How to Use Math-heuristic Deploy Tote-Based Storage and Retrieval Systems

Jie Shao[a,b],Yuexin Kang[a,b],Peng Yang[a,b]*

[a]Division of Logistics and Transportation, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China.

[b]Institution of Data and Information, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China.

*Corresponding author

ABSTRACT

This paper highlights the urgent need for a unified and standardized research methodology to optimize order fulfillment processes across various robotic warehousing systems. Despite the rapid advancements in e-commerce and logistics, existing literature lacks a cohesive approach to the joint optimization of diverse warehousing systems, resulting in inefficiencies and higher costs. To address this gap, we propose a comprehensive, integrated decision-making framework that targets three core elements: orders, retrieval units, and handling units. This framework optimizes the assignment, sequencing, and scheduling of these elements, making it applicable to different automated and robotic warehousing systems. We apply this framework to the HaiRobotics multi-tote system, developing both a three-stage model and an integrated model that demonstrate joint optimization decisions in order fulfillment. To tackle the complex joint optimization challenges inherent in these systems, we introduce two innovative methods: a mathematical programming-based model decomposition method, which provides exact solutions, and a neural-enhanced revolving math-heuristic algorithm, which efficiently solves large-scale models within a short time frame. Numerical experiments confirm the effectiveness and superiority of the proposed integrated decision-making framework and methods..

Keywords:

Robotic Warehousing Systems, Order Fulfillment Optimization, Integrated Decision-Making Framework, Model Decomposition, Neural-enhanced Reinforcement Learning

1. Introduction

The rapid development of e-commerce and logistics has driven warehouse systems to continuously evolve. Advances in automation and intelligent technologies have given rise to a wide range of automated and robotic warehousing systems (Boysen and De Koster, 2024). These systems, diverse in design and unique in their characteristics, present ongoing optimization challenges and research opportunities. Studies in this field have analyzed different systems from various perspectives, utilizing a broad spectrum of models and methodologies. Establishing a unified, standardized research methodology for studying these systems would greatly enhance the efficiency of industry research on order fulfillment processes across various types of warehouses. This would not only help warehouse users select the most suitable solutions for their needs but also enable warehouse providers to leverage their strengths and improve efficiency. Additionally, such standardization would contribute to unifying the academic research framework on warehouse operations, thereby fostering the development of a comprehensive and robust theoretical foundation for warehouse studies.

Despite significant advancements, a unified and standardized research methodology for optimizing various robotic warehouse systems has yet to be established. This challenge primarily stems from the diversity of warehouse systems and the inherent complexities of their order fulfillment processes. The absence of such a methodology hampers the development of solutions that can adapt to the rapidly evolving warehousing landscape. Consequently, each newly developed system necessitates a comprehensive and unique analysis of its order fulfillment process, as no universal framework currently exists for reference. This lack of standardization not only increases costs for industry stakeholders but also delays the rapid deployment of new warehouse technologies. In response to the pressing need for a unified research methodology, this paper proposes a general integrated framework to address the joint optimization challenges in robotic warehouse systems.

To establish a unified decision-making framework, we comprehensively consider various popular warehousing systems, including mini-load systems, shuttle-based systems, Kiva systems, and multi-tote systems. We examine their characteristics and analyze their core principles. Despite the diversity of these goods-to-person systems and the differences in their periods of emergence, their fundamental essence remains largely consistent. Specifically, the order fulfillment process follows these steps: upon arrival, an order is assigned to a picking station and sequenced; it is then matched with totes or pods,

which are sequenced for retrieval; robots are assigned to retrieval tasks and scheduled to execute them in sequence until the order is fully picked. Once completed, the order leaves the picking station, and the cycle begins anew with the next order. The key optimization decisions in these systems revolve around three core elements: orders, retrieval units (totes or pods), and handling units (robots or equipment). The decision-making process forms a loop: order → retrieval unit → handling unit → order → and so on. While subtle differences exist between systems—such as the retrieval unit being a tote in the multi-tote system versus a pod in the robotic mobile fulfillment system (RMFS), or the use of a single type of handling equipment in the multi-tote and RMFS systems versus two types in the shuttle-based systems—these variations do not alter the fundamental decision-making framework, which is composed of the three critical elements: orders, retrieval units, and handling units.

Building on the above analysis, we focus on three key elements—orders, retrieval units, and handling units—and categorize the optimization decisions in robotic warehousing systems into three critical components: the assignment and sequencing of orders, the assignment and sequencing of retrieval units, and the scheduling and path planning of handling units. We propose an integrated joint optimization framework that simultaneously optimizes decisions related to these components, addressing the operational challenges of robotic warehousing systems.

Our research addresses the following issues: (1) We propose an integrated decision-making framework for automated and robotic warehousing systems, focusing on the three key elements of orders, retrieval units, and handling units. This framework optimizes decisions related to the assignment and sequencing of orders, the assignment and sequencing of retrieval units, and the scheduling and path planning of handling units, making it applicable to all automated and robotic warehousing systems. (2) Based on this framework, we develop both a three-stage decision-making model and an integrated decision-making model, using the HaiRobotics multi-tote system as a case study. These models illustrate the joint optimization of the three key elements in the order fulfillment process of the HaiRobotics system. (3) For the complex and large-scale mixed-integer programming (MIP) integrated decision-making model, we design a model decomposition method (MP-MD) capable of providing exact solutions to the joint optimization problem. (4) To solve large-scale problems in real warehouse operations, we introduce a neural-enhanced rotating mathematical heuristic algorithm (DeepRMHA), which efficiently solves large-scale integrated models within a short time frame.

The contributions of this study are as follows: (1) To the best of our knowledge, this is the first

research to propose a unified and standardized integrated decision-making framework for optimizing order fulfillment in robotic warehousing systems. We focus on three key elements—orders, retrieval units, and handling units—and jointly optimize all decisions related to these components, making the framework applicable to all automated and robotic warehousing systems. (2) To solve the complex integrated decision-making problem, we pioneered a mathematical programming method based on model decomposition (MP-MD), which provides exact solutions to the joint optimization problem. (3) We innovatively applied reinforcement learning techniques to combinatorial optimization problems and proposed a neural-enhanced rotating mathematical heuristic algorithm (DeepRMHA) that efficiently solves large-scale integrated models within a short time frame. (4) We conducted multiple sets of numerical experiments across various scales, all of which demonstrated the effectiveness and superiority of our integrated optimization framework.

The remainder of the paper is structured as follows. Section 2 reviews the related literature. Section 3 introduces the integrated decision-making framework for the joint optimization of order fulfillment processes in robotic warehouses, focusing on the three key elements: orders, retrieval units, and handling units. A multi-tote system is used as an example to develop both staged and integrated MIP models. Section 4 details the MP-MD method for solving the integrated decision-making framework, while Section 5 presents the DeepRMHA method for efficiently solving large-scale problems in a short time. Section 6 summarizes the computational results, and Section 7 provides the conclusion and suggestions for future research.

2. Literature Review

We categorize the existing literature related to our topic into three themes: robotic warehouse system, joint optimization decision making, and learning methodology for combination optimization.

2.1 Robotic Warehouse System

We primarily review the existing research on autonomous vehicle storage and retrieval system (AVS/RS), puzzle-based storage (PBS) system and robotic mobile fulfillment systems (RMFS). For further information on robotic warehousing systems, interested readers can refer to the articles by Azadeh et al. (2019), Boysen et al. (2019), and Boysen and De Koster (2024).

AVS/RS is a storage and retrieval system that employs shuttles to manage both incoming and outgoing tasks, including those within Shuttle-Based Storage and Retrieval Systems (SBS/RS). The concept of AVS/RS was first introduced by Hausman et al. (1976), who developed an analytical model to assess the system's performance metrics. Since then, queuing theory has been widely applied to evaluate the performance of AVS/RS (Bozer & White, 1984). Researchers such as Lerher (2016), Schenone et al. (2020), and Singbal & Adil (2021) have extensively studied queuing network models and time travel models to evaluate the system's performance. Research has increasingly focused on optimizing the system's operational efficiency and effectiveness, with some employing reinforcement learning methods. Notably, Ekren & Arslan (2022) presented an innovative SBS/RS design allowing shuttles to move between system tiers. They utilized the Q-learning algorithm for transaction selection, demonstrating its superiority over static scheduling algorithms. Later, they proposed a Deep Q-Learning approach, combined with first-come-first-serve and shortest process time methods, for broader application in tier-to-tier SBS/RS (Arslan & Ekren, 2023). In contrast, other researchers have focused on operational optimization frameworks. Yang et al. (2023) introduced a two-stage heuristic to optimize shuttle scheduling and pickup requests in shuttle-based deep lane storage systems, aiming to minimize makespan. Meanwhile, Chen et al. (2023) proposed a decomposition-based heuristic for SBS/RS with two independent lifts on a single mast, targeting sequencing and retrieval assignment to minimize makespan.

PBS system is an innovative type of automated compact storage system that features high-density shelving on a square grid and one or more empty locations, known as "buffers," used for temporary parking. Gue and Kim (2007) are the first to evaluate the expected item retrieval time for a single fixed-position escort. Since then, researchers extend their work to include multiple randomly located escorts, as well as multiple escorts considering simultaneous movement and congestion. For the single-item retrieval problem, Rohit et al. (2010) propose a general integer programming model, while other authors such as Yalcin et al. (2019), Ma et al. (2022), Yu et al. (2022), and Bukchin and Raviv (2022) propose various heuristic or exact algorithms to solve the problem. For the multi-item retrieval problem, Gue et al. (2013) discuss multi-item retrieval based on GridStore technology, while Mirzaei et al. (2017) propose a closed-form expression for retrieving two items. Zou and Qi (2021) propose a heuristic algorithm to handle the multi-item retrieval problem with multiple random location escorts and I/O points, and He et al. (2023) design a DRL method to solve the multi-item retrieval problem.

RMFS is a popular parts-to-picker system, with research primarily focusing on system analysis and performance evaluation, robot scheduling and route planning, and task assignment and scheduling. Queuing network models are applied to analyze system performance, with Roy (2016) being the first to use this approach for RMFS. Other studies analyze RMFS performance concerning factors such as partitioned versus unpartitioned cases (Lamballais et al., 2017), dedicated versus shared robots (Yuan & Gong, 2017), battery management (Zou et al., 2018), picking and replenishment processes (Roy et al., 2019), and customer sorting (Gong et al., 2021). Operational optimization and control in RMFS include shelf scheduling, robot scheduling, and path planning. Weidinger et al. (2018) and Ji et al. (2020) explore storage space allocation, while Boysen et al. (2017) propose various strategies for shelf sorting and scheduling. Path planning and traffic control for robots are also studied, with Merschformann et al. (2019) and Müller et al. (2020) proposing different path planning algorithms and strategies for managing collisions, deadlocks, and congestion.

2.2 Joint Optimisation Decision Making

We review joint optimization research in automated warehousing systems, using RMFS as a key example.

In RMFS order fulfillment, many studies focus on minimizing pod visits. These studies explore various aspects of warehouse order fulfillment, particularly the joint optimization of orders and storage units like pods, racks, or shelves, often employing heuristic methods. Boysen et al. (2017) address order assignment, sequencing for picking stations, and shelf sequencing, enabling over half of the robots to be scheduled compared to previous methods. Similarly, Xiang et al. (2018) introduce a heuristic to optimize pod and order assignments, reducing pod visits. Yang et al. (2021) develop an MIP model for joint order sequencing and rack scheduling, proposing a two-stage heuristic that significantly reduces robot tasks compared to separate optimization. Valle & Beasley (2021) extend this work to multi-workstation scenarios, designing two metaheuristics for order assignment, rack assignment, and rack sequencing. A notable feature of Xie et al. (2021) is the introduction of split orders among workstations, a first in RMFS, enhancing flexibility and efficiency. Zhuang et al. (2022) further investigate joint order assignment, sequencing, and rack scheduling across multiple workstations, addressing workload balancing and rack conflicts. Additionally, other studies focus on the joint optimization of picking and replenishment decisions in robotic forward-reserve warehouses, proposing a synchronization

mechanism to streamline operations (Jiang et al., 2020).

Nevertheless, pod visits serve only as a proxy for order fulfillment efficiency, and a gap remains between pod visits and actual RMFS performance. To address this, recent studies increasingly focus on time-based metrics, such as total order fulfillment time and makespan. Li et al. (2017) formulate a 0-1 linear programming model to minimize the total time required for order assignment and rack scheduling, focusing on single-picking station scenarios. Wang et al. (2022) expand this model to multiple picking stations, highlighting the significant impact of fluctuating worker productivity on warehouse operations—a departure from traditional models that treat productivity as static. Additionally, Wang et al. (2023) introduce a two-stage hybrid heuristic algorithm to optimize various aspects of warehouse operations, including order assignment, sequencing, rack selection, and sequencing, by incorporating inter-picking station operations. Zhen et al. (2023) propose an integrated model that synchronizes the scheduling of orders, robots, and storage pods in RMFS, minimizing order fulfillment time by efficiently coordinating all components and addressing their complex interactions.

Extensive research has addressed joint optimization problems in RMFS and other robotic warehousing systems. However, no unified, standardized decision-making framework exists. This paper aims to develop such a framework for joint optimization in robotic warehousing systems.

2.3 Learning Methodology for Combination Optimization

Combinatorial optimization problems are discrete optimization challenges, involving the search for optimal elements within a set of discrete items. Examples include the knapsack problem, traveling salesman problem, and cutting stock problem. Traditional solutions encompass exact, approximation, and heuristic algorithms. Recently, there is a rapid shift toward integrating machine learning methods into combinatorial optimization.

Machine learning combined with combinatorial optimization primarily follows two approaches: end-to-end methods and leveraging machine learning to enhance traditional optimization methods. Song et al. (2023) address the flexible job shop scheduling problem by introducing an end-to-end Deep Reinforcement Learning (DRL) method to learn high-quality priority dispatching rules (PDRs). This approach integrates process selection and machine allocation into a single decision. Experimental results show that the method outperforms traditional PDRs and offers high computational efficiency. The main advantage of end-to-end methods is their ability to directly output solutions without requiring a search,

leading to fast solution times. These models also exhibit robust generalization, solving instances with similar distribution characteristics once trained. However, challenges include ensuring solution quality and a decline in performance on large-scale problems.

Comparatively, incorporating machine learning into traditional optimization methods often yields better results. Research in this area falls into two main categories: combining machine learning with exact algorithms and integrating it with heuristic methods. For example, Morabit et al. (2021) introduce a novel approach that uses machine learning to expedite column generation. In each iteration of the column generation algorithm, a model is trained to select the most appropriate column, reducing the computational time of the master problem. This method has proven effective in solving problems such as vehicle and crew scheduling and vehicle routing with time windows.

In cases where machine learning is combined with heuristic methods, Wu et al. (2020) enhance search efficiency in routing problems by using reinforcement learning to learn pairing operators. Kim et al. (2021) approximate near-optimal solutions for various NP-hard routing problems with two Deep Reinforcement Learning (DRL) iterative strategies. Wang et al. (2021) introduce a dual-layer framework that employs graph neural networks to generate an initial graph structure, which is then optimized using traditional heuristic methods. Experimental results show that this approach outperforms handcrafted heuristics. Integrating machine learning with heuristics is a promising and exploratory research direction.

With the rise of e-commerce warehousing, machine learning methods are increasingly applied to warehousing (Fuentes Saenz et al., 2011; Matusiak et al., 2017). Cals et al. (2021) used Deep Reinforcement Learning (DRL) with the Proximal Policy Optimization (PPO) algorithm to address the online order batching problem (OOBP). Pirayesh Neghab et al. (2022) employed a multi-layered neural network for dynamic demand estimation and cost reduction in inventory management. Van Der Gaast et al. (2022) applied deep neural networks (DNNs) to predict optimal order picking systems, enabling fast, objective decisions without exhaustive simulations. Wang et al. (2024) optimized reshuffling in robotic storage systems with reinforcement learning, significantly improving efficiency. Kang et al. (2024) examined the online order batching problem with a reservation mechanism (OOBPRM), using ensemble learning to predict order similarities, improving turnover time and efficiency in e-commerce warehouses.

Machine learning methods have been applied to many combinatorial optimization problems, offering significant advantages over traditional approaches. However, their use in the warehousing domain remains limited. This paper aims to integrate reinforcement learning into a joint optimization

framework for warehousing and propose a reinforcement learning-driven solution for large-scale joint optimization problems in this field.

3. Problem Description and Formulation

In this section, we propose a unified and standardized integrated decision-making model framework for various types of robotic warehousing systems. This framework focuses on three key elements: orders, retrieval units, and handling units, and jointly optimizes decisions related to the assignment and sequencing of orders, the assignment and sequencing of retrieval units, and the scheduling and path planning of handling units. The framework is applicable to all robotic warehousing systems. To concretely illustrate our integrated decision-making model framework, we use the HaiRobotics multi-tote system as an example, focusing on the joint optimization of order fulfillment. We establish both a three-stage decision-making model and an integrated decision-making model. The models and framework presented in this section can be extended to other types of robotic warehousing systems.

3.1 MTSR System with Sorting Conveyor

The MTSR system is an automated storage system employing ACRs for tote storage and retrieval. The system consists of two primary components: the upstream tote storage and retrieval system, and the downstream sorting conveyor system, as shown in Figure 3. Multiple ACRs are distributed throughout the storage and retrieval system, each responsible for transporting totes. With a maximum carrying capacity of 8 totes per ACR, the system can efficiently handle a high volume of totes simultaneously. Additionally, the conveyor system features multiple picking stations, each capable of temporarily storing a predetermined number of totes and orders, albeit with an upper limit. Figure 3 presents a comprehensive layout of the MTSR system integrated with a sorting conveyor system.

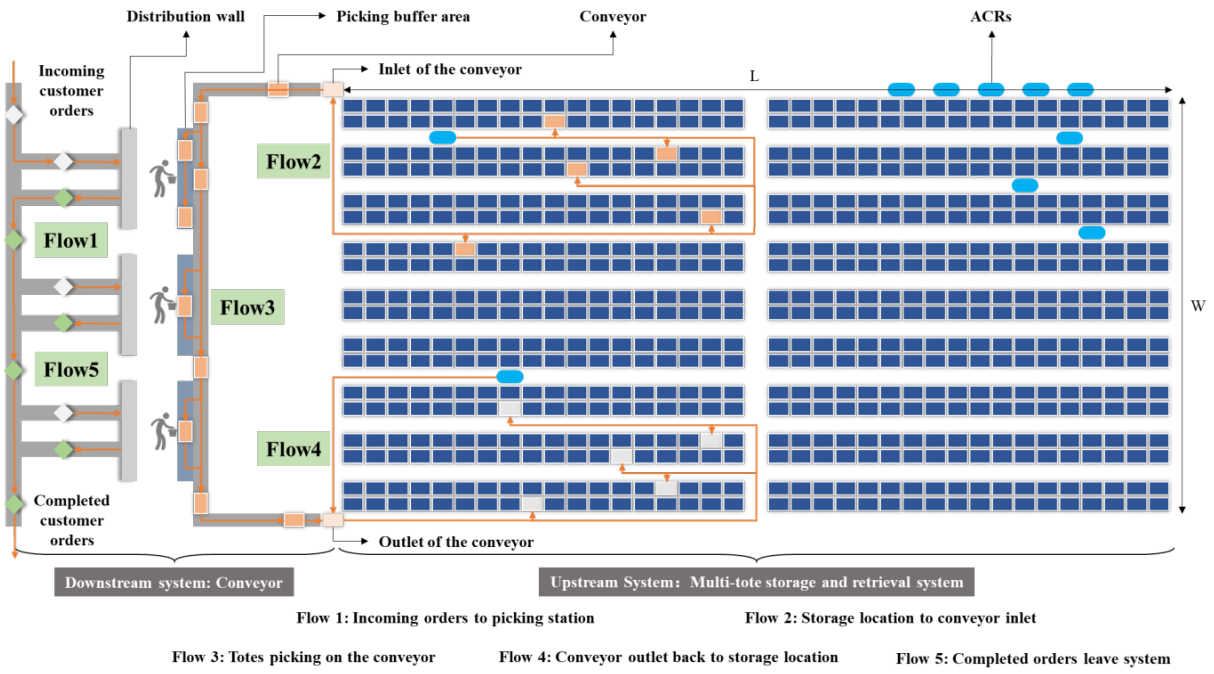

Figure 3. MTSR system with sorting conveyor and tote flow process.

The flow of totes within the MTSR system, integrated with a sorting conveyor system, can be delineated into five distinct stages, as depicted in Figure 3. During the initial stage (Flow 1), orders enter the system and are allocated to their respective picking stations. Subsequently, in the second stage (Flow 2), the ACRs transport the totes to the conveyor inlet for discharge. Upon reaching the third stage (Flow 3), the totes arrive at the conveyor inlet and are directed towards the designated picking stations for retrieval. Following this, they are conveyed to the conveyor outlet. During the fourth stage (Flow 4), the ACRs retrieve the completed totes from the conveyor outlet and return them to their original storage locations. Finally, in the fifth stage (Flow 5), orders that have completed the picking process exit the system.

The system will receive a series of order clusters already matched with totes. Matching these orders with totes will generate order cluster-tote (OT) pairs. An OT pair consists of an order cluster and a set of totes. The quantity of a SKU for an order cluster (an order) may require multiple totes to fulfill, eliminating the assumption of sufficient stock of an SKU in a single tote for an order. We focus on three primary decisions: Decision 1 (corresponding to Flow 1 and Flow 5): order assignment and sequencing; Decision 2 (corresponding to Flow 3): tote assignment and sequencing; and Decision 3 (corresponding to Flows 2 and 4): ACR task assignment and scheduling.

Our objective is to optimize the flow of totes through the MTSR system by minimizing the time

required to complete all order fulfillment processes. To model this process, we propose a comprehensive integrated model based on several key assumptions: 1) No collisions, congestion, or deadlock occur between multiple ACR paths. 2) The time required to access totes depends on the number of shelf levels they occupy. 3) ACRs move at a uniform speed, ignoring acceleration and deceleration. 4) A consistent time is required for ACRs to pick up and place totes at the entrance or exit of the ring picker. 5) The time taken for the same batch of totes to arrive and depart from the conveyor is identical. 6) Totes do not remain on the picker. 7) Totes move at a uniform speed without collision on the picker. 8) The picking time of a tote at the picking station is proportional to the number of SKUs to be picked from that tote. 9) Adequate capacity is available for temporary storage of totes at the exit of the ring picker.

3.2 Three-Stage Separate Decision Method

3.2.1 Stage 1 Model：Assigning Orders to Picking Stations and Sequence

In the initial stage, orders are assigned to specific picking stations based on the order picking time and the capacity constraints of the stations. The model determines the appropriate picking station for each order and establishes the start and end times for the picking process. The variables are defined as follows:

Input parameters：

t_o^{order}: The picking time of order o at the picking station.

Sets：

O: The set of orders.

P: The set of picking stations.

Parameters:

b_1: The maximum number of orders that can be buffered at a picking station.

M: A sufficiently large positive number

Variables:

$z_{o,p}$: Indicates whether order o is assigned to picking station p. If so, $z_{o,p} = 1$; otherwise, $z_{o,p} = 0$.

α_{o_1,o_2}: Indicates whether order o_2 has already started execution when order o_1 begins. If so, $\alpha_{o_1,o_2} = 1$; otherwise, $\alpha_{o_1,o_2} = 0$.

β_{o_1,o_2}: Indicates whether order o_2 has not yet completed execution when order o_1 begins. If so, $\beta_{o_1,o_2} = 1$; otherwise, $\beta_{o_1,o_2} = 0$

γ_{o_1,o_2}: Indicates whether order o_2 is in the process of execution when order o_1 begins. If so, $\gamma_{o_1,o_2} = 1$; otherwise, $\gamma_{o_1,o_2} = 0$

δ_{o_1,o_2}: Indicates whether order o_2 is being executed at the same picking station as order o_1 when order o_1 begins. If so, $\delta_{o_1,o_2} = 1$; otherwise, $\delta_{o_1,o_2} = 0$

T_o^s: The start time of order o

T_o^e: The end time of order o

T_1: The maximum completion time among all orders.

Mathematical model:

$$min \quad T_1 \tag{3-1}$$

$$T_1 \geq T_o^e, \forall o \in O \tag{3-2}$$

$$T_o^s + t_o^{order} \leq T_o^e, \forall o \in O \tag{3-3}$$

$$\sum_{p=1}^{P} z_{o,p} = 1, \forall o \in O \tag{3-4}$$

$$T_{o_1}^s - T_{o_2}^s \leq \alpha_{o_1,o_2} * M, \forall o_1, o_2 \in O \tag{3-5}$$

$$T_{o_2}^e - T_{o_1}^s \leq \beta_{o_1,o_2} * M, \forall o_1, o_2 \in O \tag{3-6}$$

$$\gamma_{o_1,o_2} \geq \alpha_{o_1,o_2} + \beta_{o_1,o_2} - 1, \forall o_1, o_2 \in O \tag{3-7}$$

$$\delta_{o_1,o_2} \geq \gamma_{o_1,o_2} + z_{o_1,p} + z_{o_2,p} - 2, \forall o_1, o_2 \in O, \forall p \in P \tag{3-8}$$

$$\sum_{o_2}^{O} \delta_{o_1,o_2} \leq b_1, \forall o_1 \in O \tag{3-9}$$

$$z_{o,p} \in \{0,1\}, \forall o \in O, p \in P \tag{3-10}$$

$$\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2} \in \{0,1\}, \forall o_1, o_2 \in O \tag{3-11}$$

$$T_o^s, T_o^e \geq 0, \forall o \in O \tag{3-12}$$

$$T_1 \geq 0 \tag{3-13}$$

Objective function (3-1) minimizes the maximum completion time across all orders. Constraints (3-2) set the maximum completion time, while constraints (3-3) define the relationship between the start and end times of order processing. Constraints (3-4) mandate the completion of all orders. Constraints (3-5) through (3-8) impose restrictions on the decision variables $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$. Constraints (3-9) limit the number of orders that can be processed simultaneously at a picking station. Finally, constraints (3-10) through (3-13) define the specified variables.

3.2.2 Stage 2 Model：Assigning Totes to Picking Stations and Sequence

In the second stage, totes are assigned to specific picking stations based on the picking time and the capacity constraints of the stations. The model determines the appropriate station for each tote, along with the arrival, start, and end times for tote picking at the designated station. The variables are defined as follows:

Input parameters：

t_i^{toteIn}: The time when tote *i* arrives at the conveyor entrance.

$z_{o,p}$: Indicates whether order o is assigned to picking station p. If so, $z_{o,p} = 1$; otherwise, $z_{o,p} = 0$.

Sets：

O: The set of orders.

P: The set of picking stations.

P_1: The set of first pickup points for all totes, which is also the set of all totes.

Parameters：

p_i: The picking time required for tote i, related to the number of SKUs to be picked from the tote.

$OT_{i,o}$: order cluster-tote (OT) pairs; if tote i belongs to order o, then $OT_{i,o} = 1$; otherwise, $OT_{i,o} = 0$

t_p^1: The time required to travel from the conveyor entrance to picking station p.

t_p^2: The time required to travel from picking station p to the conveyor exit.

b_2: The maximum number of totes that can be buffered at a picking station.

Variables：

$y_{i,p}$:Indicates whether tote i is assigned to picking station p. If so, $y_{i,p} = 1$; otherwise, $y_{i,p} = 0$

$f_{i,j,p}$:. Indicates whether tote i arrives at picking station p before tote j. If so, $f_{i,j,p}^2 = 1$; otherwise, $f_{i,j,p}^2 = 0$

I_i:The time when tote i arrives at the conveyor entrance.

$T_{i,p}^a$: The time when tote i arrives at picking station p.

$T_{i,p}^s$: The time when tote i starts being picked at picking station p.

$T_{i,p}^e$: The time when tote i finishes being picked at picking station p

T_2: The maximum completion time among all totes.

Mathematical model：

$$min \quad T_2 \tag{3-14}$$

$$T_2 \geq T_{i,P}^e + t_P^2, \forall i \in P_1 \tag{3-15}$$

$$I_i \geq t_i^{toteIn}, \forall i \in P_1 \tag{3-16}$$

$$\sum_{i=1}^{P_1} \sum_{p=1}^{P} y_{i,p} = \sum_{i=1}^{P_1} \sum_{o=1}^{O} OT_{i,o}, \forall o \in O \tag{3-17}$$

$$OT_{i,o} * z_{o,p} \leq y_{i,p}, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{3-18}$$

$$T_{i,1}^a = I_i + t_1^1, \forall i \in P_1 \tag{3-19}$$

$$T_{i,p}^s \geq T_{i,p}^a, \forall i \in P_1, \forall p \in P \tag{3-20}$$

$$T_{i,p}^e \geq T_{i,p}^s, \forall i \in P_1, \forall p \in P \tag{3-21}$$

$$T_{i,p}^a = T_{i,p-1}^e + t_p^1 - t_{p-1}^1, \forall i \in P_1, \forall p \in 2, \dots, P \tag{3-22}$$

$$T_{i,p}^e - T_{i,p}^s = p_i * y_{i,p}, \forall i \in P_1, \forall p \in P \tag{3-23}$$

$$T_{i,p}^a - T_{j,p}^a \leq \left(1 - f_{i,j,p}\right) * M, \forall i, j \in P_1, \forall p \in P \tag{3-24}$$

$$T_{i,p}^a - T_{j,p}^a \geq -f_{i,j,p} * M, \forall i, j \in P_1, \forall p \in P \tag{3-25}$$

$$T_{j,p}^s - T_{i,p}^e \geq \left(f_{i,j,p} + y_{i,p} + y_{j,p} - 3\right) * M, \forall i, j \in P_1, \forall p \in P \tag{3-26}$$

$$T_{i,p}^s - T_{i,p}^a \leq (b_2 - 1) * p_i, \forall i \in P_1, \forall p \in P \tag{3-27}$$

$$y_{i,p} \in \{0,1\}, \forall i \in P_1, \forall p \in P \tag{3-28}$$

$$f_{i,j,p} \in \{0,1\}, \forall i, j \in P_1, \forall p \in P \tag{3-29}$$

$$I_i \geq 0, \forall i \in P_1 \tag{3-30}$$

$$T_{i,p}^a, T_{i,p}^s, T_{i,p}^e \geq 0, \forall i \in P_1, \forall p \in P \tag{3-31}$$

$$T_2 \geq 0 \tag{3-32}$$

Objective function (3-14) minimizes the maximum time required for all totes to reach the conveyor exit. Constraints (3-15) mandate that the time a tote reaches the conveyor exit must be no less than the sum of the time it completes picking at the last station and the travel time to the conveyor exit. Constraint (3-16) defines the time a tote reaches the conveyor entrance. Constraints (3-17) ensure that all totes are picked. Constraints (3-18) stipulate that a tote i can only be assigned to picking station p if it belongs to order o, and order o has been assigned to station p. Constraints (3-19) define the time a tote arrives at the first picking station. Constraints (3-20) and (3-21) require that the start time for picking a tote at a station must be after its arrival time, and the end time must be no earlier than its arrival time. Constraint (3-22) states that a tote's arrival time at the next picking station equals its finish time at the previous

station plus the travel time between the two stations. Constraint (3-23) specifies that if a tote is assigned to a station, its end picking time equals its start picking time plus the picking duration; otherwise, the end picking time equals the start time. Constraints (3-24) and (3-25) determine that if tote i arrives at station p before tote j, then $f_{i,j,p}^{2} = 1$; otherwise, $f_{i,j,p}^{2} = 0$. Constraint (3-26) requires that if tote i arrives at station p before tote j, and both are assigned to station p, then tote j must start picking no earlier than the time tote i finishes. Constraint (3-27) limits the number of totes buffered at each picking station to not exceed the upper limit. Finally, constraints (3-28) through (3-32) define the specified variables.

3.2.3 Stage 3 Model：Assigning Totes to Robots and Robot Scheduling

In the third stage, the tasks of handling totes are assigned to AMRs based on travel time, with AMRs capable of carrying multiple totes simultaneously. The routing of AMRs is also planned, determining which AMR handles each tote and the sequence in which it visits the tote locations. The variables are defined as follows:

Input parameters：

$t_i^{toteOut}$: The time when tote i arrives at the conveyor exit.

Sets：

K: The set of AMRs.

P_1: The set of first pickup points for all totes, which is also the set of all totes.

P_2: The set of second pickup points for all totes, which is also the conveyor exit.

D_1: The set of first delivery points for all totes, which is also the conveyor entrance.

D_2: The set of second delivery points for all totes, which is also the storage locations for all totes.

W: The set of second delivery points for all totes, which is also the storage locations for all totes.

N: The set of all points $P_1 \cup P_2 \cup D_1 \cup D_2 \cup W$, , including the two pickup points, two delivery points, and the starting points of the AMRs.

Parameters：

q_i: The load at point i, with a value of 1, as each tote is considered a unit load.

Q: The maximum load capacity of the AMRs, with a value of 8, since each AMR can carry up to 8 totes.

n: The total number of totes to be picked.

t_{ij}: The travel time required by an AMR to move from tote point i to tote point j.

s_i: The service time required at tote point i, which is proportional to the shelf level where the tote is located.

e_i: The earliest retrieval time of tote i.

l_i: The latest retrieval time of tote i.

Variables：

x_{ij}: Indicates whether an AMR travels from tote point i to tote point j. If so, $x_{ij} = 1$; otherwise, $x_{ij} = 0$

u_i^k: Indicates whether AMR k visits tote point i.

Q_i: The load carried by an AMR at tote point i.

T_i: The time at which the task at tote point i.

T_{D_1}: The maximum task completion time among all D_1.

T_{D_2}:The maximum task completion time among all D_2.

T_3: The maximum task completion time among all AMRs.

Mathematical model：

$$min \ \ T_3 \tag{3-33}$$

$$T_3 \geq T_{D_1} \tag{3-34}$$

$$T_3 \geq T_{D_2} \tag{3-35}$$

$$T_{D_1} \geq T_i, \forall i \in D_1 \tag{3-36}$$

$$T_{D_2} \geq T_i, \forall i \in D_2 \tag{3-37}$$

$$t_i^{toteOut} \leq T_{i+n}, \forall i \in P_1 \tag{3-38}$$

$$\sum_{j=1,j\neq i}^{N} x_{ij} = \sum_{j=1,j\neq i}^{N} x_{ji}, \forall i \in N \tag{3-39}$$

$$\sum_{k=1}^{K} \sum_{j=1,j\neq i}^{N} x_{ij} \geq 1, \forall i \in P_1 \cup P_2 \cup D_1 \cup D_2 \tag{3-40}$$

$$u_i^k - u_j^k \geq M(x_{ij} - 1), \forall i \in N, \forall j \in N, \forall k \in K \tag{3-41}$$

$$u_i^k - u_j^k \leq M\left(1 - x_{ij}\right), \forall i \in N, \forall j \in N, \forall k \in K \tag{3-42}$$

$$u_i^k = 1, \forall i \in W, i = W_k, \forall k \in K \tag{3-43}$$

$$\sum_{k=1}^{K} u_i^k = 1, \forall i \in N \tag{3-44}$$

$$u_i^k = u_{i+2n}^k, \forall i \in P_1 \cup P_2, \forall k \in K \tag{3-45}$$

$$\sum_{j=1,j\neq i}^{N} x_{ij} \leq 1, \forall i \in W, i = W_k \tag{3-46}$$

$$Q_j \geq Q_i + q_i - Q(1 - x_{ij}), \forall i \in N \;\; i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \quad (3-47)$$

$$0 \leq Q_i \leq Q, \forall i \in N \quad (3-48)$$

$$T_j \geq T_i + t_{ij} + s_i - M(1 - x_{ij}), \forall i \in N \;\; i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \quad (3-49)$$

$$T_{i+2n} \geq T_i + t_{i,i+2n} + s_i, \forall i \in P_1 \cup P_2 \quad (3-50)$$

$$e_i \leq T_i \leq l_i, \forall i \in N \quad (3-51)$$

$$x_{ij} \in \{0,1\}, \forall i, j \in N \quad (3-52)$$

$$u_i^k \in \{0,1\}, \forall i \in N, \forall k \in K \quad (3-53)$$

$$Q_i \geq 0, \forall i \in N \quad (3-54)$$

$$T_i \geq 0, \forall i \in N \quad (3-55)$$

$$T_{D_1}, T_{D_2} \geq 0 \quad (3-56)$$

Objective function (3-33) minimizes the maximum task completion time between points D_1 and D_2, as determined by constraints (3−34) and (3−35). Constraints (3−36) and (3−37) define the maximum completion times for reaching points D_1 and D_2, respectively. Constraints (3-38) ensure that the arrival time at point P_2 is no earlier than the time the tote reaches the conveyor exit. Constraints (3-39) address the flow balance for AMRs during the pickup and delivery of totes. Constraint (3-40) requires AMRs to complete all assigned pickup or delivery tasks. Constraints (3-41) through (3-45) stipulate that each task must be completed by the same AMR. Constraint (3-46) specifies that an AMR can depart from its starting point at most once. Constraints (3-47) and (3-48) impose load constraints on AMRs, limiting the maximum number of totes they can carry. Constraint (3-49) states that the time an AMR arrives at the next target point must be no earlier than the time it arrives at the previous point, plus the service time at the previous point and the travel time between the two points. Constraint (3-50) requires that the time a tote arrives at its final destination be no earlier than the time it arrives at its starting point, plus the travel time from the start to the destination and the service time at the start. Constraint (3-51) ensures that the tote's completion time adheres to the specified time window. Finally, constraints (3-52) through (3-56) define the specified variables.

3.3 An Integrated Decision Method

This subsection introduces the JO-OTR model, an integrated framework developed for the joint optimization of order task assignments to picking stations, the assignment of tote retrieval and storage tasks, robot scheduling, and tote scheduling on the conveyor system. The model integrates both the

objective function and constraints by linking decision models across all three stages.

3.3.1 Objective

The objective of the integrated model is to minimize the maximum completion time for all orders within the system, thereby ensuring the quickest possible order fulfillment. The decision models across the three stages collectively aim to minimize four distinct maximum completion times: the maximum completion time for all orders T_1, the maximum time for all totes to reach the conveyor exit T_2, and the maximum time for robots to complete all tasks T_3. Given that the integrated model employs a sequential decision-making process, the following relational expressions can be derived.

$$T_3 \geq T_2 \geq T_1 \tag{3-57}$$

Thus, the optimization objective of the integrated model can be concluded as minimizing the maximum completion time for robots to complete all tasks, as expressed in the following equation:

$$min \quad T \tag{3-58}$$

$$T \geq T_i, \forall i \in N \tag{3-59}$$

3.3.2 Constraints

The constraints of the integrated model include those from the decision-making processes of all three stages, as well as the linking constraints that connect these stages. These linking constraints serve as input parameters for each stage's decision-making, such as t_o^{order}, t_i^{toteIn}, $t_i^{toteOut}$, among others. By converting these parameters into linking constraints, the isolated decision models are effectively integrated into a comprehensive model.

In the first stage of decision-making, the input parameter t_o^{order} is introduced to capture the start and end times of order execution. This parameter is practically related to the start and end times of tote execution in the third stage of decision-making and can be translated into two constraints:

$$t_o^s - T_{i,p}^s \leq \left(3 - OT_{i,o} - y_{i,p} - z_{o,p}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{3-60}$$

$$t_o^e - T_{i,p}^e \geq \left(OT_{i,o} + y_{i,p} + z_{o,p} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{3-61}$$

Where constraints 28 stipulate that the start time of order execution must be less than or equal to the start time of the corresponding tote, while constraints 29 ensure that the end time of order execution must be greater than or equal to the end time of the corresponding tote.

In the second stage of decision-making, the output parameter t_i^{toteIn} is introduced to capture the

time when a tote arrives at the conveyor entrance. In reality, this aligns with the arrival time at point D_1 in the third stage of decision-making and can be converted into the following constraints:

$$I_i \geq T_{i+2n}, \forall i \in P_1 \quad (3-62)$$

Where constraints 30 specify that the time a tote arrives at the conveyor entrance must be greater than or equal to its arrival time at point D_1.

In the third stage of decision-making, the input parameter $t_i^{toteOut}$ is introduced to determine the start time for executing tasks at point P_2. This corresponds to the time a tote arrives at the conveyor exit and can be translated into the following constraints:

$$T_{i,P}^e + t_P^2 \leq T_{i+n}, \forall i \in P_1 \quad (3-63)$$

Where constraints 31 specify that the start time for tasks at point P_2 must be greater than or equal to the time the tote arrives at the last picking station plus the travel time from the last picking station to the conveyor exit.

3.3.3 Mathematical model

In summary, constraints (3-3) are replaced by constraints (3-61) and (3-62), constraints (3-16) are replaced by constraints (3-62), and constraints (3-38) are replaced by constraints (3-63). Additionally, equation (3-58) is adopted as the final objective function, and constraints (3-59) are incorporated to formulate the integrated mathematical model, as detailed below:

$$min \quad T \quad (3-64)$$

$s.t.$

$$(3-4)-(3-12); (3-17)-(3-31); (3-39)-(3-55);$$

$$T \geq T_i, \forall i \in N \quad (3-65)$$

$$t_o^s - T_{i,p}^s \leq \left(3 - OT_{i,o} - y_{i,p} - z_{o,p}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (3-66)$$

$$t_o^e - T_{i,p}^e \geq \left(OT_{i,o} + y_{i,p} + z_{o,p} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (3-67)$$

$$T_{i,P}^e + t_P^2 \leq T_{i+n}, \forall i \in P_1 \quad (3-68)$$

$$I_i \geq T_{i+2n}, \forall i \in P_1 \quad (3-69)$$

$$T \geq 0 \quad (3-70)$$

The objective function (3-64) aims to minimize the maximum completion time for all tasks. Constraints (3-4) through (3-12) pertain to the first stage of decision-making, while constraints (3-17) through (3-31) are associated with the second stage. Constraints (3-39) through (3-55) belong to the

third stage. Constraints (3-66) and (3-67) serve as the linking constraints between the first and second stages. Constraints (3-68) connect the second and third stages, and constraints (3-69) link the third stage back to the second. Finally, constraint (3-70) defines the variables used in the model.

4. Mathematical Programming Based Model Decomposition

In this section, we introduce a framework called Mathematical Programming-based Model Decomposition (MP-MD), specifically designed to address large-scale mixed-integer programming (MIP) models that involve joint optimization of multiple decision variables, such as the JO-OTR model. In complex joint optimization models, certain decision variables often have a significantly greater impact on the model than others. For example, in the JO-OTR model, variables such as x_{ij}, y_{ip}, z_{op} are key decision variables. Attempting to simultaneously optimize these complex key decision variables within a single, comprehensive model would result in a model of such scale that it becomes unsolvable. Therefore, we explored whether these key decision variables could be solved separately within their respective subproblems, with the results from all subproblems unified to apply to the main problem, thereby arriving at a solution to the original problem.

This motivation led us to propose the MP-MD framework for large-scale MIP models involving joint optimization of multiple decisions, such as the JO-OTR model. MP-MD involves rotating and fixing decision variables while solving the model. The main process is as follows: first, for a complex MIP, we identify the key decision variables; next, based on this identification, we construct a corresponding logic-based main problem; then, we create relaxed models for different key decision variables based on the original problem, referred to as subproblems; finally, we accelerate the solution process by deriving bounds for the original problem or subproblems (in the case of JO-OTR, the lower bound since it is a minimization problem) and by setting initial values for the key decision variables in the subproblems. The following sections will detail how we apply the MP-MD framework to solve the JO-OTR model.

4.1 Logic-based Master Problem

Since the subproblems in the MP-MD framework correspond directly to the key variables, and solving each subproblem requires fixing the other key variables, the master problem in MP-MD employs logical constraints to determine which subproblem should be optimized in each iteration and whether

the current decision has reached the optimal solution. In each iteration, the input for the master problem is the subproblem from the previous iteration along with its objective function value, while the output is the subproblem that needs to be optimized in the next iteration. The model for the master problem is presented as follows.

Master problem：

$$min \sum_{s=1}^{S} \theta_s \cdot \mathbb{F}_s^S(\mathcal{X}_s) \quad (4-1)$$

$s.t.$

$$\sum_{s=1}^{S} \theta_s = 1, \forall s \in S \quad (4-2)$$

$$\theta_s \leq 1 - \vartheta_s, \forall s \in S \quad (4-3)$$

$$\theta_s \in \{0,1\}, \forall s \in S \quad (4-4)$$

Let ϑ_s represent the subproblem s most recently optimized in the previous iteration, $\mathcal{X}_s$ denote the key variables of subproblem s, and $\mathbb{F}_s^S(\mathcal{X}_s)$ represent the optimal objective function value of subproblem s given the key variables $\mathcal{X}_s$. The decision variable θ_s来 is used to determine which subproblem will be optimized in the next iteration. The objective function is set to 0, indicating that the subproblem s with the minimum $\mathbb{F}_s^S(\mathcal{X}_s)$ under the current conditions will be chosen for optimization. Constraints 1 ensure that only one subproblem can be selected per iteration. Constraints 2 prevent the rotation of the subproblem that was most recently optimized. Constraints 3 define the decision variable θ_s。

Theorem 1. The Logic-based Master Problem will terminate after a finite number of iterations and yield the optimal solution. At termination, the optimal solution of the Logic-based Master Problem will correspond to the optimal solutions of all subproblems and the original problem, such that $\mathbb{F}^{\mathrm{O}} = \mathbb{F}^{\mathrm{M}} = \mathbb{F}_s^S(\mathcal{X}_s)$.

Proof. The relevant proofs and derivations can be found in Appendix A.

4.2 Relaxation-based Subproblem

For any given subproblem s, its model is derived by fixing all key variables except $\mathcal{X}_s$. This is done by relaxing all key variables other than $\mathcal{X}_s$ from the original model, with the relaxed variables assigned fixed values $\mathcal{X}_s'$ and incorporated into the model's constraints. In the case of JO-OTR, the

models for the three subproblems are presented as follows.

4.2.1 Relaxed Model for Sub-problem 1

Sub-problem 1：Fix x_{ij} and y_{ip} as x'_{ij} and y'_{ip}:

$$min \quad \mathbb{F}_1^S(z_{op}) \tag{4-5}$$

$s.t.$

$$(3-4)-(3-12);(3-19)-(3-22);(3-24)-(3-25);(3-27);(3-29)-(3-31);$$

$$(3-43)-(3-45);(3-48);(3-50)-(3-51);(3-53)-(3-55);(3-68)-(3-69);$$

$$\mathbb{F}_1^S(z_{op}) \geq T_i, \forall i \in N \tag{4-6}$$

$$\sum_{i=1}^{P_1}\sum_{p=1}^{P} y'_{ip} = \sum_{i=1}^{P_1}\sum_{o=1}^{O} OT_{i,o}, \forall o \in O \tag{4-7}$$

$$OT_{i,o} * z_{o,p} \leq y'_{ip}, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{4-8}$$

$$T_{i,p}^e - T_{i,p}^s = p_i * y'_{ip}, \forall i \in P_1, \forall p \in P \tag{4-9}$$

$$T_{j,p}^s - T_{i,p}^e \geq \left(f_{i,j,p} + y'_{ip} + y'_{jp} - 3\right) * M, \forall i,j \in P_1, \forall p \in P \tag{4-10}$$

$$\sum_{j=1,j\neq i}^{N} x'_{ij} = \sum_{j=1,j\neq i}^{N} x'_{ji}, \forall i \in N \tag{4-11}$$

$$\sum_{k=1}^{K}\sum_{j=1,j\neq i}^{N} x'_{ij} \geq 1, \forall i \in P_1 \cup P_2 \cup D_1 \cup D_2 \tag{4-12}$$

$$u_i^k - u_j^k \geq M(x'_{ij} - 1), \forall i \in N, \forall j \in N, \forall k \in K \tag{4-13}$$

$$u_i^k - u_j^k \leq M\left(1 - x'_{ij}\right), \forall i \in N, \forall j \in N, \forall k \in K \tag{4-14}$$

$$\sum_{j=1,j\neq i}^{N} x'_{ij} \leq 1, \forall i \in W, i = W_k \tag{4-15}$$

$$Q_j \geq Q_i + q_i - Q\left(1 - x'_{ij}\right), \forall i \in N \;\; i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \tag{4-16}$$

$$T_j \geq T_i + t_{ij} + s_i - M\left(1 - x'_{ij}\right), \forall i \in N \;\; i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \tag{4-17}$$

$$t_o^s - T_{i,p}^s \leq \left(3 - OT_{i,o} - y'_{ip} - z_{o,p}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{4-18}$$

$$t_o^e - T_{i,p}^e \geq \left(OT_{i,o} + y'_{ip} + z_{o,p} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \tag{4-19}$$

$$\mathbb{F}_1^S\left(z_{op}\right) \geq 0 \tag{4-20}$$

4.2.2 Relaxed Model for Sub-problem 2

Sub-problem 2：Fix x_{ij} and z_{op} as x'_{ij} and z'_{op}:

$$min \quad \mathbb{F}_2^S\left(y_{ip}\right) \tag{4-21}$$

$s.t.$

$$(3-5)-(3-7);(3-9);(3-11)-(3-12);(3-43)-(3-45);(3-17)-(3-31);$$
$$(3-48);(3-50)-(3-51);(3-53)-(3-55);(3-68)-(3-69);$$

$$\mathbb{F}_2^S(y_{ip}) \geq T_i, \forall i \in N \quad (4-22)$$

$$\sum_{p=1}^{P} z'_{op} = 1, \forall o \in O \quad (4-23)$$

$$\delta_{o_1,o_2} \geq \gamma_{o_1,o_2} + z'_{o_1p} + z'_{o_2p} - 2, \forall o_1, o_2 \in O, \forall p \in P \quad (4-24)$$

$$t_o^s - T_{i,p}^s \leq \left(3 - OT_{i,o} - y_{ip} - z'_{op}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (4-25)$$

$$t_o^e - T_{i,p}^e \geq \left(OT_{i,o} + y_{ip} + z'_{op} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (4-26)$$

$$\mathbb{F}_2^S(y_{ip}) \geq 0 \quad (4-27)$$

4.2.3 Relaxed Model for Sub-problem 3

Sub-problem 3：Fix y_{ip} and z_{op}as y'_{ip} and z'_{op}

$$min \quad \mathbb{F}_3^S(x_{ij}) \quad (4-28)$$

$s.t.$

$$(3-5)-(3-7);(3-9);(3-11)-(3-12);(3-19)-(3-22);(3-24)-(3-25);$$
$$(3-27);(3-29)-(3-31);(3-39)-(3-55);(4-7)-(4-10);(4-23)-(4-24);$$

$$\mathbb{F}_3^S(x_{ij}) \geq T_i, \forall i \in N \quad (4-29)$$

$$t_o^s - T_{i,p}^s \leq \left(3 - OT_{i,o} - y'_{ip} - z'_{op}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (4-30)$$

$$t_o^e - T_{i,p}^e \geq \left(OT_{i,o} + y'_{ip} + z'_{op} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (4-31)$$

$$\mathbb{F}_3^S(x_{ij}) \geq 0 \quad (4-32)$$

4.3 Initial Set of Variables

4.3.1 Initial Set of the Order Assignment and Sequence

The assignment and sequencing of orders serve two main purposes. First, to assign orders with similar structures to the same picking station. Second, to ensure that the sequence of orders executed at the same picking station is such that consecutive orders are similar, thereby minimizing the number of totes required to fulfill as many orders as possible. The initial solution construction for order assignment and sequencing decisions consists of two parts. The first part involves using Equation 1 to evaluate the similarity of orders and classify them into categories, which are then assigned to picking stations. The second part employs the greedy algorithm proposed by Qi to sequence the orders assigned to each

picking station. The complete algorithm is presented in Algorithm 1.

$$\psi_{i_1,i_2} = \frac{O_{i_1}^T O_{i_2}}{O_{i_1}^T O_{i_1} + O_{i_2}^T O_{i_2} - O_{i_1}^T O_{i_2}}$$

We define the similarity between two orders as the proportion of the number of common SKUs to the total number of SKUs contained in the two orders. Here, ψ_{i_1,i_2} represents the similarity between orders i_1 and i_2, and $O_{set_{ik}}$ indicates whether order i includes SKU type k种 SKU. Therefore, $O_{i_1} = \left(O_{set_{i1}}, O_{set_{i2}}, \ldots, O_{set_{iK}}\right)^T$.

Algorithm 1: Initial solution of the order assignment and sequence.

Input: $O, P_1, P, OT_{i,o}$

Output: $z_{o,p}, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$

Randomly select P orders as representatives of the initial clusters.
$for\ o \in O$:
  For each order, calculate its similarity to each cluster representation ψ_{o,o_p}.
  Assign orders to the most similar clusters: $max\left(\psi_{o,o_p}\right) \rightarrow z_{op} = 1$.
  Calculate the average similarity of the cluster as a new cluster representative.
$end\ for$.
$for\ p \in P$:
  Select the order with the largest SKU as the first order o_p to be executed.
$end\ for$.
$for\ p \in P$:
  Calculate O_p according to z_{op}
  $for\ o \in O_p$:
    Calculate ψ_{o,o_p} and select $max\left(\psi_{o,o_p}\right) \rightarrow o$ as the next order to be executed.
    Replace o_p with o。
  $end\ for$.
$end\ for$.

Output: $z_{o,p}, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$

4.3.2 Initial Set of the Tote Assignment and Sequence

料 The primary objective of tote assignment and sequencing is to determine which picking stations each tote needs to visit and the order in which totes should be retrieved. After assigning totes based on the order assignments, we determine the retrieval sequence of each tote according to its current level of urgency. The urgency of a tote refers to the number of times it is currently needed by the orders at the picking stations. As described by Equation 3, the more orders that require a specific tote, the higher its urgency, and thus, it should be retrieved sooner. The complete pseudocode for the algorithm is provided in Algorithm 2.

$$\omega_i = \sum_{p=1}^{P} OT_{i,o} \cdot z_{o,p}\,, \forall o \in O_{on}$$

Here, O_{on} represents the orders currently positioned at the picking station slots.

Algorithm 2: Initial solution of the tote assignment and sequence.

Input: $O, P_1, P, OT_{i,o}, z_{o,p}, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$

Output: $y_{i,p}, I_i, f_{i,j,p}$
$i = 0$.
while $i \leq P_1$:
  $for\ \ i \in P_1$:
    Calculate the number of bindings per bin: ω_i.
  $end\ \ for$.
  The $max(\omega_i)$ is used as the currently executed tote.
  Update order status at each picking station.
  $i = i + 1$.
$end\ \ while$.
Output: $y_{i,p}, I_i, f_{i,j,p}$

4.3.3 Initial Set of the Robot Allocation and Scheduling

The objective of robot task assignment and scheduling is to determine which robots will execute specific tote tasks and the precise routes they will follow. Given the similarity to the routing problem, we employ the Adapted Nearest Neighbor (ANN) method to construct the initial solution. ANN is a fast and simple greedy selection strategy that has demonstrated strong performance in routing-related problems. Starting from a defined or randomly chosen initial node, ANN assigns the nearest unvisited node to the robot as the next node, continuing this process until all nodes are included in the route. For the problem addressed in this paper, it is only necessary to ensure that each node insertion satisfies the priority and capacity constraints of the AMRs, thereby guaranteeing that the constructed initial solution is always feasible. The complete pseudocode for the algorithm is provided in Algorithm 3.

Algorithm 3: Initial solution of the robot allocation and scheduling.

Input: $K, P_1, P_2, D_1, D_2, W, N, I_i, f_{i,j,p}$
Output: $x_{i,j}, u_i^k, Q_i, T_i$
$n = 0,\ q_k = 0$.
while $n \leq P_1 + P_2$:
  Select a P_1 or P_2, and assign it to an AMRs.
  $for\ \ k \in K$:
    The current location of AMRs is p.
    $if\ \ q_k \leq Q$:
      When the load of the AMRs is less than Q, a P_1 or P_2 closest to the current position of the AMRs is selected to join in the path of that AMR.
      $x_{p,p_1}\ \ or\ \ x_{p,p_2} = 1, u_{p_1}^k\ \ or\ \ u_{p_2}^k = 1, Q_{p_1}\ \ or\ \ Q_{p_2} = q_k$.
      $n = n + 1, q_k = q_k + 1$.
    $else$:
      Add D_1 or D_2 that already has a P_1 or P_2 counterpart in the path of the AMRs.
      $q_k = 0$.
    $end\ \ if$.
  $end\ \ for$.
$end\ \ while$.
Output: $x_{i,j}, u_i^k, Q_i, T_i$

4.4 Lower Bounds for Original Problem

In the integrated model JO-TOR, the objective value T represents the completion time of the last tote among all totes, specifically the time taken by the tote with the latest completion time. The start time for each tote is defined as the moment it leaves its storage location. It is then transported by a robot

to the conveyor entrance, where it is processed at the picking station, proceeds to the conveyor exit, and finally returns to its storage location with the help of a robot. The sets of starting and ending points for the totes are defined as P_1, P_2, D_1, D_2; the set of robots as R; the set of robot starting points as W; and the set of picking stations as P. The time taken by robot r to travel from its starting point w to the starting point p_1 of a tote is denoted as t_{wp_1}. The time for a tote to travel from its storage location to the conveyor entrance and from the conveyor exit back to its storage location are denoted as $t_{p_1d_1}$ and $t_{p_2d_2}$, respectively. The time a tote spends on the conveyor is denoted as t_c., respectively. The time a tote spends on the conveyor is denoted as t_{p1} , can be expressed by the following equation.

$$t_{p1} = t_{wp_1} + t_{p_1d_1} + t_c + t_{p_2d_2}$$

Since we are deriving the lower bound for the integrated model JO-TOR, the time for robot r to travel from its starting point w to the starting point p_1 of a tote is taken as the minimum value, as shown in the following equation. Similarly, the time a tote spends on the conveyor is also taken as the minimum value, as expressed in the following equation. Therefore, the completion time t_{p1} for each tote can be further modified as shown below:

$$\min_{w \in W}\{t_{wp_1}\}$$

$$t_c = t_P^1 + t_P^2 + p_{p1}$$

$$t_{p1} = \min_{w \in W}\{t_{wp_1}\} + t_{p_1d_1} + t_P^1 + t_P^2 + p_{p1} + t_{p_2d_2}$$

The lower bound for the integrated model JO-TOR is thus the maximum of the completion times for all totes, as shown in the following equation:

$$LB_1 = \max_{p_1 \in P_1}\{\min_{w \in W}\{t_{wp_1}\} + t_{p_1d_1} + t_P^1 + t_P^2 + p_{p1} + t_{p_2d_2}\}$$

Furthermore, when considering the number of robots, we observe that if the number of robots is less than the number of totes, a single robot must complete tasks for multiple totes. In this case, the lower bound of the integrated model JO-TOR becomes the completion time of the last robot to finish its tasks. Let t_1^r denote the time robot r takes to pick up a tote and t_2^r represent the time robot r takes to deliver a tote. The completion time for robot r is then given by the following equation:

$$t_r = t_1^r + t_2^r + t_c$$

Thus, the lower bound for the integrated model JO-TOR is the maximum of the completion times for all robots, as shown in the following equation:

$$LB_2 = \max_{r \in R}\{t_r\}$$

Calculating t_1^r or t_2^r for multiple robots simultaneously is a challenging task. Therefore, we simplify this by converting it into the problem of finding the minimum t_1^r and t_2^r to calculate LB_2, as detailed in Equations (3) to (4). First, we calculate the average number of tote tasks that each robot needs to complete, denoted by $\bar{n}$. Then, we determine the minimum time required for each robot to complete these $\bar{n}$ tote tasks, and we take the smallest of these times as LB_2. During this process, we do not consider the load capacity of the robots, ensuring that the required time represents the lower bound for the integrated model JO-TOR.

$$min \quad t_1^r$$

$s.t.$

$$t_1^r \geq T_i, \forall i, j \in W \cup P_1 \cup D$$

$$\sum_{j \in W \cup P_1} v_{ji} \geq \tau_i, \forall i \in P_1$$

$$\sum_{j \in P_1 \cup D} v_{ij} \geq \tau_i, \forall i \in P_1$$

$$\sum_{j \in P_1} v_{ij} = 1, i = W$$

$$\sum_{j \in P_1} v_{ji} = 1, i = D$$

$$\sum_{i \in P_1} \tau_i \geq \bar{n}$$

$$T_j \geq T_i + t_{ij} + s_i - M(1 - v_{ij}), \forall i, j \in W \cup P_1 \cup D$$

$$\tau_i \in \{0,1\}, \forall i \in P_1$$

$$v_{ij} \in \{0,1\}, \forall i, j \in W \cup P_1 \cup D$$

$$T_i \geq 0, \forall i \in W \cup P_1 \cup D$$

Here, τ_i indicates whether tote i indicates whether tote, v_{ij} represents whether the robot travels from point i to point j, and T_i denotes the time the robot arrives at point i. The objective function aims to minimize the robot's pickup time t_1^r. Constraints 1 represent the robot's completion time, Constraints 2-3 define the outdegree and indegree at tote points, Constraints 4-5 specify the outdegree and indegree at the starting or ending points, Constraints 6 are the time propagation constraints, and Constraints 7-9 define the range of decision variables.

Using this model, we can calculate each robot's completion time t_1^r. Using this model, we can

calculate each robot's completion time t_2^r, as these are identical problems. Therefore, the lower bound for the integrated model JO-TOR can be set as the minimum of t_1^r and t_2^r, as shown in Equation (10).

$$LB_2 = \min_{r \in R}\{t_1^r\} + t_c + \min_{r \in R}\{t_2^r\}$$

5. Neural-enhanced Revolving Math-heuristic Algorithm for Integrated Method

This section provides a detailed explanation of the "Neural-enhanced Revolving Math-heuristic Algorithm for Integrated Method" (DeepRMHA), a specially designed algorithm developed to solve the large-scale integrated model JO-TOR within a short time frame using the MP-MD method we proposed. DeepRMHA is a two-layer solution framework that iteratively fixes variables. The core concept of DeepRMHA is illustrated in Figure 1. Variables are categorized into multiple groups, and some are temporarily fixed in a rotating manner during each iteration. The master problem's mathematical heuristic framework controls the algorithm's iterations and termination, determining which key variables to fix and relax. The subproblem's neural-enhanced sub-algorithm is designed to solve the relaxed subproblem models.

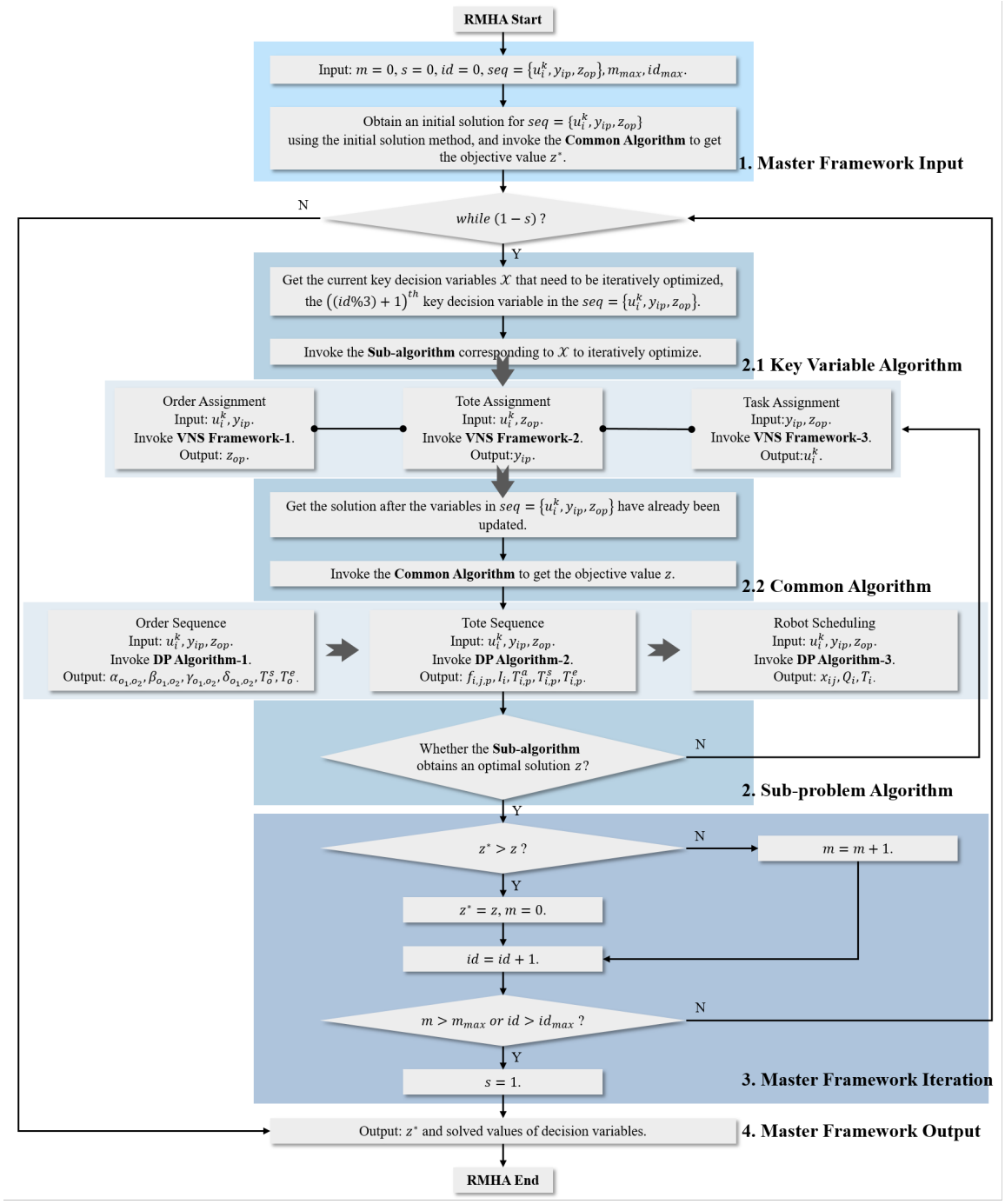

Figure 1. RMHA framework.

5.1 Math-heuristic Framework for Master-problem

The role of the mathematical heuristic framework in the primary problem within DeepRMHA is to determine which key variables to fix at each iteration, update the problem's upper and lower bounds, and control the start and end of the algorithm. The algorithm begins by initializing the solution, then iteratively fixes key variables and updates the solution's bounds at each step. The algorithm terminates

and outputs the final solution when either the maximum number of iterations is reached or the exit conditions are met. The pseudocode for this process is provided in Algorithm 4.

Algorithm 4: Math-heuristic framework for master-problem.	
Input:	$m, s, id, seq, m_{max}, id_{max}$.
Output:	Optimal objective value z^* and solved values of decision variables.
1	$m = 0,\ s = 0,\ id = 0,\ seq = (u_i^k, y_{ip}, z_{op})$.
2	Obtain an initial solution for $seq = (u_i^k, y_{ip}, z_{op})$ using the initial solution method, and invoke the **Common Algorithm** to get the objective value z^*.
3	$while\ (1 - s)$:
4	Get the current key decision variables $\mathcal{X}$ that need to be iteratively optimized, the $((id\%3) + 1)^{th}$ key decision variable in the $seq = (u_i^k, y_{ip}, z_{op})$.
5	Invoke the **Sub-algorithm** corresponding to $\mathcal{X}$ to iteratively optimize.
6	Obtain the **Sub-algorithm** optimised solution z.
7	$if\ (z < z^*)$:
8	$z^* = z,\ m = 0$.
9	$else$:
10	$m = m + 1$.
11	$end\ if$.
12	$if\ (m \geq m_{max}\ or\ id \geq id_{max})$:
13	$s = 1$.
14	$end\ if$.
15	$end\ while$.
Output:	Optimal objective value z^* and solved values of decision variables.

5.2 Sub-algorithms for Sub-problems

In DeepRMHA, we design three sub-algorithms to solve the relaxed models of the three subproblems individually. Each subproblem's relaxed model is solved in two parts. The first part uses a Variable Neighborhood Search (VNS) framework to explore the neighborhood of the key decision variable values for each subproblem. Once the neighborhood of the key decision variables and the values of the other two fixed decision variables are obtained, a general algorithm based on dynamic programming is designed to solve the remaining integer and continuous decision variables in the model. The detailed content of this Common Algorithm is discussed in Section 5.3.

For the key decision variables of each subproblem, which are all assignment problems, we have designed a generic subproblems algorithm based on the VNS framework to solve all three subproblems. This includes generating the initial solution, neighborhood shaking and local search, and the acceptance and output of new solutions. The pseudocode for this process is provided in Algorithm 5.

Algorithm 5: Sub-algorithm for sub-problem.	
Input:	$z_{o,p}, y_{i,p}(z_{o,p}, u_i^k$ or $y_{i,p}, u_i^k)$, $count_{max}$, $iter_{max}$, a set of neighbourhood structures $N_k\ for\ k = 1,2, \dots, K$ for shaking, a set of neighbourhood structures $N_l\ for\ l = 1,2, \dots, L$ for local search.
Output:	Optimal objective value z^* and $u_i^k(y_{i,p}$ or $z_{o,p})$ and $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}, I_i, f_{i,j,p}, x_{i,j}, Q_i, T_i$.
1	Initialize $u_i^k(y_{i,p}$ or $z_{o,p})$ according to the solution obtained in the previous step.
2	Invoke **Common Algorithm**, obtain $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}, I_i, f_{i,j,p}, x_{i,j}, Q_i$, and obtain objective value z of the initial solution.
3	Set $z^* = z,\ z_{min} = z$.
4	Set $count = 0,\ iter = 0$.

5	$while\ \ iter \le iter_{max}$:
6	$k = 1$.
7	$while\ \ k \le K$:
8	Shaking:
9	$u_i^{k'}(y'_{i,p}\ \text{or}\ z'_{o,p}) = Shake(u_i^k(y_{i,p}\ \text{or}\ z_{o,p}), k)$, obtain updated $u_i^{k'}(y'_{i,p}\ \text{or}\ z'_{o,p})$, and invoke **Common Algorithm**, obtain objective value z'.
10	Local search:
11	$l = 1$.
12	$while\ \ l \le L$:
13	$u_i^{k''}(y''_{i,p}\ \text{or}\ z''_{o,p}) = LS(u_i^{k'}(y'_{i,p}\ \text{or}\ z'_{o,p}), l)$.
14	Invoke **Common Algorithm**, obtain objective value z''.
15	$if\ \ z'' \le z'$:
16	$u_i^{k'}(y'_{i,p}\ \text{or}\ z'_{o,p}) = u_i^{k''}(y''_{i,p}\ \text{or}\ z''_{o,p}),\ l = 1$.
17	$else$:
18	$l = l + 1$.
19	$end\ \ if$.
20	$end\ \ while$.
21	Move or not.
22	$if\ \ z'' \le z$:
23	$u_i^k(y_{i,p}\ \text{or}\ z_{o,p}) = u_i^{k''}(y''_{i,p}\ \text{or}\ z''_{o,p}),\ z_{min} = z'',\ k = 1$.
24	$else$:
25	$k = k + 1$.
26	$end\ \ if$.
27	$end\ \ while$.
28	$if\ \ z_{min} \le z^*$:
29	$z^* = z_{min},\ count = 0$.
30	$else$:
31	$count = count + 1$.
32	$end\ \ if$.
33	$if\ \ count \le count_{max}$:
34	$break$.
35	$end\ \ if$.
36	$iter = iter + 1$.
37	$end\ \ while$.
Output:	Optimal objective value z^* and $u_i^k(y_{i,p}\ \text{or}\ z_{o,p})$ and $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}, I_i, f_{i,j,p}, x_{i,j}, Q_i, T_i$.

5.3 Dynamic Programming Based Common Algorithm for All Sub-problems

In the sub-algorithms described in Section 5.2, the Common Algorithm is invoked multiple times. This Common Algorithm involves solving the remaining decision variables once the values of the three key decision variables $u_i^k, y_{i,p}, z_{o,p}$ are known. The process of solving these remaining variables can be divided into three parts: (1) solving for $x_{i,j}$ given u_i^k; (2) solving for $f_{i,j,p}$ given $y_{i,p}, I_i$; and (3) solving for $T_o^s, T_o^e, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$ given $z_{o,p}, T_{i,p}^a, T_{i,p}^s, T_{i,p}^e$. The sequence of solving these parts is as follows: First, solve part 1 to determine the robot's path and the time it arrives at each task point, which in turn determines the flow of totes within the warehouse. Next, based on the tote flow, solve part 2 to determine the flow of totes on the conveyor. Finally, using the tote flow on the conveyor, solve part 3 to determine the order picking process at each picking station. This completes the solution of all decision variables.

In addition to the decision variables mentioned above, the Common Algorithm must also solve for

all remaining decision variables in the problem, such as time variables and auxiliary variables. The pseudocode for this process is provided in Algorithm 6.

Algorithm 6: Dynamic programming based Common Algorithm for sub-algorithm.

Input: $z_{o,p}, y_{i,p}, u_i^k$.

Output: Objective value z and $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}, I_i, f_{i,j,p}, x_{i,j}, Q_i, T_i$.

Based on u_i^k and equations (5-1)-(5-8), invoke the **Dynamic Programming Algorithm 1** and calculate $x_{i,j}$.
$for\ \ i,j \in N$:
  $if\ \ i! = j$:
    According to $Q_j \geq Q_i + q_i - Q(1 - x_{ij})$, calculate Q_i.
    According to $T_j \geq T_i + t_{ij} + s_i - M(1 - x_{ij})$ and $T_{i+2n} \geq T_i + t_{i,i+2n} + s_i$, calculate T_i.
  $end\ \ if.$
$end\ \ for.$
$for\ \ i \in P_i$:
  According to $I_i \geq T_{i+2n}$, calculate I_i.
$end\ \ for.$
Based on $y_{i,p}, I_i$ and equations (3-19)-(3-27), invoke the **Dynamic Programming Algorithm 2** and calculate $T_{i,p}^a, T_{i,p}^s, T_{i,p}^e, f_{i,j,p}$.
$for\ \ i \in P_i$:
  $if\ \ T_{i,P}^e + t_P^2 \leq T_{i+n}$ is not satisfied:
    $z = +\infty$, return.
  $end\ \ if.$
$end\ \ for.$
$for\ \ p \in P, i \in P_i$:
  $if\ \ T_{i,p}^s - T_{i,p}^a \leq (b_2 - 1) * p_i$ is not satisfied:
    $z = +\infty$, return.
  $end\ \ if.$
$end\ \ for.$
Based on $z_{o,p}, T_{i,p}^a, T_{i,p}^s, T_{i,p}^e$ and equations (3-5)-(3-8), (3-60)-(3-61), invoke the **Dynamic Programming Algorithm 3** and calculate $T_o^s, T_o^e, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$.
$for\ \ o_1 \in O$:
  $if\ \ \sum_{o_2}^{O} \delta_{o_1,o_2} \leq b_1$ is not satisfied:
    $z = +\infty$, return.
  $end\ \ if.$
$end\ \ for.$
$for\ \ i \in N$:
  $z \geq T_i$.
$end\ \ for.$

Output: Objective value z and $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}, I_i, f_{i,j,p}, x_{i,j}, Q_i, T_i$.

The Common Algorithm requires three dynamic programming algorithms, each corresponding to one of the three parts. In the following sections, we will analyze each part individually and design a dynamic programming algorithm for each.

The first part involves solving for $x_{i,j}$ given u_i^k. For the decision-making process of each robot k, the starting point can be set as the endpoint simultaneously, converting it into a Traveling Salesman Problem (TSP) with visit order constraints and resource limitations. The mathematical model is presented in Equations (5-1) to (5-8).

$$min \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} d_{ij}\, x_{ij} \tag{5-1}$$

$$s.t. \sum_{i=1}^{N} x_{ij} = 1, \forall j \in \{2,3,\dots,N+1\}, i \neq j \tag{5-2}$$

$$\sum_{j=2}^{N+1} x_{ij} = 1, \forall i \in \{1,2,\dots,N\}, i \neq j \tag{5-3}$$

$$\mu_i - \mu_j + Nx_{ij} \leq N - 1, \forall i \in \{1,2,\dots,N\}, \forall j \in \{2,3,\dots,N+1\}, i \neq j \quad (5-4)$$

$$\mu_j - \mu_i + N\omega_{ij} \geq N + 1, \forall i \in \{1,2,\dots,N\}, \forall j \in \{2,3,\dots,N+1\}, i \neq j \quad (5-5)$$

$$x_{ij} \in \{0,1\}, \forall i,j \in \{1,2,\dots,N+1\} \quad (5-6)$$

$$\omega_{ij} \in \{0,1\}, \forall i,j \in \{1,2,\dots,N+1\} \quad (5-7)$$

$$\mu_i \geq 0, \forall i \in \{1,2,\dots,N+1\} \quad (5-8)$$

In this model, N represents the number of points, 1 is the starting point of the robot, $N+1$ is the robot's virtual endpoint (the same as the starting point), and d_{ij} is the distance between points i and j. . The objective function (5-1) minimizes the travel distance. Constraints (5-2) and (5-3) ensure that the indegree and outdegree are both equal to 1. Constraint (5-4) eliminates sub-tours, constraint (5-5) enforces the visit order, and constraints (5-6) to (5-8) define the range of the decision variables.

To solve this model, we have designed a dynamic programming algorithm, with the state transition equation given in (5-9) and the pseudocode provided in Algorithm 7.

Algorithm 7: Dynamic programming algorithm 1.

Input: Graph $G = (V, A)$, start node $s = 1$.
Output: Optimal routing r^*.
$for\ \ i = 2,3,\dots,|V|$:
  $f(i,\emptyset) \leftarrow d_{i,s}$.
  The next node of $(i,\emptyset) \leftarrow s$.
$end\ \ for$.
$for\ \ k = V, V-1, V-2, \dots, 2$:
  $for\ \ current\ \ node\ \ i = 2,3,\dots,|V|$:
    $for\ \ S_l \subset V\backslash\{i\}\ \ and\ \ |S_l| = |V| - k$:
      Calculate the current load Q_i of the robot.
      $if\ \ i\ \ is\ \ delivery\ \ node\ \ and\ \ i's\ \ pick\ \ up\ \ node\ \ \in S_l$:
        $break$.
      $end\ \ if$.
      $if\ \ Q_i > Q$:
        $break$.
      $end\ \ if$.
      $f(i,S_l) = min(d_{i,v} + f(v, S_l - \{v\}))$.
      The next node of $(i,S_l) \leftarrow \underset{v}{arg}\ \underset{v \in S_l}{min}(d_{i,v} + f(v, S_l - \{v\}))$.
    $end\ \ for$.
  $end\ \ for$.
$end\ \ for$.
$S_l \leftarrow V - \{s\}$.
$f(s,S_l) \leftarrow \underset{v \in S_l}{min}(d_{s,v} + f(v, S_l - \{v\}))$.
The next node of $(s,S_l) \leftarrow \underset{v}{arg}\ \underset{v \in S_l}{min}(d_{i,v} + f(v, S_l - \{v\}))$.
Obtain the optimal routing r^* by checking the next node from the current node.
Output: Optimal routing r^* and optimal objective $f(s,S_l)$.

The second part involves solving for $T^a_{i,p}, T^s_{i,p}, T^e_{i,p}, f_{i,j,p}$ given $y_{i,p}, I_i$. This is a straightforward sequencing problem. We construct a forward dynamic programming algorithm based on Equations (3-19) through (3-27). The pseudocode is provided in Algorithm 8.

Algorithm 8: Dynamic programming algorithm 2.

Input: $y_{i,p}, I_i$.
Output: $T^a_{i,p}, T^s_{i,p}, T^e_{i,p}, f_{i,j,p}$.

According to I_i, sort the totes and get $seq_{tote} = \{s_1, s_2, \dots, s_n\}$.
$for\ i \in seq_{tote}$:
  $for\ p \in P$:
    $for\ j \in (i+1, seq_{tote})$:
      $f_{i,j,p} = 1$.
    $end\ for$.
    $if\ p = 1$:
      According to $T^a_{i,1} = I_i + t^1_1$, calculate $T^a_{i,1}$.
      $if\ i = 1$:
        $T^s_{1,1} = T^a_{1,1}$.
      $else$:
        $if\ T^a_{i,1} \geq T^s_{i-1,1}$:
          $T^s_{i,1} = T^a_{i,1}$.
        $else$:
          $T^s_{i,1} = T^s_{i-1,1}$.
        $end\ if$.
      $end\ if$.
      According to $T^e_{i,1} - T^s_{i,1} = p_i * y_{i,1}$, calculate $T^e_{i,1}$.
    $else$:
      According to $T^a_{i,p} = T^e_{i,p-1} + t^1_p - t^1_{p-1}$, calculate $T^a_{i,p}$.
      $if\ i = 1$:
        $T^s_{1,p} = T^a_{1,p}$.
      $else$:
        $if\ T^a_{i,p} \geq T^s_{i-1,p}$:
          $T^s_{i,p} = T^a_{i,p}$.
        $else$:
          $T^s_{i,p} = T^s_{i-1,p}$.
        $end\ if$.
      $end\ if$.
      According to $T^e_{i,p} - T^s_{i,p} = p_i * y_{i,p}$, calculate $T^e_{i,p}$.
    $end\ if$.
  $end\ for$.
$end\ for$.

Output: $T^a_{i,p}, T^s_{i,p}, T^e_{i,p}, f_{i,j,p}$.

The third part involves solving for $T^s_o, T^e_o, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$ given $z_{o,p}, T^a_{i,p}, T^s_{i,p}, T^e_{i,p}$. For each picking station p, this part of the decision-making process is also a straightforward sequencing problem, similar to the second part. We construct a forward dynamic programming algorithm based on Equations (3-5) to (3-8) and (3-60) to (3-61). The pseudocode is provided in Algorithm 9.

Algorithm 9: Dynamic programming algorithm 3.

Input: $z_{o,p}, T^a_{i,p}, T^s_{i,p}, T^e_{i,p}$.
Output: $T^s_o, T^e_o, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$.

$for\ o \in O, p \in P$:
  $if\ z_{o,p} == 1$.
    $T^s_o = min_{i \in P_1}(OT_{io} \cdot T^s_{i,p})$.
    $T^e_o = max_{i \in P_1}(OT_{io} \cdot T^e_{i,p})$.
  $end\ if$.
$end\ for$.
$for\ o_1, o_2 \in O$:
  According to $T^s_{o_1} - T^s_{o_2} \leq \alpha_{o_1,o_2} * M,\ T^e_{o_2} - T^s_{o_1} \leq \beta_{o_1,o_2} * M, \forall o_1, o_2 \in O,\ \gamma_{o_1,o_2} \geq \alpha_{o_1,o_2} + \beta_{o_1,o_2} - 1, \forall o_1, o_2 \in O$, calculate $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}$.
$end\ for$.
$for\ o_1, o_2 \in O, p \in P$:
According to $\delta_{o_1,o_2} \geq \gamma_{o_1,o_2} + z_{o_1,p} + z_{o_2,p} - 2$, calculate δ_{o_1,o_2}.
$end\ for$.

Output: $T^s_o, T^e_o, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$.

6. Numerical Experiments

6.1 Experiment Settings

We randomly generated a series of benchmark instances for various experiments, including small-scale, medium-scale, and large-scale instances, as shown in Table 1. Each benchmark instance setup includes five key elements: the number of blocks in the warehouse layout along the length and width directions (W and L), the total number of totes to be retrieved (N), the total number of order tasks to be completed (O), the number of ACRs (R), and the number of picking stations (P). Each block in the warehouse has a length of 18, a width of 2, and a height of 10, resulting in 360 tote storage locations per block.

Table 1. Information on instances.

S-s ID	W	L	N	O	R	P	M-s ID	W	L	N	O	R	P	L-s ID	W	L	N	O	R	P
I-1	2	4	2	2	2	2	II-1	6	8	12	10	6	4	III-1	10	12	30	20	10	5
I-2	2	4	3	2	2	2	II-2	6	8	14	10	6	4	III-2	10	12	60	30	15	5
I-3	2	4	4	2	2	2	II-3	6	8	16	10	6	4	III-3	10	12	90	40	20	5
I-4	4	4	5	5	4	4	II-4	8	8	18	15	8	6	III-4	12	12	110	50	25	10
I-5	4	4	6	5	4	4	II-5	8	8	20	15	8	6	III-5	12	12	140	60	30	10
I-6	4	4	7	5	4	4	II-6	8	8	22	15	8	6	III-6	12	12	170	70	35	10
I-7	6	6	8	8	6	6	II-7	10	8	24	20	10	8	III-7	14	12	210	80	40	15
I-8	6	6	9	8	6	6	II-8	10	8	26	20	10	8	III-8	14	12	240	90	45	15
I-9	6	6	10	8	6	6	II-9	10	8	28	20	10	8	III-9	14	12	270	100	50	15

6.2 Comparison of the Three Stage Separate Decision Method and the Integrated Decision Method

To evaluate our algorithm and the advantages of joint optimization across multiple decisions, we conducted experiments on small, medium, and large scales. We directly compared our proposed MPMD with Gurobi. Additionally, we compared the integrated MPMD and RMHA with the results obtained by optimizing each decision in the system separately. Specifically, we utilized two specially designed fast decision methods, R* and G*, based on rule-based and greedy algorithms, respectively, to represent the outcomes of separately optimizing each decision.

6.2.1 Small-scale Instance

The results of the small-scale instances are presented in Tables 2 and 3, with 10 instances in each category. We use the average of these results as the final outcome. The experimental results indicate that for small-scale problems, both MPMD and RMHA achieve optimal solutions equivalent to those of Gurobi. However, in terms of solution time, both MPMD and RMHA demonstrate a significant speed advantage over Gurobi in most cases. Meanwhile, the R* and G* algorithms, which optimize the three

stages separately, show noticeable gaps compared to MPMD and RMHA, with differences of 15.20% and 21.73% from the optimal solution, respectively. This underscores the advantages and necessity of jointly optimizing multiple decisions.

Table 2. Gurobi and MPMD results for small-scale instances.

Instance ID	Gurobi				MPMD						
	UB_G	LB_G	T_G	G^*	UB_M	LB_M	T_M	Δ_{UB}^{Obj}	Δ_{LB}^{Obj}	Δ_G^{Time}	M^*
I-1	573.00	573.00	0.05	10/10	573.00	573.00	0.09	0.00%	0.00%	0.04	10/10
I-2	573.00	573.00	0.23	10/10	573.00	573.00	0.24	0.00%	0.00%	0.01	10/10
I-3	597.40	597.40	2.27	10/10	597.40	597.40	1.48	0.00%	0.00%	-0.79	10/10
I-4	678.00	678.00	1.37	10/10	678.00	678.00	0.48	0.00%	0.00%	-0.89	10/10
I-5	683.50	683.50	6.38	10/10	683.50	683.50	0.71	0.00%	0.00%	-5.67	10/10
I-6	683.50	683.50	32.61	10/10	683.50	683.50	0.94	0.00%	0.00%	-31.67	10/10
I-7	684.00	684.00	34.07	10/10	684.00	684.00	3.44	0.00%	0.00%	-30.63	10/10
I-8	684.00	684.00	137.31	10/10	684.00	684.00	13.90	0.00%	0.00%	-123.41	10/10
I-9	684.00	684.00	341.76	10/10	684.00	684.00	13.19	0.00%	0.00%	-328.57	10/10
Average	648.93	648.93	61.78	10/10	648.93	648.93	3.83	0.00%	0.00%	-57.95	10/10

Note. $G^*(M^*)$*: the number of optimal solutions obtained by Gurobi (MPMD);* $\Delta_{UB}^{Obj} = (UB_G - UB_M)/UB_G$*;* $\Delta_{LB}^{Obj} = (LB_G - LB_M)/LB_G$*;* $\Delta_G^{Time} = T_G - T_M$.

Table 3. RMHA, R* and G* Results for small-scale instances.

Instance ID	Integrated algorithm MPMD		Integrated algorithm RMHA		3 stages rule-based algorithm R^3		3 stages greedy algorithm G^3		Gap		
	O_M	T_M	O_R	T_R	O_{R^3}	T_{R^3}	O_{G^3}	T_{G^3}	Δ_R^{Obj}	$\Delta_{R^3}^{Obj}$	$\Delta_{G^3}^{Obj}$
I-1	573.00	0.09	573.00	9.44	577.00	0.00	576.00	0.00	0.00%	-0.70%	-0.52%
I-2	573.00	0.24	573.00	10.32	658.00	0.00	663.00	0.00	0.00%	-14.83%	-15.71%
I-3	597.40	1.48	597.40	12.30	659.00	0.00	693.60	0.00	0.00%	-10.31%	-16.10%
I-4	678.00	0.48	678.00	15.25	699.00	0.00	706.00	0.00	0.00%	-3.10%	-4.13%
I-5	683.50	0.71	683.50	17.45	767.00	0.00	832.00	0.00	0.00%	-12.22%	-21.73%
I-6	683.50	0.94	683.50	18.52	767.00	0.00	832.00	0.00	0.00%	-12.22%	-21.73%
I-7	684.00	3.44	684.00	18.20	741.60	0.00	710.00	0.00	0.00%	-8.42%	-3.80%
I-8	684.00	13.90	684.00	19.02	741.60	0.00	755.00	0.00	0.00%	-8.42%	-10.38%
I-9	684.00	13.19	684.00	18.97	788.00	0.00	762.00	0.00	0.00%	-15.20%	-11.40%
Average	648.93	3.83	648.93	15.50	710.91	0.00	725.51	0.00	0.00%	-9.55%	-11.80%

Note. $\Delta_R^{Obj} = (O_M - O_R)/O_M$*;* $\Delta_{R^3}^{Obj} = (O_M - O_{R^3})/O_M$ $\Delta_{G^3}^{Obj} = (O_M - O_{G^3})/O_M$.

6.2.2 Medium-scale Instance

For the medium-scale problems, we use Gurobi, MPMD, RMHA, R*, and G* to solve the instances, with the results shown in Table 4. We observe that while Gurobi is able to provide lower bounds for the problem despite the large number of decision variables and constraints, it cannot provide upper bounds in most cases. This indirectly highlights the complexity of our model. However, it is encouraging to find that even with a large number of decision variables and constraints, MPMD provides both upper and lower bounds for the problem, with the gap between them being relatively small, within 5%. This reflects the efficiency of our MPMD method. We can reasonably speculate that the result obtained by MPMD is close to the optimal solution. Similar to the results observed in the small-scale experiments, the results of the three-stage joint optimization are significantly better than those obtained by optimizing the stages

separately. Moreover, compared to MPMD, RMHA significantly reduces the solution time while finding an approximate optimal solution, as shown in Table 5.

Table 4. Gurobi and MPMD results for medium-scale instances.

Instance ID	Gurobi				MPMD						
	UB_G	LB_G	T_G	G^*	UB_M	LB_M	T_M	Δ_{UB}^{Obj}	Δ_{LB}^{Obj}	Δ_G^{Time}	M^*
II-1	892.00	892.00	742.02	10/10	892.00	892.00	83.30	0.00%	0.00%	-658.72	10/10
II-2	\	856.60	3600.00	0/10	892.00	882.30	3600.00	\	-2.91%	0.00	0/10
II-3	\	849.50	3600.00	0/10	892.10	892.10	1854.32	\	-4.78%	0.00	10/10
II-4	\	848.00	3600.00	0/10	934.70	891.30	3600.00	\	-4.86%	0.00	0/10
II-5	\	904.00	3600.00	0/10	937.60	895.90	3600.00	\	0.90%	0.00	0/10
II-6	\	843.00	3600.00	0/10	941.00	898.80	3600.00	\	-6.21%	0.00	0/10
II-7	\	834.00	3600.00	0/10	931.00	899.20	3600.00	\	-7.25%	0.00	0/10
II-8	\	831.00	3600.00	0/10	931.00	897.00	3600.00	\	-7.36%	0.00	0/10
II-9	\	830.00	3600.00	0/10	940.40	897.00	3600.00	\	-7.47%	0.00	0/10
Average	\	854.23	3282.45	1/9	921.31	893.96	3015.29	\	-4.44%	-73.19	2/9

Table 5. RMHA, R* and G* Results for medium-scale instances.

Instance ID	Integrated algorithm MPMD		Integrated algorithm RMHA		3 stages rule-based algorithm R^3		3 stages greedy algorithm G^3		Gap		
	O_M	T_M	O_R	T_R	O_{R^3}	T_{R^3}	O_{G^3}	T_{G^3}	Δ_R^{Obj}	$\Delta_{R^3}^{Obj}$	$\Delta_{G^3}^{Obj}$
II-1	892.00	83.30	893.40	26.97	1179.00	0.00	910.00	0.00	-0.16%	-32.17%	-2.02%
II-2	892.00	3600.00	894.50	36.11	1361.60	0.00	950.00	0.00	-0.28%	-52.65%	-6.50%
II-3	892.10	1854.32	898.40	39.27	1361.60	0.00	970.50	0.00	-0.71%	-52.63%	-8.79%
II-4	934.70	3600.00	937.30	47.08	1306.00	0.00	1134.00	0.00	-0.28%	-39.72%	-21.32%
II-5	937.60	3600.00	938.30	60.23	1313.20	0.00	1126.00	0.00	-0.07%	-40.06%	-20.09%
II-6	941.00	3600.00	940.40	71.24	1313.20	0.00	1123.80	0.00	0.06%	-39.55%	-19.43%
II-7	931.00	3600.00	932.10	106.38	1216.30	0.00	1098.00	0.00	-0.12%	-30.64%	-17.94%
II-8	931.00	3600.00	938.60	74.11	1233.00	0.00	1077.10	0.00	-0.82%	-32.44%	-15.69%
II-9	940.40	3600.00	940.60	92.04	1400.00	0.00	1108.00	0.00	-0.02%	-48.87%	-17.82%
Average	921.31	3015.29	923.73	61.49	1298.21	0.00	1055.27	0.00	-0.26%	-40.91%	-14.54%

6.2.3 Large-scale Instance

For the large-scale problems, we use Gurobi, MPMD, RMHA, R*, and G* to solve the instances, with the results shown in Table 6. It is gratifying to see that even with a large number of decision variables and constraints, MPMD provides upper bounds in all cases and lower bounds in four cases, while Gurobi cannot provide upper bounds in any case and only provides lower bounds in two cases. This highlights the advantage of our MPMD method. Furthermore, compared to MPMD, RMHA significantly reduces the solution time while finding an approximate optimal solution, as shown in Table 7. The approximate optimal solution provided by RMHA is far superior to those obtained by separately optimizing the three-stage decisions within an acceptable time frame.

Table 6. Gurobi and MPMD results for large-scale instances.

Instance ID	Gurobi				MPMD						
	UB_G	LB_G	T_G	G^*	UB_M	LB_M	T_M	Δ_{UB}^{Obj}	Δ_{LB}^{Obj}	Δ_G^{Time}	M^*
III-1	\	975.00	3600.00	0/10	1131.20	975.00	3600.00	\	0.00%	0	0/10
III-2	\	1023.00	3600.00	0/10	1198.50	1023.00	3600.00	\	0.00%	0	0/10
III-3	\	\	3600.00	0/10	1410.90	1033.00	3600.00	\	\	0	0/10
III-4	\	\	3600.00	0/10	1627.50	1019.00	3600.00	\	\	0	0/10
III-5	\	\	3600.00	0/10	1618.10	\	3600.00	\	\	0	0/10
III-6	\	\	3600.00	0/10	1914.40	\	3600.00	\	\	0	0/10

III-7	\	\	3600.00	0/10	1959.00	\	3600.00	\	\	0	0/10
III-8	\	\	3600.00	0/10	1893.20	\	3600.00	\	\	0	0/10
III-9	\	\	3600.00	0/10	1907.80	\	3600.00	\	\	0	0/10
Average	\	\	3600.00	0/10	1628.96	\	3600.00	\	\	0	0/10

Table 7. RMHA, R* and G* Results for large-scale instances.

Instance ID	Integrated algorithm MPMD		Integrated algorithm RMHA		3 stages rule-based algorithm R^3		3 stages greedy algorithm G^3		Gap		
	O_M	T_M	O_R	T_R	O_{R^3}	T_{R^3}	O_{G^3}	T_{G^3}	Δ_R^{Obj}	$\Delta_{R^3}^{Obj}$	$\Delta_{G^3}^{Obj}$
III-1	1131.20	3600.00	1128.40	70.09	1784.00	0.00	1422.00	0.00	0.25%	-57.71%	-25.71%
III-2	1198.50	3600.00	1166.20	80.06	2882.00	0.00	1564.00	0.00	2.70%	-140.47%	-30.50%
III-3	1410.90	3600.00	1184.70	202.18	3141.00	0.00	1708.20	0.01	16.03%	-122.62%	-21.07%
III-4	1627.50	3600.00	1207.00	786.61	4492.00	0.00	1963.00	0.02	25.84%	-176.01%	-20.61%
III-5	1618.10	3600.00	1213.60	1207.16	3848.00	0.00	1765.30	0.02	25.00%	-137.81%	-9.10%
III-6	1914.40	3600.00	1277.40	1802.32	4374.00	0.00	1916.00	0.04	33.27%	-128.48%	-0.08%
III-7	1959.00	3600.00	1313.50	1902.06	4574.00	0.00	2007.00	0.05	32.95%	-133.49%	-2.45%
III-8	1893.20	3600.00	1330.20	1987.27	4472.00	0.00	1941.00	0.08	29.74%	-136.21%	-2.52%
III-9	1907.80	3600.00	1274.30	2028.35	4412.00	0.00	1910.00	0.12	33.21%	-131.26%	-0.12%
Average	1628.96	3600.00	1232.81	1118.46	3775.44	0.00	1799.61	0.04	24.32%	-131.77%	-10.48%

6.3 Impact of Order, tote, robot Composition and Warehouse Configurations

For sensitivity analysis, we conducted experiments by varying the warehouse's length and width, the numbers of orders and totes, and the numbers of robots and picking stations. We designed nine scenarios for each factor, repeating the tests multiple times for each scenario. We calculated the average times required (ATRs) for each scenario, considering the mean of these repetitions as the final result.

5.4.1 Varying the Warehouse's Length and Width

The impact of warehouse dimensions on system efficiency is summarized in Appendix B, Table B1 and Table B2. As shown in Figure 2, the length of the warehouse has a much greater effect on system efficiency compared to the width of the warehouse. This is because in our warehouse layout, picking stations are located on the left side of the warehouse. As the length of the warehouse increases, the average travel time for robots increases significantly, leading to a decrease in overall system efficiency.

5.4.2 Varying the Number of Orders and Totes

The impact of the number of orders and totes on system efficiency is summarized in Appendix B, Table B3 and Table B4. As shown in Figure 3, system efficiency is more sensitive to the number of totes. More totes imply more SKUs to be picked, leading to longer average times required (ATRs) to complete all tasks. On the other hand, increasing the number of orders while keeping the number of totes basically unchanged does not significantly increase the number of SKUs to be picked, so ATRs are not as sensitive to the number of orders.

5.4.3 Varying the Number of Robots and Picking Stations

The impact of the number of robots and picking stations on system efficiency is summarized in Appendix B, Table B5 and Table B6. As shown in Figure 4, since the length of the circular conveyor in our system is only related to the width of the warehouse, changing the number of picking stations on the circular conveyor does not significantly affect the average times required (ATRs). On the contrary, the number of robots has a more obvious impact on ATRs. We can see that as the number of robots increases, ATRs decrease rapidly and eventually stabilize. This is evident because the more robots there are, the faster the totes can reach the conveyor for picking.

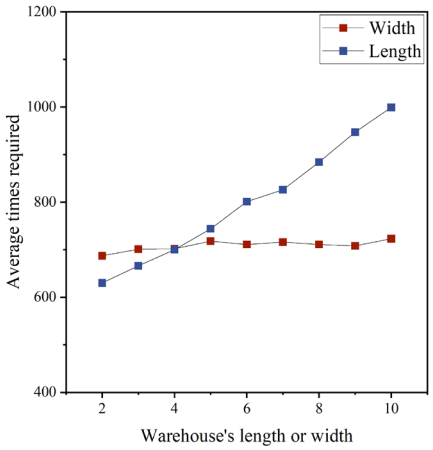

Figure 2. ATRs as the warehouse's length or width change

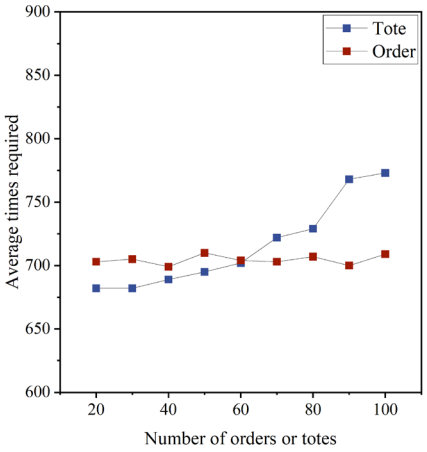

Figure 3. ATRs as the order or tote numbers change

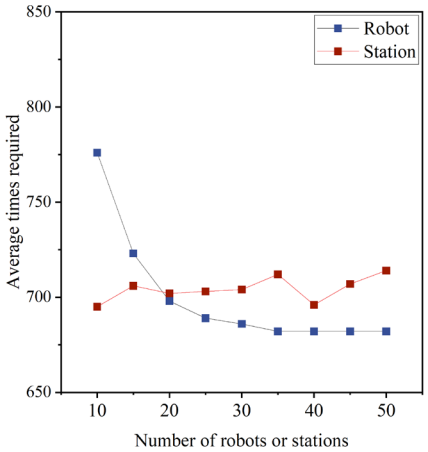

Figure 4. ATRs as the robot or station numbers change

7. Conclusion and Future Research

This paper addresses the optimization of order fulfillment processes in various robotic warehousing

systems by proposing a unified and standardized integrated decision-making optimization framework. Previously, no existing literature had introduced a unified and standardized methodology for the joint optimization of diverse warehousing systems.

To develop a unified, standardized, and general integrated framework, we thoroughly considered the characteristics of various robotic warehousing systems. We concluded that essential decision-making optimization in these systems revolves around three key elements: orders, retrieval units (such as totes or pods), and handling units (robots or equipment). The entire order fulfillment process in robotic warehousing systems can be broken down into a continuous cycle of three processes: the assignment and sequencing of orders, the assignment and sequencing of retrieval units, and the task assignment and scheduling of handling units. Based on this understanding, we propose an integrated decision-making model framework for automated and robotic warehousing systems that optimizes all decisions related to these three key elements, making the framework applicable to all automated and robotic warehousing systems.

Using the integrated decision-making model framework, we apply the HaiRobotics multi-tote system as a case study to develop both a three-stage model and an integrated model, illustrating joint optimization decisions in order fulfillment. We designed the MP-MD method to solve the integrated decision-making MIP model, which yields exact solutions for joint optimization problems. Additionally, we proposed the DeepRMHA capable of solving large-scale integrated models within a short time frame. The numerical experiment results demonstrate that our MP-MD method achieves the same optimal solutions as Gurobi while solving larger instances faster. Moreover, DeepRMHA significantly accelerates the solution process for the integrated model based on MP-MD, providing a high-quality approximate solution that far surpasses the results obtained by separately optimizing each decision in the system.

In the future, as new warehousing systems emerge, this integrated decision-making framework can be applied to a broader range of warehouse scenarios, enabling further optimization and refinement. Additionally, the MPMD and DeepRMHA methods introduced in this paper can be adapted to other complex combinatorial optimization problems across various fields.

References

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

Boysen, N., Briskorn, D., & Emde, S. (2017). Parts-to-picker based order processing in a rack-moving mobile robots environment. European Journal of Operational Research, 262(2), 550-562.

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

Bozer, Y. A., & White, J. A. (1984). Travel-time models for automated storage/retrieval systems. IIE transactions, 16(4), 329-338.

Bukchin, Y., & Raviv, T. (2022). A comprehensive toolbox for load retrieval in puzzle-based storage systems with simultaneous movements. Transportation Research Part B: Methodological, 166, 348-373.

Cals, B., Zhang, Y., Dijkman, R., Van Dorst, C., 2021. Solving the online batching problem using deep reinforcement learning. Computers & Industrial Engineering 156, 107221. https://doi.org/10.1016/j.cie.2021.107221

Fuentes Saenz, H., 2011. Data mining framework for batching orders in real-time warehouse operations (Master of Science). Iowa State University, Digital Repository, Ames. https://doi.org/10.31274/etd-180810-2712Gong, Y., Jin, M., & Yuan, Z. (2021). Robotic mobile fulfilment systems considering customer classes. International Journal of Production Research, 59(16), 5032-5049.

Gue, K. R., Furmans, K., Seibold, Z., & Uludağ, O. (2013). GridStore: a puzzle-based storage system with decentralized control. IEEE Transactions on Automation Science and Engineering, 11(2), 429-438.

Gue, K. R., & Kim, B. S. (2007). Puzzle-based storage systems. Naval Research Logistics (NRL), 54(5), 556-567.

Hausman, W. H., Schwarz, L. B., & Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. Management science, 22(6), 629-638.

He, J., Liu, X., Duan, Q., Chan, W. K. V., & Qi, M. (2023). Reinforcement learning for multi-item retrieval in the puzzle-based storage system. European Journal of Operational Research, 305(2), 820-837.

Ji, T., Zhang, K., & Dong, Y. (2020). Model-based Optimization of Pod Point Matching Decision in Robotic Mobile Fulfillment System. 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA),

Jiang, M., Leung, K.H., Lyu, Z., Huang, G.Q., 2020. Picking-replenishment synchronization for robotic forward-reserve warehouses. Transportation Research Part E: Logistics and Transportation Review 144, 102138. https://doi.org/10.1016/j.tre.2020.102138

Kang, Y., Qu, Z., Yang, P., 2024. Enhancing E-Commerce Warehouse Order Fulfillment Through Predictive Order Reservation Using Machine Learning. IEEE Trans. Automat. Sci. Eng. 1–14. https://doi.org/10.1109/TASE.2024.3428541

Kim, M., Park, J., Kim, J., 2021. Learning Collaborative Policies to Solve NP-hard Routing Problems.

Lamballais, T., Roy, D., & De Koster, M. (2017). Estimating performance in a robotic mobile fulfillment system. European Journal of Operational Research, 256(3), 976-990.

Lerher, T. (2016). Travel time model for double-deep shuttle-based storage and retrieval systems. International Journal of Production Research, 54(9), 2519-2540.

Li, Z.P., Zhang, J.L., Zhang, H.J., 2017. Optimal Selection of Movable Shelves under Cargo-to-Person Picking Mode. Int. j. simul. model. 16, 145–156. https://doi.org/10.2507/IJSIMM16(1)CO2

Matusiak, M., De Koster, R., Saarinen, J., 2017. Utilizing individual picker skills to improve order

batching in a warehouse. European Journal of Operational Research 263, 888–899. https://doi.org/10.1016/j.ejor.2017.05.002
Merschformann, M., Lamballais, T., De Koster, M., & Suhl, L. (2019). Decision rules for robotic mobile fulfillment systems. Operations Research Perspectives, 6, 100128.
Mirzaei, M., De Koster, R. B., & Zaerpour, N. (2017). Modelling load retrievals in puzzle-based storage systems. International Journal of Production Research, 55(21), 6423-6435.
Morabit, M., Guy Desaulniers, G., Lodi, A., 2021. Machine-learning-based column selection for column generation. Transportation Science. 55(4):815–831.
Müller, M., Ulrich, J. H., Reggelin, T., Lang, S., & Reyes-Rubiano, L. S. (2020). Comparison of deadlock handling strategies for different warehouse layouts with an AGVS. 2020 Winter Simulation Conference (WSC),
Pirayesh Neghab, D., Khayyati, S., Karaesmen, F., 2022. An integrated data-driven method using deep learning for a newsvendor problem with unobservable features. European Journal of Operational Research 302, 482–496. https://doi.org/10.1016/j.ejor.2021.12.047
Rohit, K. V., Taylor, G. D., & Gue, K. R. (2010). Retrieval time performance in puzzle-based storage systems. IIE Annual Conference. Proceedings,
Roy, D. (2016). Semi-open queuing networks: a review of stochastic models, solution methods and new research areas. International Journal of Production Research, 54(6), 1735-1752.
Roy, D., Nigam, S., de Koster, R., Adan, I., & Resing, J. (2019). Robot-storage zone assignment strategies in mobile fulfillment systems. Transportation Research Part E: Logistics and Transportation Review, 122, 119-142.
Schenone, M., Mangano, G., Grimaldi, S., & Cagliano, A. C. (2020). An approach for computing AS/R systems travel times in a class-based storage configuration. Production & Manufacturing Research, 8(1), 273-290.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. J. a. p. a. (2017). Proximal policy optimization algorithms.
Singbal, V., & Adil, G. K. (2021). Designing an automated storage/retrieval system with a single aisle-mobile crane under three new turnover based storage policies. International Journal of Computer Integrated Manufacturing, 34(2), 212-226.
Song, W., Chen, X., Li, Q., Cao, Z., 2023. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning. IEEE Trans. Ind. Inf. 19, 1600–1610. https://doi.org/10.1109/TII.2022.3189725
Valle, C.A., Beasley, J.E., 2021. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. Computers & Operations Research 125, 105090. https://doi.org/10.1016/j.cor.2020.105090
Van Der Gaast, J.P., Weidinger, F., 2022. A deep learning approach for the selection of an order picking system. European Journal of Operational Research 302, 530–543. https://doi.org/10.1016/j.ejor.2022.01.006
Wang, B., Yang, X., Qi, M. J. F. S., & Journal, M. (2023). Order and rack sequencing in a robotic mobile fulfillment system with multiple picking stations. 1-39.
Wang, R., Yang, P., Gong, Y., Chen, C., 2024. Operational policies and performance analysis for overhead robotic compact warehousing systems with bin reshuffling. International Journal of Production Research 62, 5236–5251. https://doi.org/10.1080/00207543.2023.2289643
Wang, R., Hua, Z., Liu, G., Zhang, J., Yan, J., Qi, F., Yang, S., Zhou, J., Yang, X., 2021. A Bi-Level

Framework for Learning to Solve Combinatorial Optimization on Graphs.
Wang, Z., Sheu, J., Teo, C., Xue, G., 2022. Robot Scheduling for Mobile-Rack Warehouses: Human–Robot Coordinated Order Picking Systems. Production and Operations Management 31, 98–116. https://doi.org/10.1111/poms.13406
Weidinger, F., Boysen, N., & Briskorn, D. (2018). Storage assignment with rack-moving mobile robots in KIVA warehouses. Transportation Science, 52(6), 1479-1495.
Wu, Y., Song, W., Cao, Z., Zhang, J., Lim, A., 2020. Learning Improvement Heuristics for Solving Routing Problems.
Xiang, X., Liu, C., Miao, L., 2018. Storage assignment and order batching problem in Kiva mobile fulfilment system. Engineering Optimization 50, 1941–1962.
Xie, L., Thieme, N., Krenzler, R., Li, H., 2021. Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. European Journal of Operational Research 288, 80–97. https://doi.org/10.1016/j.ejor.2020.05.032
Yalcin, A., Koberstein, A., & Schocke, K.-O. (2019). An optimal and a heuristic algorithm for the single-item retrieval problem in puzzle-based storage systems with multiple escorts. International Journal of Production Research, 57(1), 143-165.
Yang, P., Jin, G., & Duan, G. (2022). Modelling and analysis for multi-deep compact robotic mobile fulfilment system. International Journal of Production Research, 60(15), 4727-4742.
Yang, X., Hua, G., Hu, L., Cheng, T.C.E., Huang, A., 2021. Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system. Computers & Operations Research 135, 105467.
Yu, H., Yu, Y., & de Koster, R. (2022). Dense and fast: Achieving shortest unimpeded retrieval with a minimum number of empty cells in puzzle-based storage systems. IISE Transactions, 55(2), 156-171.
Yuan, Z., & Gong, Y. Y. (2017). Bot-in-time delivery for robotic mobile fulfillment systems. IEEE Transactions on Engineering Management, 64(1), 83-93.
Zhang, W., & Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. IJCAI,
Zhen, L., & Li, H. (2022). A literature review of smart warehouse operations management. Frontiers of Engineering Management, 9(1), 31-55.
Zhen, L., Tan, Z., De Koster, R., Wang, S., 2023. How to Deploy Robotic Mobile Fulfillment Systems. Transportation Science trsc.2022.0265. https://doi.org/10.1287/trsc.2022.0265
Zhuang, Y., Zhou, Y., Yuan, Y., Hu, X., & Hassini, E. (2022). Order picking optimization with rack-moving mobile robots and multiple workstations. European Journal of Operational Research, 300(2), 527-544.
Zou, B., Xu, X., & De Koster, R. (2018). Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. European Journal of Operational Research, 267(2), 733-753.
Zou, Y., & Qi, M. (2021). A heuristic method for load retrievals route programming in puzzle-based storage systems. arXiv preprint arXiv:2102.09274.