins is greater than 0.

Claim 2 *For any given initial BGC $\overline{\mathbf{B}} \notin \hat{\mathbb{B}}$ and a given ε, under the LCP, the expected number of bin requests to transform $\overline{\mathbf{B}}$ into a quasi-equivalent optimal BGC $\tilde{\mathbf{B}}$ is finite.*

The proof is similar to the proof of Claim 1 by setting $\mathbf{p}$ to include the popularity of all popular out-of-place bins and applying setup 1.

After transforming the BGC into a quasi-equivalent optimal BGC, the LCP ensures that every new resulting BGC is still a quasi-equivalent optimal BGC. In other words, the set of all quasi-equivalent optimal BGCs should be positively invariant under LCP. Consequently, we modify the equivalent class of a BGC (Definition 3.2.3) to the *quasi-equivalent class of a BGC* as follows.

Remark 4.2.2 (quasi-equivalent class of a BGC parameterized by ε) *In the set of all feasible BGCs Φ, we define the equivalence relation R_ε as follows: given $\tilde{\mathbf{B}} \in \Phi$ and $\mathbf{B} \in \Phi$, then $\mathbf{B} R_\varepsilon \tilde{\mathbf{B}}$ if and only if for each column $i \in \{1, \ldots, m_f\}$ the $[h_e + 1, \ldots, h_e + \lceil h_c \cdot \varepsilon \rceil]$ entries of $\tilde{\mathbf{B}}$ exist in the corresponding column of $\mathbf{B}$ (with possibly different ordering). We let $[\tilde{\mathbf{B}}]_{R_\varepsilon} = \{\mathbf{B} \in \Phi | \mathbf{B} R_\varepsilon \tilde{\mathbf{B}}\}$ denote the quasi-equivalence class to which $\tilde{\mathbf{B}}$ belongs.*

In summary, by retrieving bins from and returning bins to the grid, the proposed policy (i) reduces or maintains the distance between the resulting BGC after each bin request and an equivalent optimal BGC $\hat{\mathbf{B}}$ and (ii) ensures that any BGC can be transformed into a quasi-equivalent optimal BGC $\tilde{\mathbf{B}}$ in a finite number of bin requests. After the first time the system achieves $\hat{\mathbf{B}}$ or $\tilde{\mathbf{B}}$, the BGC persists in $[\hat{\mathbf{B}}]$ or $[\tilde{\mathbf{B}}]_{R_\varepsilon}$ until there is a change in the distribution of bin popularity. Subsequently, the proposed policy will, again corresponding to the updated bin popularity, transform the current BGC into an updated quasi-equivalent optimal BGC (guaranteed) or/and an updated equivalent optimal BGC (unguaranteed).

Chapter 5

Policy Validation and Comparisons

In this section, we evaluate and compare the proposed policy with the shared storage policy coupled with random storage stacks using delayed and immediate reshuffling methods [66]. Section 5.1 introduces the 3D discrete-event simulation environment built for policy validation and evaluation. We consider scenarios starting with three initial BGCs (one optimal and two non-optimal) and evaluate three global objectives: the digging workload in Section 5.2, the bin retrieval time in Section 5.3, and the robot working time in Section 5.4.

5.1   Implementation Details

We built simulation models in FlexSim (version 23.0.3). The parameters of the storage bin and the robot are shown in Table 5.1. All simulations are carried out using the system parameterized as shown in Table 5.2. The grid uses a rectangular footprint, and each stack has the same number of cells in our model. We use red-floor stacks to represent workstations located on the perimeter of the bottom layer of the system (see Fig. 5.1a). The bin requests are generated by the distribution presented in Fig. 5.1b, which depends

Table 5.1: System Parameters

Storage Bin		Robot			
Length	$0.65\ m$	Top Speed	$3.1\ m/s$	Load	$1.2\ s$
Width	$0.45\ m$	Acceleration	$0.8\ m/s^2$	Unload	$1\ s$
Height	$0.33\ m$	Lift Speed	$1.6\ m/s$	Turn	$1\ s$

Table 5.2: System Setup

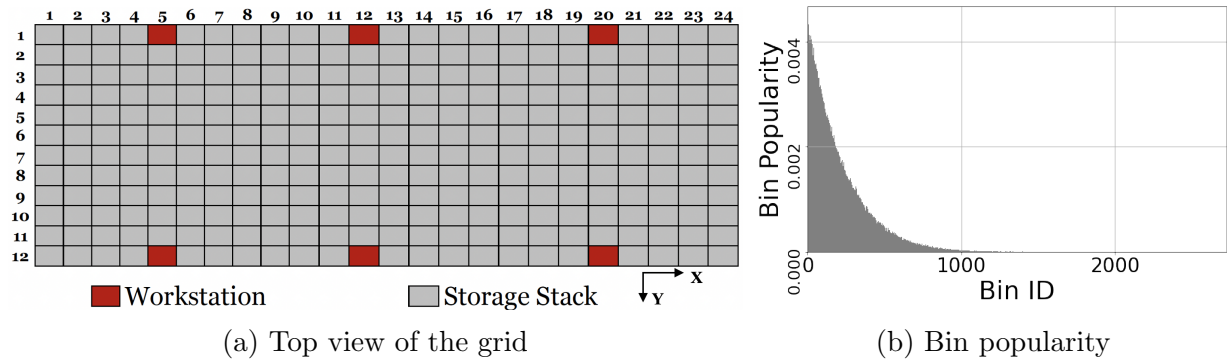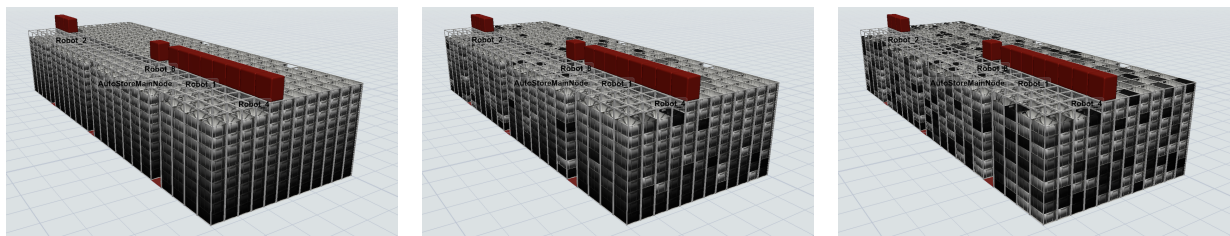| Grid Size (cell)		Number of Components	
Length	24	Robots	12
Width	12	Workstations	6
Height	10	Bins	2730
Orders			
Bin Request Rate			5 *requests/min*
Order Processing Time at Workstation			30 *s/order*

(a) Top view of the grid

(b) Bin popularity

Figure 5.1: Simulation environment setups

(a) 0% randomization from the optimal BGC (BGC/0)

(b) 40% randomization from the optimal BGC (BGC/40)

(c) 100% randomization from the optimal BGC (BGC/100)

Figure 5.2: Initial states with different randomization from the optimal BGC

39

on the popularity of the bins. To exclude the influence of different warehouse management systems (WMS), we do not streamline bin requests (recall Assumption 1).

We generate different initial BGCs for the simulation by taking an optimal BGC and swapping selected pairs of bins. The pairs are selected uniformly at random and without replacement. We denote the initial BGC by the percentage of bins that have been swapped in an optimal BGC. For example, in a system that stores 100 bins, an initial 40% swap corresponds to swapping 20 pairs of bins (40 bins in total). We call this the *randomization from optimal BGC*, with 0% corresponding to the optimal BGC and 100% corresponding to each bin involved in a swap. Figure 5.2 presents three initial BGCs corresponding to the three scenarios discussed in this section. Simulations starting with BGC/40 and BGC/100 show us how the system responds to changes in bin popularity (or demand level for the bins).

We integrate the system with the proposed policy and the delayed and immediate policies (i.e., the shared storage policy coupled with random storage stacks using delayed and immediate reshuffling methods [66]). In each combination of scenarios and policies, we simulate the model for 100 hours and use the result for the following analysis.

This thesis focuses on the effect of digging depth of required bins on bin retrieval time. The location of the workstations is not our object of study, and in our simplified simulation model:

1. from the start point to the destination, robots first run in x direction and then in y direction and do not perform collision avoidance between robots and bins,

2. when a new task arrives, the first available robot in the queue accepts the task but not necessarily the nearest one, and

3. a robot releases the bin at the workstation immediately when it arrives at the top of the workstation.

When evaluating the performance of the system, specifically the digging time to retrieve the target bin, we do not count the time consumption of robots moving from the dwelling point to the task location (e.g., stacks or workstations). These simplifications allow us to run simulations on larger systems. In fact, our simulation results have shown that they do not affect the general observed trends (see the second half of Section 5.2).

Note that deadlock can occur in AS/RS where two or more entities (e.g., robots) become mutually blocked and unable to proceed further, resulting in a halt or significant delay in system operations. This problem has been well discussed in AVS/RS [26, 50, 11]. In
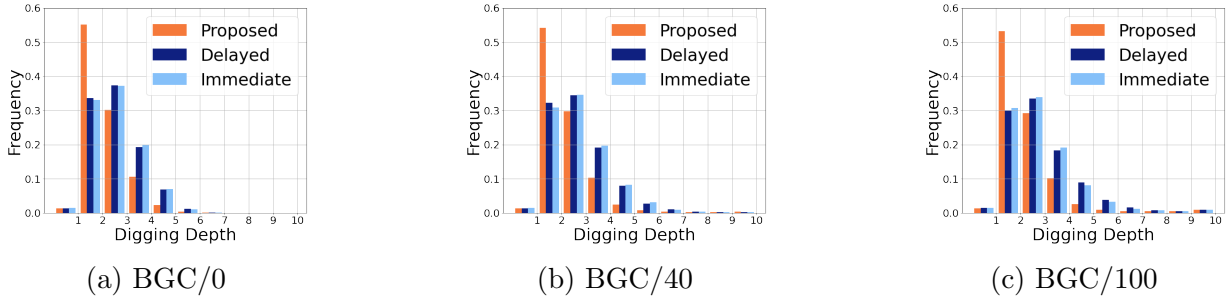
(a) BGC/0   (b) BGC/40   (c) BGC/100

Figure 5.3: Comparisons of the digging depth of the target bins.
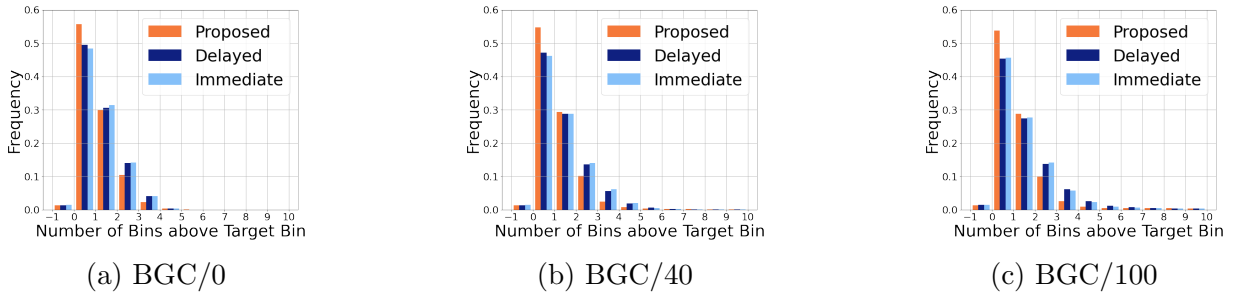

(a) BGC/0   (b) BGC/40   (c) BGC/100

Figure 5.4: Comparisons of the number of bins above the target bins.

RCS/RS, blocked stacks and bins (recall Section 3.1.2) may cause deadlocks in the system. By prioritizing the jobs in order (return > reshuffle > retrieval), we assign free robots to the higher-rank jobs first to prevent the system from deadlock. With the setup and scenarios in our simulations, deadlock does not occur in our model (the result is not specifically discussed in the thesis).

5.2 Digging Workload

In each of the three scenarios (BGC/0, BGC/40, and BGC/100), we compare the digging workload with the three policies. Figure 5.3 (resp. Fig. 5.4) presents the distribution of the digging depth of (resp. the number of bins above) the target bins.

The first group of bars in each histogram encapsulates the target bins that have already been placed at the workstation when requested. Thus, these bin requests do not trigger any robot task. Comparing the target bins that are from the top layer (resp. with zero bin above the target bin), we find that the portion in the proposed policy is at least 65%
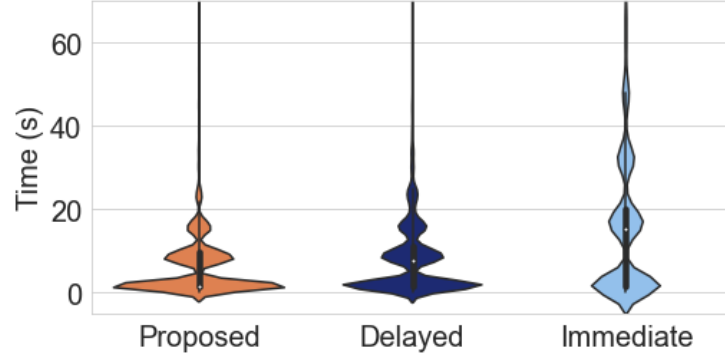
(a) Dig Time Distribution (BGC/40)
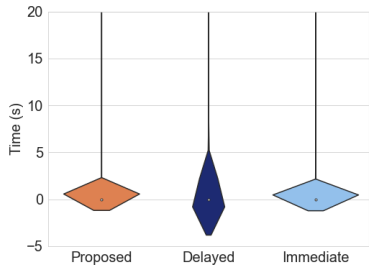
(b) Wait Time Distribution (BGC/40)

(c) Delivery1 Time Distribution (BGC/40)
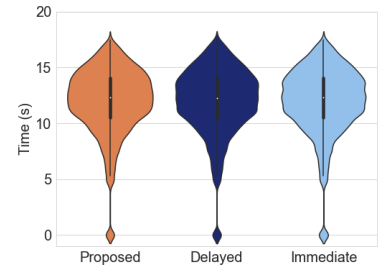
(d) Delivery2 Time Distribution (BGC/40)

Figure 5.5: Comparisons of partitions of the bin retrieval time.

(resp. 16%) greater than in delayed and immediate policies. As should be, the proposed policy has a smaller number of target bins dug up from the deeper layers (resp. smaller number of bins above the target bin) than the other two policies.

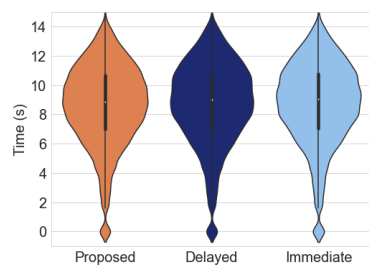By applying the proposed policy, more than 50% of the requested bins are retrieved from the surface layer, which means that robots do less digging work to retrieve the target bins. To support this finding and to verify that the impact of the simplifications made in the model (mentioned in Section 5.1) is negligible, we partition the bin retrieval time into four parts: (1) *waiting time* is the time of a task waiting for a robot to be assigned and the time of a blocked target stack or bin waiting to be freed; (2) *delivery1 time* is the time of a robot moving from the dwelling point to the target stack; (3) *digging time* is the time from the start of removing the first bin out of the target stack to bring the target bin to the top of the target stack; and (4) *delivery2 time* is the time of a robot moving from the target stack to the workstation. The results illustrate that the proposed policy has
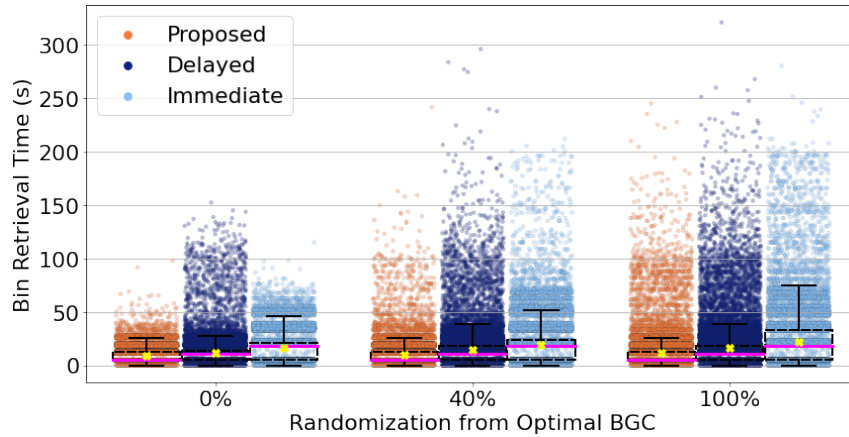
Figure 5.6: The distribution of the bin retrieval time - All Bin Requests

1. a significant advantage in reducing the digging time (see Fig. 5.5a);

2. almost no impact on the waiting time (see Fig. 5.5b); and

3. almost no impact on the delivery motion, i.e., delivery1 time (see Fig. 5.5c) and delivery2 time (see Fig. 5.5d).

In summary, the proposed policy effectively reduces the digging workload by keeping the target bin closer to the top of the grid.

5.3 Bin Retrieval Time

Figure 5.6 presents the bin retrieval time of all bin requests (approximately 30000 bin requests) in the 100-hour simulation for each of the three policies in three scenarios. We observe that when the system uses the proposed policy, the bin retrieval times are more concentrated at lower values in all three scenarios compared to the other two policies. We superimpose each set of data points with a box plot, where the central bright solid line indicates the median, the bottom and top edges of the box indicate the 25th and 75th percentiles, and the whiskers extend to the most extreme data points (1.5 IQR).

First, we look at the interquartile range (25%−75%) of the simulation results presented in Fig. 5.7, which contains more scenarios (i.e., initial BGCs) that have different randomization from an optimal BGC (from 0% to 100% in increments of 10%). In all scenarios,

Figure 5.7: The trend of bin retrieval time – IQR (box), medians (bright solid line), and means ($\mathbf{x}$) – versus the initial state with randomization from the optimal BGC.
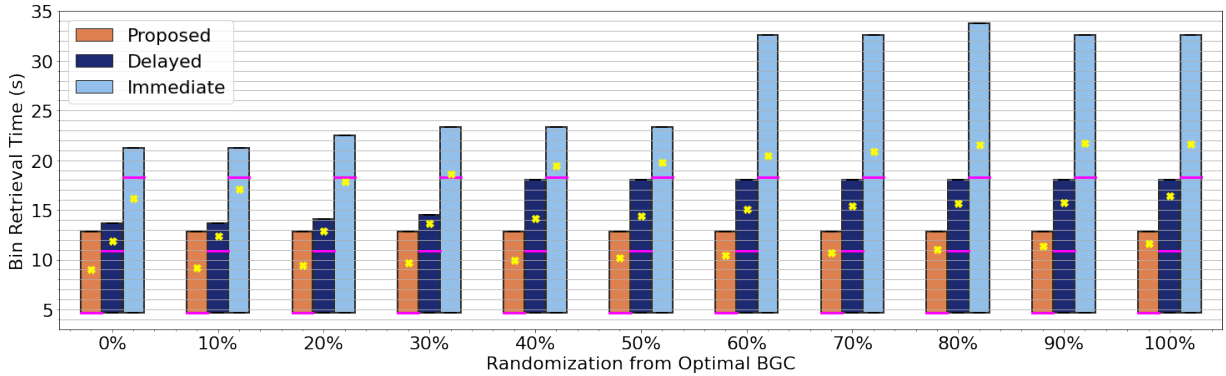
Table 5.3: Improvement in bin retrieval time with the proposed policy

Initial BGC/	Reduce (%) Compare to	Threshold (s)						
		30	40	50	60	70	80	90
0	Delayed	77.64	91.68	96.12	98.46	99.51	99.38	99.13
0	Immediate	95.68	95.79	97.21	95.27	95.83	83.33	60.00
40	Delayed	65.97	69.39	71.98	74.30	74.35	76.82	79.13
40	Immediate	89.41	85.47	85.56	79.18	68.68	68.58	70.07
100	Delayed	56.04	52.60	50.95	49.10	48.12	49.47	56.19
100	Immediate	81.63	70.69	68.18	57.37	47.99	50.83	60.26

the proposed policy results in the smallest IQR, and the IQR remains the same. The IQRs of the delayed and immediate policies extend as the initial randomization from optimal BGCs increases. The median of each policy remains the same in all scenarios. Note that the median for the proposed policy overlaps with the 25th percentile. As the randomness of the initial BGC increases, the mean value under all three policies rises. Among them, the lowest mean value and rate of increase are seen in simulations using the proposed policy. These results indicate that the proposed policy is effective in controlling the bin retrieval time to a small fluctuation, thus ensuring a more stable bin retrieval time compared to the other two policies.

Next, we look at the distributions of longer bin retrieval times ($\geq$ 30s), resulting in a longer waiting time at the workstation. In Table 5.3, the proposed policy reduces the number of bin requests with retrieval time greater than $30s$ (and longer) by at least more
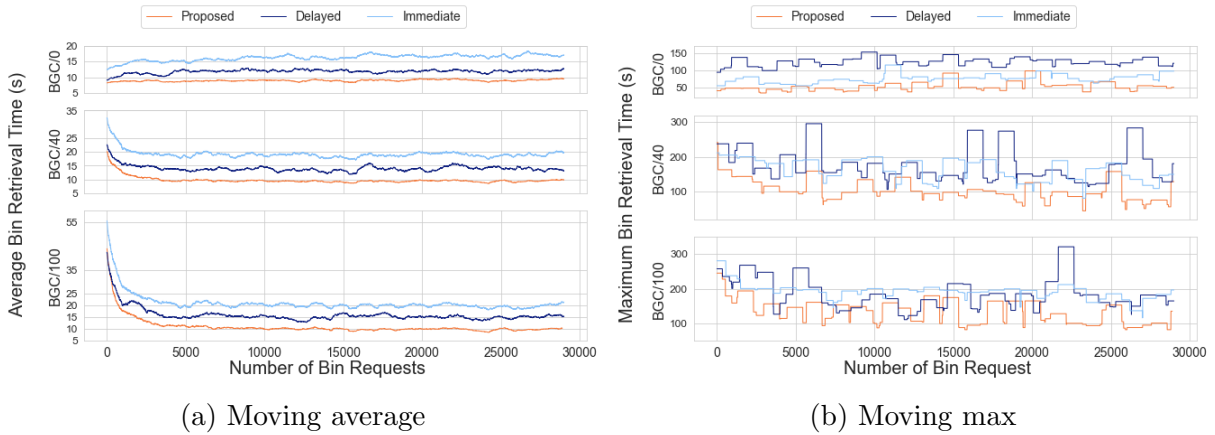
(a) Moving average   (b) Moving max

Figure 5.8: Rolling plots with the smoothing interval of 1000 bin requests

than 50% compared to the other two policies. This result indicates that the proposed policy reduces the frequency of bin retrieval time exceeding certain time thresholds, thus improving service quality.

Figure 5.8a presents the moving average bin retrieval time with the smoothing interval of 1000 bin requests in the 100-hour simulation for the three policies and three scenarios. In the BGC/0 scenario, starting with the optimal BGC, we see a small increase for three policies. The optimal BGC is disrupted by bin requests where the target bin is not in the surface layer, and this results in this small increase. The increase in the line representing the proposed policy is very small as the BGC is transformed among the set of the equivalent optimal BGCs. In BGC/40 and BGC/100 scenarios, starting with a non-optimal BGC that has some popular bins stored near the bottom and some less popular bins near the top of the grid, the initial higher average bin retrieval cost is due to retrieving the deeply buried popular bins. As popular bins move to the top and unpopular bins sink to the bottom while processing bin requests, the average bin retrieval time decreases. With the proposed policy, the BGC transforms into a quasi-equivalent optimal BGC and/or an equivalent optimal BGC, resulting in a stable average time thereafter.

In scenarios BGC/40 and BGC/100 with the proposed policy, our simulation shows that these BGCs are transformed into quasi-equivalent optimal BGCs (with $\varepsilon = 20\%$) for the first time after completing approximately 8000 bin requests, and then are transformed among quasi-equivalent optimal BGCs. This value explains well the lines of the proposed policy shown in Fig. 5.8a, which become stable after the 8000th bin request. However, these BGCs are not transformed into equivalent optimal BGCs until the end of the simulation, as there are some out-of-place bins that have zero popularity (see Fig. 5.1b). We run

simulations with additional scenarios BGC/10 and BGC/20, where we set all out-of-place bins to have non-zero popularity (with the bin ID from $[\![800]\!]$). The initial BGC/10 (resp. BGC/20) is transformed into an equivalent optimal BGC for the first time after completing about 3100 (resp. 6600) bin requests and then is transformed among the set of equivalent optimal BGCs.

An intriguing observation is that the line representing the proposed policy consistently remains below the lines representing the delayed policy (approximately 30% improvement on average) and the immediate policy (approximately 50% improvement on average) throughout the entire plot for any scenario. We also observe that the lines representing the proposed policy stabilized at the same value for the three different initial BGCs. The lines (stabilized part) representing the other two policies increase as the randomization from the optimal BGC increases. The moving maximum bin retrieval time with the smoothing interval of 1000 bin requests presented in Fig. 5.8b indicates that the proposed policy effectively reduces the maximum bin retrieval time for the three scenarios.

In summary, the proposed policy effectively reduces the bin retrieval time. More specifically, the proposed policy improves the performance of the system compared to the delayed policy and the immediate policy in the following aspects:

1. More of the bin retrieval times are clustered in ranges of lower values, and the range (IQR) remains the same for all scenarios.

2. The counts of bin retrieval times that exceed given thresholds (e.g., $\geq 30s$) are significantly reduced.

3. The average bin retrieval times for any small portion within the simulation are always the lowest and stabilize around the same value for different scenarios.

4. The maximum bin retrieval times for any small portion within the simulation are below the lines representing the other two policies for most of the time.

The proposed policy guarantees the transformation of any initial BGC into a quasi-equivalent optimal BGC. The proposed policy guarantees the transformation of initial BGCs with all out-of-place bins having bin popularity greater than zero into an equivalent optimal BGC.

5.4   Robot Working Time

In this section, we evaluate the performance of the proposed methods measured via three metrics: the time of the delivery motion (i.e., horizontal movements of robots on top of the

Table 5.4: Improvement in robot working time with the proposed policy

| Initial BGC | Proposed vs. Delayed (%) | | | Proposed vs. Immediate (%) | | |
	Overall	Delivery Motion	Gripper Motion	Overall	Delivery Motion	Gripper Motion
0	42.99	57.61	35.91	41.70	50.51	38.17
40	35.58	55.07	25.21	34.43	48.76	28.00
100	24.06	48.16	10.22	20.67	39.49	11.55

grid), the time of the gripper motion (i.e., vertical movements of the gripper of robots), and the overall time that is the sum of delivery and gripper times. We compare the simulation result of the proposed policy with that of the delayed policy and the immediate policy, and the results are summarized in Table 5.4. Here, values for the three metrics are expressed as percent improvement relative to the other two state-of-the-art baselines.

We observe that all percentages are positive, which supports the improvement of the proposed policy over the other two policies in simulations with different degrees of randomness of the initial BGC. In the scenario starting with an optimal BGC (BGC/0), the proposed policy keeps the BGC transformed among the set of equivalent optimal BGCs, and we observed a maximum improvement of approximately 42%. Although the improvement decreases as the randomness of the initial BGC increases, the proposed policy continues to reduce the overall working time by almost 20%, which is the minimum improvement in the scenario starting with BGC/100.

These reductions are the result of the reduced digging workload. The fewer bins we need to remove above the target bin and place on top of nearby stacks, the less gripping motion (grab and release bins) and delivery motion (move between stacks) are induced.

In summary, the proposed policy effectively reduces the robot working time compared to the delayed policy and the immediate policy for all scenarios. This saves energy for system operation and follows the trend of zero emissions in AS/RS [34].

Chapter 6

Conclusions and Future Directions

In this thesis, we studied the problem of minimizing the bin retrieval time for a robotic-based compact storage and retrieval system. To achieve the objective, we focused on reducing the digging depth of the requested bins in the grid. We derived the optimal BGC that ensures the minimum expected bin retrieval cost for a single bin request. However, the optimal BGC cannot be maintained when the requested bin is not in the surface layer. Therefore, we constructed a set of equivalent optimal BGCs and a set of quasi-equivalent optimal BGCs, and then proposed the layer complete policy that transforms any BGC into an equivalent optimal BGC (unguaranteed) and a quasi-equivalent optimal BGC (guaranteed) while processing a series of bin requests. The set of equivalent optimal BGCs and the set of quasi-equivalent optimal BGCs are each positively invariant under the proposed policy, which ensures that the proposed structures are maintainable while processing a series of bin requests after being achieved. The proposed solution naturally transforms any BGC into a target structure (an equivalent optimal BGC or a quasi-equivalent optimal BGC) without affecting the normal bin requests, which allows the performance of RCS/RS to be improved in response to changes in the demand for bins.

We compared the proposed policy with the delayed policy and the immediate policy in discrete event simulation models. Our results showed that with the layer complete policy, more than 50% of the required bins are retrieved from the surface layer, effectively reducing the bin retrieval time (approximately 30% less than the delayed policy and 50% less than the immediate policy) and decreasing the number of bin requests that exceed certain time thresholds (improvement of $\geq 48\%$ for each). In addition, the proposed policy outperforms the other two policies in saving robot working time (at least 20% improvement) by completing the same amount of bin requests.

However, the proposed method requires extra memory space to store the layer group information, inducing a cost. Depending on the required throughput, if the most significant goal is to achieve the shortest waiting time, the RCS/RS should employ the proposed method.

6.1 Future Work

In this thesis, we made a few assumptions and simplifications for the RCS/RS in problem formulation (Section 3.2), mathematical analysis (Section 3.2.2) and simulation models (Section 5.1). By considering and integrating the following design factors, we can increase the accuracy of the analysis model, improve the performance of the system, and achieve simulation results that are closer to the real world.

There are many design factors that affect robot delivery motions and further impact bin retrieval time, and we present some ideas for further research on the system:

1. **Workstations -** The number of workstations and the transaction mode of the workstation (e.g., dual-command, single-command, or some innovations) could affect the throughput of the system. Studying the relationship between the location of workstations and bin grid configuration contributes to the analysis of robot delivery motions above the grid, which could help reduce bin retrieval time.

2. **Route Planning -** A more accurate delivery motion combined with intelligent route planning to avoid robot and bin collisions will improve the reliability of the simulation model.

3. **Combining Bin Retrievals -** An interesting and challenging method waiting to be discovered is to retrieve multiple requested bins in one digging. We also believe that by grouping bin requests, robots will work more efficiently. This really depends on the structure of the product, the inner correlation of the products, and the distributions of the bin popularity. To solve this problem, we could refer to the extensive work that has been done in other AS/RS, such as closely storing the products that are usually ordered together (e.g., in the same stack or in adjacent stacks) [58], grouping and resequencing bin requests before they enter the system [61], and intelligently assigning delivery and digging tasks to robots [32].

4. **Assignments to Robots -** Furthermore, grouping the tasks awaiting robots and assigning a bundle of tasks to an appropriate robot is also considered to contribute to improving the performance of the system.

5. **Demand of Bins -** The new distribution of bin popularity, i.e. the demand level for bins, is usually predicted by historical data. We should build more accurate and reliable data models to predict the change in demand. We should also verify to what extent the proposed solution method can tolerate prediction errors.

Robot-based compact storage and retrieval systems are receiving increasing attention from both businesses and academia. We hope that our work provides you with a comprehensive introduction to the system, sparks your interest in studying the system, and contributes to the improvement of the system.

References

[1] AutoStore. AutoStore AS. https://www.autostoresystem.com/, n.d. 1

[2] AutoStore. AutoStore Customer Cases. https://www.autostoresystem.com/cases, n.d. 1

[3] Kaveh Azadeh, René De Koster, and Debjit Roy. Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, 53(4):917–945, 2019. 5, 10

[4] Kaveh Azadeh, Debjit Roy, and René De Koster. Design, modeling, and analysis of vertical robotic storage and retrieval systems. *Transportation Science*, 53(5):1213–1234, 2019. 9

[5] Seunghyeok Back, Joosoon Lee, Taewon Kim, Sangjun Noh, Raeyoung Kang, Seongho Bak, and Kyoobin Lee. Unseen object amodal instance segmentation via hierarchical occlusion modeling. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5085–5092. IEEE, 2022. 12

[6] John J Bartholdi III and Loren K Platzman. Retrieval strategies for a carousel conveyor. *IIE Transactions*, 18(2):166–173, 1986. 5

[7] Michaela Beckschäfer, Simon Malberg, Kevin Tierney, and Christoph Weskamp. Simulating storage policies for an automated grid-based warehouse system. In *Computational Logistics: 8th International Conference, ICCL 2017, Southampton, UK, October 18-20, 2017, Proceedings 8*, pages 468–482. Springer, 2017. 2, 11

[8] Arnon Boneh and Micha Hofri. The coupon-collector problem revisited—a survey of engineering problems and computational methods. *Stochastic Models*, 13(1):39–66, 1997. 36

[9] Nils Boysen, Stefan Schwerdfeger, and Konrad Stephan. A review of synchronization problems in parts-to-picker warehouses. *European Journal of Operational Research*, 2022. 11

[10] Yavuz A Bozer and Francisco J Aldarondo. A simulation-based comparison of two goods-to-person order picking systems in an online retail setting. *International Journal of Production Research*, 56(11):3838–3858, 2018. 8

[11] Héctor J Carlo and Iris FA Vis. Sequencing dynamic storage systems with multiple lifts and shuttles. *International Journal of Production Economics*, 140(2):844–853, 2012. 40

[12] Goran Dukic, Tihomir Opetuk, and Tone Lerher. A throughput model for a dual-tray vertical lift module with a human order-picker. *International Journal of Production Economics*, 170:874–881, 2015. 5

[13] Banu Y Ekren, Sunderesh S Heragu, Ananth Krishnamurthy, and Charles J Malmborg. Simulation based experimental design to identify factors affecting performance of avs/rs. *Computers & Industrial Engineering*, 58(1):175–185, 2010. 9

[14] Banu Y Ekren, Sunderesh S Heragu, Ananth Krishnamurthy, and Charles J Malmborg. An approximate solution for semi-open queueing network model of an autonomous vehicle storage and retrieval system. *IEEE Transactions on Automation Science and Engineering*, 10(1):205–215, 2012. 10

[15] Banu Yetkin Ekren and Sunderesh S Heragu. Simulation-based regression analysis for the rack configuration of an autonomous vehicle storage and retrieval system. *International Journal of Production Research*, 48(21):6257–6274, 2010. 9

[16] Element Logic. The world's first AutoStore-warehouse – still going strong. https://www.elementlogic.net/cases/the-worlds-first-autostore-warehouse-still-going-strong/, n.d. 10

[17] Miki Fukunari, Kristin P Bennett, and Charles J Malmorg. Decision-tree learning in dwell point policies in autonomous vehicle storage and retrieval systems (avsrs). In *2004 International Conference on Machine Learning and Applications, 2004. Proceedings.*, pages 81–84. IEEE, 2004. 9

[18] Miki Fukunari and Charles J Malmborg. A network queuing approach for evaluation of performance measures in autonomous vehicle storage and retrieval systems. *European Journal of Operational Research*, 193(1):152–167, 2009. 10

[19] Jean-Philippe Gagliardi, Jacques Renaud, and Angel Ruiz. Models for automated storage and retrieval systems: a literature review. *International Journal of Production Research*, 50(24):7110–7125, 2012. 4

[20] S Galka, C Scherbarth, and L Troesch. Autostore–was nutzer über das system berichten können. *Ergebnisse einer Online-Umfrage, OTH Regensburg, doi*, 10, 2020. 1

[21] Stefan Galka and Christoph Scherbarth. Simulationsbasierte untersuchung der grenzproduktivität von robotern in einem autostore-lagersystem. *Simulation in Produktion und Logistik 2021: Erlangen, 15.-17. September 2021*, pages 197–206, 2021. 11

[22] Amir Hossein Gharehgozli, Gilbert Laporte, Yugang Yu, and René De Koster. Scheduling twin yard cranes in a container block. *Transportation Science*, 49(3):686–705, 2015. 4

[23] Erico Guizzo. Kiva Systems: three engineers, hundreds od robots, one warehouse. https://spectrum.ieee.org/three-engineers-hundreds-of-robots-one-warehouse, 2008. 8

[24] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. 11

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer vision*, pages 2961–2969, 2017. 12

[26] SJ He and J Luo. Deadlock control of autonomous vehicle storage and retrieval systems via coloured timed petri nets and digraph tools. *International Journal of Production Research*, 47(12):3253–3263, 2009. 40

[27] Sunderesh S Heragu, Xiao Cai, Ananth Krishnamurthy, and Charles J Malmborg. Analytical models for analysis of automated warehouse material handling systems. *International Journal of Production Research*, 49(22):6833–6861, 2011. 10

[28] Hopstack. Evolution of Warehousing Systems: History and Timelines. https://www.hopstack.io/blog/evolution-warehousing-systems-history-timelines, 2022. 4

[29] Dayoung Ko and Jinil Han. A rollout heuristic algorithm for order sequencing in robotic compact storage and retrieval systems. *Expert Systems with Applications*, 203:117396, 2022. 11

[30] Po-Hsun Kuo, Ananth Krishnamurthy, and Charles J Malmborg. Design models for unit load storage and retrieval systems using autonomous vehicle technology and resource conserving storage and dwell point policies. *Applied Mathematical Modelling*, 31(10):2332–2346, 2007. 10

[31] Po-Hsun Kuo, Ananth Krishnamurthy, and Charles J Malmborg. Performance modelling of autonomous vehicle storage and retrieval systems using class-based storage policies. *International Journal of Computer Applications in Technology*, 31(3-4):238–248, 2008. 9

[32] Hung-Yu Lee and Chase C Murray. Robotics in order picking: evaluating warehouse layouts for pick, place, and transport vehicle routing systems. *International Journal of Production Research*, 57(18):5821–5841, 2019. 49

[33] T Lerher, M Borovinsek, M Ficko, and I Palcic. Parametric study of throughput performance in sbs/rs based on simulation. *International Journal of Simulation Modelling*, 16(1):96–107, 2017. 10

[34] Tone Lerher, Milan Edl, and Bojan Rosi. Energy efficiency model for the mini-load automated storage and retrieval systems. *The International Journal of Advanced Manufacturing Technology*, 70:97–115, 2014. 47

[35] Tone Lerher, Banu Y Ekren, Goran Dukic, and Bojan Rosi. Travel time model for shuttle-based storage and retrieval systems. *The International Journal of Advanced Manufacturing Technology*, 78:1705–1725, 2015. 10

[36] Nelly Litvak and Maria Vlasiou. A survey on performance analysis of warehouse carousel systems. *Statistica Neerlandica*, 64(4):401–447, 2010. 5

[37] Charles J Malmborg. Conceptualizing tools for autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, 40(8):1807–1822, 2002. 9, 10

[38] Charles J Malmborg. Interleaving dynamics in autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, 41(5):1057–1069, 2003. 10

[39] Gino Marchet, Marco Melacini, Sara Perotti, and Elena Tappia. Analytical model to estimate performances of autonomous vehicle storage and retrieval systems for product totes. *International Journal of Production Research*, 50(24):7134–7148, 2012. 10

[40] Gino Marchet, Marco Melacini, Sara Perotti, and Elena Tappia. Development of a framework for the design of autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, 51(14):4365–4387, 2013. 9

[41] Russell D Meller and John F Klote. A throughput model for carousel/vlm pods. *IIE Transactions*, 36(8):725–741, 2004. 5

[42] Jeng Nguyen. What is the 80/20 inventory rule? 6 tips to implement it efficiently. https://www.magestore.com/blog/what-is-the-80-20-inventory-rule/, 2022. 36

[43] Lenoble Nicolas, Frein Yannick, and Hammami Ramzi. Order batching in an automated warehouse with several vertical lift modules: Optimization and experiments with real data. *European Journal of Operational Research*, 267(3):958–976, 2018. 5

[44] Ocado. Ocado Group. https://www.ocadogroup.com/technology/powering-osp/, n.d. 1

[45] John Olsson, Daniel Hellström, and Henrik Pålsson. Framework of last mile logistics research: A systematic review of the literature. *Sustainability*, 11(24):7131, 2019. 4

[46] Jennifer A Pazour and Russell D Meller. The impact of batch retrievals on throughput performance of a carousel system serviced by a storage and retrieval machine. *International Journal of Production Economics*, 142(2):332–342, 2013. 5

[47] Kees Jan Roodbergen and Iris FA Vis. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194(2):343–362, 2009. 4

[48] Debjit Roy, Ananth Krishnamurthy, Sunderesh Heragu, and Charles Malmborg. Queuing models to analyze dwell-point and cross-aisle location in autonomous vehicle-based warehouse systems. *European Journal of Operational Research*, 242(1):72–87, 2015. 9

[49] Debjit Roy, Ananth Krishnamurthy, Sunderesh S Heragu, and Charles J Malmborg. Performance analysis and design trade-offs in warehouses with autonomous vehicle technology. *Iie Transactions*, 44(12):1045–1060, 2012. 9

[50] Debjit Roy, Ananth Krishnamurthy, Sunderesh S Heragu, and Charles J Malmborg. Blocking effects in warehouse systems with autonomous vehicles. *IEEE Transactions on Automation Science and Engineering*, 11(2):439–451, 2013. 7, 40

[51] Debjit Roy, Shobhit Nigam, René de Koster, Ivo Adan, and Jacques Resing. Robot-storage zone assignment strategies in mobile fulfillment systems. *Transportation Research Part E: Logistics and Transportation Review*, 122:119–142, 2019. 8

[52] Elena Tappia, Debjit Roy, René De Koster, and Marco Melacini. Modeling, analysis, and design insights for shuttle-based compact storage systems. *Transportation Science*, 51(1):269–295, 2017. 9

[53] S Tjeerdsma. Redesign of the autostore order processing line. Master's thesis, University of Twente, 2019. 11

[54] Philipp Trost, Georg Kartnig, and Michael Eder. Simulation des grenzdurchsatzes von autostore-systemen. *Logistics Journal: Proceedings*, 2022(18), 2022. 11

[55] Cristiano Arbex Valle and John E Beasley. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. *Computers & Operations Research*, 125:105090, 2021. 8

[56] Teun Van Gils, Katrien Ramaekers, An Caris, and René BM De Koster. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, 267(1):1–15, 2018. 4

[57] Kun Wang, Yiming Yang, and Ruixue Li. Travel time models for the rack-moving mobile robot system. *International Journal of Production Research*, 58(14):4367–4385, 2020. 8

[58] Felix Weidinger and Nils Boysen. Scattered storage: How to distribute stock keeping units all around a mixed-shelves warehouse. *Transportation Science*, 52(6):1412–1427, 2018. 49

[59] Felix Weidinger, Nils Boysen, and Dirk Briskorn. Storage assignment with rack-moving mobile robots in kiva warehouses. *Transportation Science*, 52(6):1479–1495, 2018. 8

[60] Yu Xiang, Christopher Xie, Arsalan Mousavian, and Dieter Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning*, pages 461–470. PMLR, 2021. 12

[61] Lin Xie, Nils Thieme, Ruslan Krenzler, and Hanyi Li. Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. *European Journal of Operational Research*, 288(1):80–97, 2021. 49

[62] Biqi Yang, Xiaojie Gao, Kai Chen, Rui Cao, Yidan Feng, Xianzhi Li, Qi Dou, Chi-Wing Fu, Yun-Hui Liu, and Pheng-Ann Heng. Sesr: Self-ensembling sim-to-real instance segmentation for auto-store bin picking. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5987–5994. IEEE, 2022. 12

[63] Nima Zaerpour, Yugang Yu, and René de Koster. Small is beautiful: A framework for evaluating and optimizing live-cube compact storage systems. *Transportation Science*, 51(1):34–51, 2017. 7

[64] Li Zhang, Ananth Krishnamurthy, Charles J Malmborg, and Sunderesh S Heragu. Variance-based approximations of transaction waiting times in autonomous vehicle storage and retrieval systems. *European Journal of Industrial Engineering*, 3(2):146–169, 2009. 10

[65] Bipan Zou, Yeming Gong, Xianhao Xu, and Zhe Yuan. Assignment rules in robotic mobile fulfilment systems for online retailers. *International Journal of Production Research*, 55(20):6175–6192, 2017. 8

[66] Bipan Zou, René De Koster, and Xianhao Xu. Operating policies in robotic compact storage and retrieval systems. *Transportation Science*, 52(4):788–811, 2018. 2, 3, 10, 16, 20, 38, 40