Joint Scheduling Solution of Order, Tote, and Robot in a Multi-tote Storage and Retrieval System with Sequential Picking Stations

Jie Shao[a,b], Mingzhe Li[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

The multi-tote storage and retrieval (MTSR) systems employ autonomous tote-handling mobile robots to enhance order fulfillment efficiency by storing and retrieving multiple totes simultaneously. To increase the utilization of retrieved totes, MTSR systems configure the sequential picking stations to fulfill orders with less tote-handling effort. Considering the tight coupling among order, tote, and robots in MTSR systems, order fulfillment will significantly benefit from the meticulous design of order, tote, and robot scheduling solutions. We first investigate the challenging joint order-tote-robot scheduling problem in an MTSR system with sequential picking stations. To capture the unique features of order fulfillment decisions, we formulate an integrated mixed integer programming model and introduce the hyper adaptive large neighborhood search algorithm—a tailored, end-to-end framework based on deep reinforcement learning—to solve it. Extensive numerical experiments demonstrate the efficiency of our model and the superior performance of our hyper adaptive large neighborhood search algorithm. Compared to a common-used multi-stage greedy decision-making approach, our method achieves a 59.23% reduction in order fulfillment makespan (maximum order completion time) for large-scale instances, completing tasks in just 65.65 seconds. Our findings indicate that the performance of MTSR systems is more dependent on the number of totes than the number of orders. Specifically, increasing the number of totes significantly extends the average task completion time, while increasing the number of orders has a negligible impact. Additionally, increasing the number of robots rapidly decreases and eventually stabilizes the maximum completion time, whereas varying the number of picking stations on

the circular conveyor has minimal effect on the maximum completion time.

Keywords: Robotic Warehouse System、Intralogistics Optimization、MTSR、Hyper-heuristic

1. Introduction

The multi-tote storage and retrieval (MTSR) system is an advanced automated parts-to-picker system designed for e-commerce, utilizing autonomous case-handling mobile robots (ACRs) to efficiently store and retrieve multiple totes, thereby reducing manual labor and operating costs (Figure 1). Widely adopted in various e-commerce warehouses, MTSR system are frequently equipped with discrete picking stations. However, discrete picking station configurations often suffer from lower tote reuse rates, slower order matching, increased scheduling complexity, congestion, and reduced overall outbound throughput due to dispersed tote handling. In contrast, MTSR systems with sequential picking stations (Figure 2) offer several advantages. Firstly, sequential picking stations significantly enhance tote reuse rates and facilitate quicker transfers by allowing multiple totes to be cached for reuse, expanding the order buffer pool, and enabling more orders to match with more totes, thereby improving overall picking efficiency. Secondly, sequential picking stations reduce the complexity of robot scheduling and alleviate congestion in the picking area, ensuring a safer work environment for human pickers. Lastly, sequential picking stations focus robots on retrieving totes rather than shuttling between stations, thereby increasing the overall throughput and efficiency of the outbound process.

Figure 1. MTSR Systems.

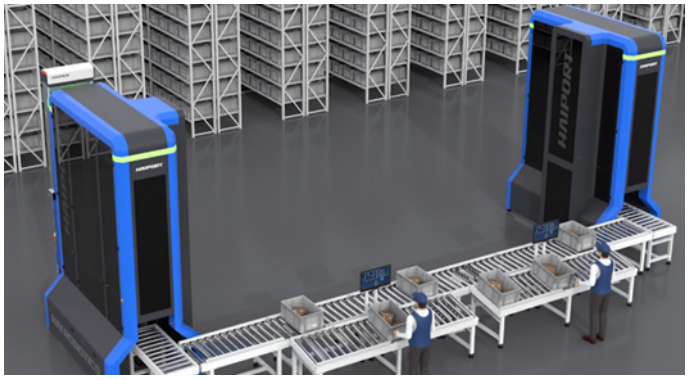


Figure 2. Sorting Conveyor with multiple picking stations.

Despite the extensive implementation of MTSR systems in industries such as e-commerce, retail, food, pharmaceuticals, manufacturing, and parts supply, most still utilize discrete picking stations. This underscores the urgent need for designs featuring sequential picking stations. However, there is currently minimal academic research on MTSR systems with sequential picking stations. Therefore, this paper aims to optimize order fulfillment in MTSR systems with sequential picking stations.

The MTSR system with sequential picking stations offers significant advantages, as previously mentioned. However, it also introduces substantial challenges in storage and retrieval complexity. Configuring the system with multiple serial picking stations and conveyor belts complicates the effective

flow of totes throughout the entire operation. The order fulfillment process in an MTSR system with sequential picking stations proceeds as follows: Orders are received and assigned to slots on the sorting wall at picking stations. Totes containing required items are identified, and their arrival sequence is determined, generating tote handling tasks. These tasks are allocated to ACRs, with routes efficiently planned to transport multiple totes to picking stations in a single tour. Pickers then select items from totes to fill order boxes. Once all items for an order are fulfilled, the order box leaves the station, allowing new orders to enter. This process repeats until all orders are completed. Notably, the sequential picking station configuration involves a detailed examination of the flow of totes, including the scheduling of totes on the conveyor belts, the rhythm control between the robot's picking and delivering operations, and the timing of tote delivery at both the conveyor's entry and exit points. Consequently, decision-making and coordination among the three key components—orders, totes, and robots—toward a unified optimization objective are complex. Given the unique challenges of the MTSR system with sequential picking stations, our research emphasizes joint optimization within this system.

The research questions in this study are:

(1) **Order assignment and sequencing:** How should orders be assigned to slots on the sorting wall across different picking stations, and what sequencing strategy should be utilized for their entry into these picking stations to enhance overall order fulfillment efficiency?

(2) **Tote assignment and sequencing:** What methods can determine the sequence of tote arrivals at various picking stations, considering the order requirements on the sorting wall?

(3) **Robot scheduling:** What is the optimal scheduling plan for organizing ACRs in tote transportation tasks? Specifically, how can task allocation and path planning for ACRs retrieving multiple totes be optimized to minimize the maximum completion time for transporting all necessary totes?

The tote flow involves the joint optimization of several factors, including assigning incoming orders to slots on the sorting wall of picking stations and sequencing their execution, routing totes to picking stations (route control on conveyors) and determining the sequence of outbound operations, and task allocation and path planning for multiple ACRs. This challenge encompasses the entire order fulfillment process in the MTSR system with sorting conveyor. We adopt a novel approach to investigate the joint optimization problem of orders, totes, and robots. Specifically, we introduce an integrated mixed integer programming (MIP) model to depict the complete flow of totes and the joint optimization

problem within the system.

The main contributions of this paper are as follows:

(1) **First investigation**: We present the first investigation into the joint optimization problem of orders, totes, and robots in the MTSR system with sequential picking stations, representing the inaugural utilization of optimization techniques in researching integrated storage and retrieval systems with sorting conveyor systems. We introduce an MIP model to depict the complete flow of totes and the joint optimization problem within the system.

(2) **Novel algorithmic framework:** To address this MIP model, we devise a tailored end-to-end algorithmic framework based on deep reinforcement learning (DRL). Extensive numerical experiments validate its capability to deliver high-quality solutions.

(3) **Managerial insights**: We conduct a sensitivity analysis by varying the length and width of the warehouse, as well as the number of orders, totes, robots, and picking stations. This analysis offers valuable managerial insights into the optimal deployment of robots and workstations within the MTSR systems with sorting conveyor.

The remainder of this paper is organized as follows: Section 2 reviews and summarizes the related research. Section 3 describes and formulates the problem. In Section 4, we propose an end-to-end algorithmic framework based on ALNS to solve our MIP model. Section 5 presents the numerical experiments and discusses the results. Section 6 concludes with a discussion of the findings and directions for future research.

2. Literature Review

We categorize the existing literature related to our topic into three themes: robotic mobile fulfillment systems (RMFS), tote-handling robotic warehousing systems, and AI technologies in warehousing systems. For further information on robotic warehousing systems, interested readers can refer to the articles by Azadeh et al. (2019), Boysen et al. (2019), and Zhen and Li (2022).

2.1 Robotic Mobile Fulfillment Systems

RMFS, a widely-used goods-to-person system, involves robots transporting pods to workstations for order fulfillment. Research on RMFS primarily focuses on system analysis and performance evaluation, robot scheduling and route planning, as well as task assignment and scheduling. Queuing

network models have been used to analyze the system performance, with Roy (2016) being the first to apply this approach to RMFS. Other studies have analyzed the performance of RMFS concerning factors such as partitioned or unpartitioned cases (Lamballais et al., 2017), dedicated or shared robots (Yuan & Gong, 2017), battery management (Zou et al., 2018), picking and replenishment processes (Roy et al., 2019), customer sorting (Gong et al., 2021) and arrangement of picking stations (Yang et al., 2022). Operational optimization and control of RMFS include shelf scheduling, robot scheduling, and path planning. Weidinger et al. (2018) and Ji et al. (2020) have explored storage space allocation, while Boysen et al. (2017) and Zhuang et al. (2022) have proposed various strategies for shelf sorting and scheduling. Path planning and traffic control for robots have also been studied, with Merschformann et al. (2019) and Müller et al. (2020) proposing different path planning algorithms and strategies for dealing with collisions, deadlocks, and congestion.

2.2 Tote-handling Robotic Warehousing Systems

In tote-handling robotic warehousing systems, robots retrieve totes from racks, transport them to workstations, wait for pickers to collect the required items, and then return the totes to the racks. Given its relevance to our topic, we have focused our review on three types of robotic storage systems: autonomous vehicle storage and retrieval system (AVS/RS), puzzle-based storage (PBS) system and MTSR system.

The AVS/RS is a storage and retrieval system that employs shuttles to handle incoming and outgoing tasks, including the shuttle-based storage and retrieval system (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 various performance metrics. Since then, queuing theory has been applied to evaluate the performance of AVS/RS (Bozer & White, 1984). Researchers such as Lerher (2016), Schenone et al. (2020), Singbal and Adil (2021) have extensively studied queuing network models or time travel models to evaluate the performance of AVS/RS. To optimize the system's operational efficiency and effectiveness, research has focused on developing methods to enhance its performance. For instance, Borovinšek et al. (2017) proposed a multi-objective optimization solution framework to design an eco-friendly warehouse and obtained a solution using genetic algorithms. Similarly, Yetkin Ekren (2021) presented a hierarchical solution that considers minimizing the average turnaround time and the average energy consumption of the system.

The 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," that can be used for temporary parking. Gue and Kim (2007) were the first to evaluate the expected item retrieval time for a single fixed-position escort. Since then, researchers have extended 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) proposed a general integer programming model, while other authors such as Yalcin et al. (2019), Ma et al. (2022), Yu et al. (2022), Bukchin and Raviv (2022) have proposed various heuristic or exact algorithms to solve the problem. For the multi-item retrieval problem, Gue et al. (2013) discussed multi-item retrieval based on GridStore technology, while Mirzaei et al. (2017) proposed a closed expression for retrieving two items. Zou and Qi (2021) proposed a heuristic algorithm to handle the multi-item retrieval problem with multiple random location escorts and I/O points, and He et al. (2023) designed a DRL method to solve the multi-item retrieval problem.

The MTSR system falls under the category of tote-handling robotic warehousing systems. Qin et al. (2024) evaluated the performance of MTSR systems by developing optimal and heuristic routing algorithms. They also used a semi-open queuing network model to estimate travel times, throughput capacity, and system costs under various operational conditions in multi-block warehouse environments. However, few studies have focused on the operational optimization of MTSR systems.

2.3 AI Technologies in Warehousing Systems

Deep learning and reinforcement learning are effective for high-dimensional feature extraction and decision-making in complex environments (LeCun et al., 2015). DRL has been applied in various fields, including robotics, recommender systems, and routing (Li, 2017), and has a history in combinatorial optimization (Zhang & Dietterich, 1995; Sutton & Barto, 2018). Successful DRL applications include solving the TSP (Vinyals et al., 2015; Bello et al., 2016) and extending to the VRP (Nazari et al., 2018). Graph neural networks (GNNs) are also utilized in combinatorial optimization problems (COPs) modeled as graphs, with notable methods including GCNs for TSP (Joshi et al., 2019), adaptive GCNs for VRP (Duan et al., 2020), and GPNs for large-scale TSP (Ma et al., 2019). Methods for improving feasible solutions involve reinforcement learning and heuristics, such as learning TSP heuristics (Deudon et al., 2018) and iterative VRP optimization (Lu et al., 2020). Generic frameworks have further

advanced COP development, with efforts to define generic MDPs for graph problems (Khalil et al., 2017) and flexible GNN-based architectures (Drori et al., 2020).

In recent years, with the rise of e-commerce warehousing systems, data-driven and machine learning research methods have been increasingly applied to the field of warehousing(Saenz et al, 2011; Matusiak et al, 2017). Cals et al. (2021) employed the DRL approach to tackle the online order batching problem (OOBP) using the Proximal Policy Optimization (PPO) algorithm. Pirayesh Neghab et al. (2022) utilized a multi-layered neural network to optimize inventory management by dynamically estimating demand factors and integrating historical data with real-time updates to reduce costs. Van Der Gaast et al. (2022) applied deep neural networks (DNNs) to order picking (OP) problem to predict the optimal order picking system and its configuration based on various order structures, enabling quick and objective decision-making without exhaustive simulations. Wang et al. (2024) used reinforcement learning to optimize the reshuffling process in a overhead robotic compact storage and retrieval system, significantly reducing reshuffling distances and improving efficiency compared to traditional methods. Kang et al. (2024) examined the online order batching problem with reservation mechanism (OOBPRM) for the first time, employing ensemble learning methods to predict similarities between current and future orders. This enables the reservation of suitable orders for later batching, optimizing the overall order batching process and improving order turnover time and efficiency in e-commerce warehouses.

Despite the extensive research on robotic warehouse systems, to the best of our knowledge, there has been no prior investigation specifically targeting the MTSR system with sequential picking stations. Moreover, as indicated in Table 1, there is limited literature addressing the operational optimization of systems from a holistic perspective, encompassing the entire process flow, in publications related to RMFS and tote-handling robotic warehousing system. In particular, no previous studies have utilized optimization methods to examine the flow of totes in integrated systems. Additionally, despite significant advancements in applying AI technologies in warehousing systems, as depicted in Table 2, there remains a lack of research employing deep learning and reinforcement learning for end-to-end solutions in the joint optimization of multiple decisions within the warehouse. Hence, our study endeavors to bridge these research gaps.

Table 1. Overview of the robotic warehouse system.

Article	Research areas	Methodology	Research content
Roy (2016) (Lamballais et al., 2017)	RMFS	Analytic	Layout design

(Yuan & Gong, 2017) (Zou et al., 2018) (Roy et al., 2019) (Gong et al., 2021) Weidinger et al. (2018) Ji et al. (2020) Boysen et al. (2017) Zhuang et al. (2022)		Optimization	Pod scheduling
Merschformann et al. (2019) Müller et al. (2020)		Simulation	Robot dispatching
Bozer and White (1984) Lerher (2016) Schenone et al. (2020) Singbal and Adil (2021)	Tote-handling System	Analytic	Layout design
Qin et al. (2024)			Multi-item retrieval
Borovinšek et al. (2017) Yetkin Ekren (2021)		Optimization	Energy consumption Depot efficiency
Gue and Kim (2007)		Heuristic	Single-item retrieval
Rohit et al. (2010)		IP	
Yalcin et al. (2019)		Heuristic	
MA et al. (2022)		Heuristic	
Yu et al. (2022)		Heuristic	
Bukchin and Raviv (2022)		DP	
Gue et al. (2013)		Heuristic	Multi-item retrieval
Mirzaei et al. (2017)		Heuristic	
Zou and Qi (2021)		Heuristic	
He et al. (2023)		IP & RL	

Table 2. Overview of the AI technologies in warehousing systems.

Article	Research areas	Methodology	Research content
Fuentes Saenz (2011)	Data-driven & Machine Learning	Decision Tree	OBP
Matusiak et al. (2017)		Multiple Linear Regression	OP
Cal er al. (2021)		PPO	OOBP
Pirayesh Neghab et al. (2022)		HMM&DNN	Inventory
Van Der Gaast et al. (2022)		DNN	OP
Wang et al. (2024)		RL	Analysis
Kang et al. (2024)		Ensemble Learning	OOBPRM

3. Problem Description and Formulation

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.

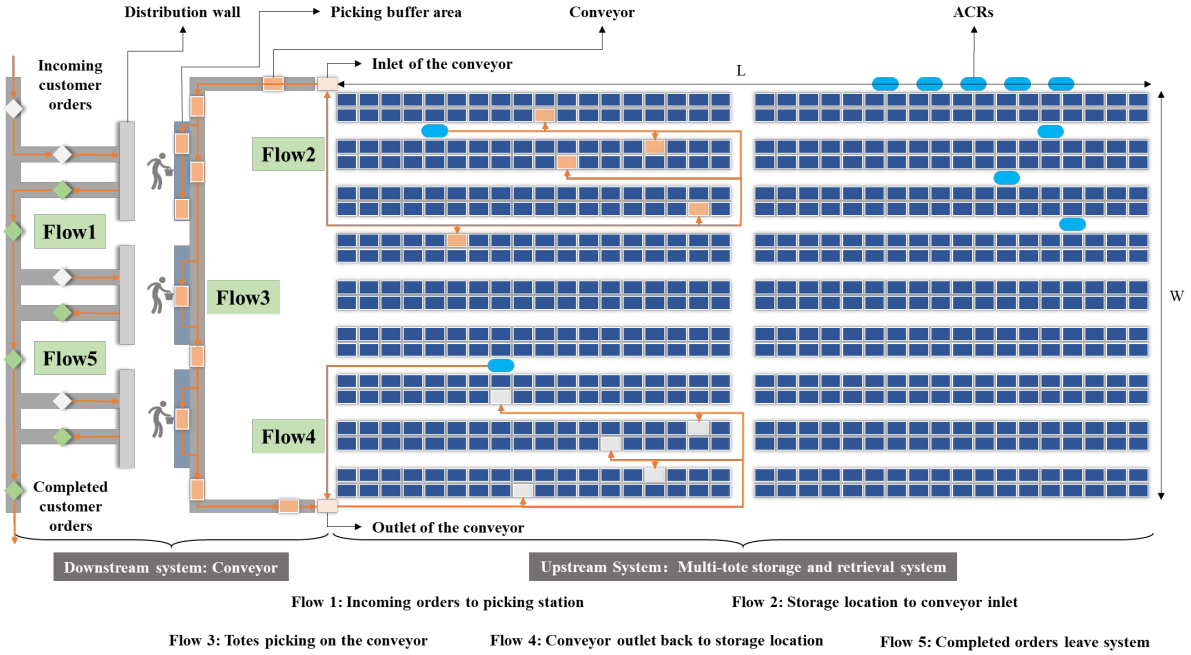


Figure 3. MTSR system with sorting conveyor and tote flow process.

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 Mixed Integer Programming Formulation

In this section, we present a MIP model that covers multiple orders, multiple totes, multiple picking stations, multiple ACRs, and different warehouse layouts. These variables can be assigned arbitrary feasible values. To enhance understanding of our model, we first explain the sets, parameters, and decision variables, as shown in Table 3.

Table 3. Symbols used to describe and formulate the problems.

Sets	
O	The set of orders.
P	The set of picking stations.
K	The set of ACRs.
P_1	The set of all totes' first pick-up points, also known as the set of all totes.
P_2	The set of all totes' second pick-up points, also known as the conveyor outlet.
D_1	The set of all totes' first delivery points, also known as the conveyor inlet.
D_2	The set of all totes' second delivery points, also known as the storage locations for all totes.
W	The set of starting points for all ACRs.
N	The set of all points $P_1 \cup P_2 \cup D_1 \cup D_2 \cup W$, includes the two pick-up points, the two delivery points, and the starting points for the ACRs.
Parameters	
q_i	The load capacity of point i is 1, as we consider each tote to have a weight of 1 unit.
Q	The maximum load capacity of ACRs is 8, as each ACR can carry a maximum of 8 totes.
n	The total number of totes awaiting picking.
t_{ij}	The travel time required for ACRs to travel from point i to point j.
s_i	The service time required at point i.
e_i	The earliest departure time for tote i.
l_i	The latest departure time for tote i.
p_i	The time for picking station staff to pick tote i.
$OT_{i,o}$	The order and tote pairs, if tote i belongs to order o, then $OT_{i,o} = 1$, otherwise $OT_{i,o} = 0$.

t_p^1	The time required from the conveyor inlet to picking station p.
t_p^2	The time required from picking station p to the conveyor outlet.
v	The speed at which totes move on the conveyor.
b_1	The maximum limit on the number of orders cached at the picking station.
b_2	The maximum limit on the number of totes cached at the picking station.
M	A sufficiently large positive number.
Variables	
T	The time when all tasks are completed.
T_o^s	The start time of executing order o.
T_o^e	The end time of executing order o.
T_i	The time when ACRs arrive at point i.
$T_{i,p}^a$	The time when tote i arrives at picking station p.
$T_{i,p}^s$	The time when tote i starts picking at picking station p.
$T_{i,p}^e$	The time when tote i finishes picking at picking station p.
x_{ij}	If a robot traveling from point i to j, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$.
$y_{i,p}$	If tote i is assigned to the picking station p, then $y_{i,p} = 1$; otherwise, $y_{i,p} = 0$.
$z_{o,p}$	If order o is assigned to the picking station p, then $z_{o,p} = 1$; otherwise, $z_{o,p} = 0$.
α_{o_1,o_2}	If order o_1 has started execution when o_2 begins execution, then $\alpha_{o_1,o_2} = 1$; otherwise, $\alpha_{o_1,o_2} = 0$.
β_{o_1,o_2}	If order o_1 starts execution when o_2 has not yet completed execution, then $\beta_{o_1,o_2} = 1$; otherwise, $\beta_{o_1,o_2} = 0$.
γ_{o_1,o_2}	If order o_1 starts execution when o_2 is currently executing, if so, then $\gamma_{o_1,o_2} = 1$; otherwise, $\gamma_{o_1,o_2} = 0$.
δ_{o_1,o_2}	If order o_1 starts execution when o_2 is executing at the same picking station, if so, then $\delta_{o_1,o_2} = 1$; otherwise, $\delta_{o_1,o_2} = 0$.
u_i^k	If robot k passes through point i, then $u_i^k = 1$, otherwise $u_i^k = 0$.
Q_i	The load of the robot at point i.
I_i	The time when the tote i arrives at the conveyor inlet.
$f_{i,j,p}$	If tote i arrives at picking station p before tote j, then $f_{i,j,p} = 1$; otherwise, $f_{i,j,p} = 0$.

We have divided all decision-making aspects into three distinct parts: the first part involves assigning multiple orders to multiple picking stations and determining the sequence of order execution at each picking station; the second part entails assigning multiple totes to multiple picking stations and establishing the sequence in which totes arrive at the conveyor or picking stations; the third part encompasses assigning multiple tote tasks to multiple ACRs and scheduling their routes to complete all tasks. Time constraints are used to integrate these different parts. In summary, the comprehensive tote flow problem in MTSR can be formulated as follows.

$$\min T \tag{3-1}$$

$s.t.$

Part 1.

$$\sum_{p=1}^{P} z_{o,p} = 1, \forall o \in O \tag{3-2}$$

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

$$T_{o_2}^e - T_{o_1}^s \le \beta_{o_1,o_2} * M, \forall o_1, o_2 \in O \quad (3-4)$$

$$\gamma_{o_1,o_2} \ge \alpha_{o_1,o_2} + \beta_{o_1,o_2} - 1, \forall o_1, o_2 \in O \quad (3-5)$$

$$\delta_{o_1,o_2} \ge \gamma_{o_1,o_2} + z_{o_1,p} + z_{o_2,p} - 2, \forall o_1, o_2 \in O, \forall p \in P \quad (3-6)$$

$$\sum_{o_2}^{O} \delta_{o_1,o_2} \le b_1, \forall o_1 \in O \quad (3-7)$$

$$T_o^s - T_{i,p}^s \le \left(3 - OT_{i,o} - y_{i,p} - z_{o,p}\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (3-8)$$

$$T_o^e - T_{i,p}^e \ge \left(OT_{i,o} + y_{i,p} + z_{o,p} - 3\right) * M, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (3-9)$$

Part 2.

$$\sum_{i=1}^{P_1} \sum_{p=1}^{P} y_{i,p} = \sum_{i=1}^{P_1} \sum_{o=1}^{O} OT_{i,o}, \forall o \in O \quad (3-10)$$

$$OT_{i,o} * z_{o,p} \le y_{i,p}, \forall i \in P_1, \forall o \in O, \forall p \in P \quad (3-11)$$

$$I_i \ge T_{i+2n}, \forall i \in P_1 \quad (3-12)$$

$$T_{i,1}^a = I_i + t_1^1, \forall i \in P_1 \quad (3-13)$$

$$T_{i,p}^s \ge T_{i,p}^a, \forall i \in P_1, \forall p \in P \quad (3-14)$$

$$T_{i,p}^e \ge T_{i,p}^s, \forall i \in P_1, \forall p \in P \quad (3-15)$$

$$T_{i,p}^a = T_{i,p-1}^e + t_p^1 - t_{p-1}^1, \forall i \in P_1, \forall p \in 2, \dots, P \quad (3-16)$$

$$T_{i,p}^e - T_{i,p}^s = p_i * y_{i,p}, \forall i \in P_1, \forall p \in P \quad (3-17)$$

$$T_{i,p}^a - T_{j,p}^a \le \left(1 - f_{i,j,p}\right) * M, \forall i, j \in P_1, \forall p \in P \quad (3-18)$$

$$T_{i,p}^a - T_{j,p}^a \ge -f_{i,j,p} * M, \forall i, j \in P_1, \forall p \in P \quad (3-19)$$

$$T_{j,p}^s - T_{i,p}^e \ge \left(f_{i,j,p} + y_{i,p} + y_{j,p} - 3\right) * M, \forall i, j \in P_1, \forall p \in P \quad (3-20)$$

$$T_{i,p}^s - T_{i,p}^a \le (b_2 - 1) * p_i, \forall i \in P_1, \forall p \in P \quad (3-21)$$

Part 3.

$$T \ge T_i, \forall i \in N \quad (3-22)$$

$$\sum_{j=1, j\neq i}^{N} x_{ij} = \sum_{j=1, j\neq i}^{N} x_{ji}, \forall i \in N \quad (3-23)$$

$$\sum_{k=1}^{K} \sum_{j=1, j\neq i}^{N} x_{ij} \ge 1, \forall i \in P_1 \cup P_2 \cup D_1 \cup D_2 \quad (3-24)$$

$$u_i^k - u_j^k \ge M(x_{ij} - 1), \forall i \in N, \forall j \in N, \forall k \in K \quad (3-25)$$

$$u_i^k - u_j^k \le M\left(1 - x_{ij}\right), \forall i \in N, \forall j \in N, \forall k \in K \quad (3-26)$$

$$u_i^k = 1, \forall i \in W, i = W_k, \forall k \in K \quad (3-27)$$

$$\sum_{k=1}^{K} u_i^k = 1, \forall i \in N \quad (3-28)$$

$$u_i^k = u_{i+2n}^k, \forall i \in P_1 \cup P_2, \forall k \in K \quad (3-29)$$

$$\sum_{j=1, j\neq i}^{N} x_{ij} \leq 1, \forall i \in W, i = W_k \quad (3-30)$$

$$Q_j \geq Q_i + q_i - Q(1 - x_{ij}), \forall i \in N \ \ i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \quad (3-31)$$

$$0 \leq Q_i \leq Q, \forall i \in N \quad (3-32)$$

$$T_j \geq T_i + t_{ij} + s_i - M(1 - x_{ij}), \forall i \in N \ \ i \neq j, \forall j \in P_1 \cup P_2 \cup D_1 \cup D_2 \quad (3-33)$$

$$T_{i+2n} \geq T_i + t_{i,i+2n} + s_i, \forall i \in P_1 \cup P_2 \quad (3-34)$$

$$e_i \leq T_i \leq l_i, \forall i \in N \quad (3-35)$$

$$T_{i,P}^e + t_P^2 \leq T_{i+n}, \forall i \in P_1 \quad (3-36)$$

$$T \geq 0 \quad (3-37)$$

$$T_i \geq 0, \forall i \in N \quad (3-38)$$

$$T_o^s, T_o^e \geq 0, \forall o \in O \quad (3-39)$$

$$T_{i,p}^a, T_{i,p}^s, T_{i,p}^e \geq 0, \forall i \in P_1, \forall p \in P \quad (3-40)$$

$$Q_i \geq 0, \forall i \in N \quad (3-41)$$

$$I_i \geq 0, \forall i \in P_1 \quad (3-42)$$

$$x_{ij} \in \{0,1\}, \forall i, j \in N \quad (3-43)$$

$$y_{i,p} \in \{0,1\}, \forall i \in P_1, \forall p \in P \quad (3-44)$$

$$z_{o,p} \in \{0,1\}, \forall o \in O, p \in P \quad (3-45)$$

$$\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2} \in \{0,1\}, \forall o_1, o_2 \in O \quad (3-46)$$

$$u_i^k \in \{0,1\}, \forall i \in N, \forall k \in K \quad (3-47)$$

$$f_{i,j,p} \in \{0,1\}, \forall i, j \in P_1, \forall p \in P \quad (3-48)$$

We have decided to adopt minimizing the maximum completion time as the objective function, as depicted in objective function (3-1). The constraints are analyzed from three aspects: assignment of order tasks to picking station slots and order execution sequence, assignment of totes to picking stations and tote execution sequence, and assignment and scheduling of tasks for ACRs.

Let's refine the description of the constraints regarding order assignment and order sequencing: Constraint (3-2) ensures that all orders must be completed. Constraints (3-3)-(3-6) properly constrain the decision variables $\alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$. Constraint (3-7) sets the upper limit on the number of orders executed simultaneously at each picking station. Constraints (3-8) and (3-9) restrict the start and end times for order execution. These constraints collectively ensure the proper assignment and sequencing of orders at the picking stations.

Next, let's discuss the constraints regarding the assignment of totes to picking stations and the sequence of tote execution: Constraint (3-10) ensures that all totes must undergo picking. Constraint (3-11) stipulates that a tote i can only be assigned to picking station p if tote i belongs to order o and order o has been assigned to picking station p. Constraint (3-12) specifies that the time when a tote arrives at the conveyor inlet must be greater than or equal to the time it arrives at D_1. Constraint (3-13) determines the time when a tote arrives at the first picking station. Constraints (3-14) and (3-15) dictate that the start picking time of a tote at a picking station must be greater than the time it arrives at the picking station, while the end picking time of a tote at a picking station must be greater than or equal to the time it arrives at the picking station. Constraint (3-16) ensures that the time when a tote arrives at the next picking station is equal to the end picking time of the tote at the previous picking station plus the travel time between the two picking stations. Constraint (3-17) states that if a tote is assigned to a picking station, its end picking time equals its start picking time plus the picking time of the tote; otherwise, its end picking time equals its start picking time. Constraints (3-18) and (3-19) dictate that if tote i arrives at picking station p before tote j, then $f_{i,j,p} = 1$; otherwise, $f_{i,j,p} = 0$. Constraint (3-20) ensures that if tote i arrives at picking station p before tote j, and both totes are assigned to picking station p, the start picking time of tote j must be greater than or equal to the end picking time of tote i. Constraint (3-21) specifies that the number of totes cached at each picking station cannot exceed the maximum limit.

Finally, let's discuss the constraints regarding task assignment and scheduling for ACRs: Constraint (3-22) ensures that the time of completion for all task points is greater than or equal to the time of completion for each individual task point. Constraint (3-23) enforces flow balance for ACRs during the process of picking up and delivering totes. Constraint (3-24) states that ACRs must complete all picking up or delivering tasks. Constraints (3-25)-(3-29) dictate that the same ACR can only be assigned to complete the same task. Constraint (3-30) specifies that each ACR can depart from its starting point at most once. Constraints (3-31)-(3-32) enforce the weight capacity constraints for ACRs. Constraint (3-33) ensures that the time when an ACR arrives at the next target point is greater than or equal to 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 (3-34) ensures that the time when a tote arrives at the destination point is greater than or equal to the time it arrives at the starting point plus the travel time between the starting and destination points and the service time at the starting point. Constraint (3-35) ensures that the completion

time for totes must satisfy the time window constraints. Constraint (3-36) specifies that the time of arrival at point P_2 must be greater than or equal to the time when the tote arrives at the conveyor outlet. Constraints (3-37)-(3-48) define the variables.

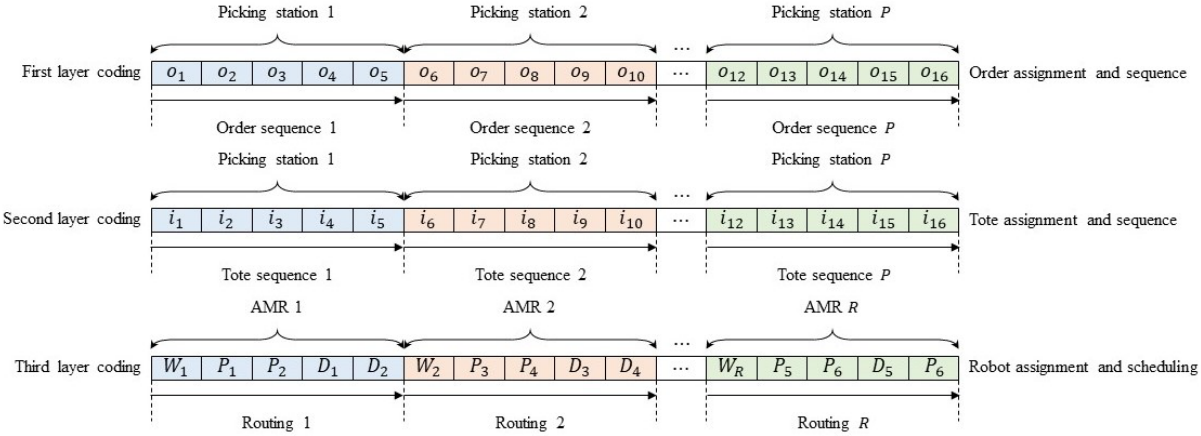


Figure 4. Coding of the three parts.

4. End-to-end Solution Methods

This section elaborates on a tailored metaheuristic framework designed to address multi-decision joint optimization problems, developed based on the large-scale Adaptive Large Neighborhood Search (ALNS). As we know, ALNS mainly comprises the construction of initial solutions, the design of disruption and repair operators, dynamic adjustment of operator selection and weights, acceptance criteria for new solutions, and algorithm termination strategies. It is an efficient framework for solving combinatorial optimization problems. However, joint optimization of multiple decisions presents significant challenges, particularly in designing efficient disruption and repair operators and dynamically adjusting operator weights. For instance, in the joint optimization problem of orders, totes, and robots that we investigate, even after careful simplification, three parts of encoding, as shown in Figure 4, are still required to fully represent all decisions. Adjusting operator weights and selecting operators based on the three-part encoding is a time-consuming and laborious task. Therefore, we adopt DRL to handle this aspect, as depicted in the basic framework of H-ALNS in Figure 5.

Initially, considering the complexity of the solutions, we introduced three distinct initial solution generation methods tailored for the three-part encoding to enhance solution efficiency. Subsequently, we provided a detailed exposition of operator design. Following that, we elucidated how DRL replaces operator weight adjustment and operator selection. Finally, we presented the conditions for accepting new solutions and terminating the algorithm. The pseudocode for ALNS is shown in Algorithm 1.

Algorithm 1: Hyper adaptive large neighborhood search.

Input:	Current solution S_i, maximum number of iterations n^{max}, destructor operators. $\{d_1, d_2, \cdots, d_m\}$, repair operators $\{r_1, r_2, \cdots, r_n\}$.
Output:	Best solution S^*
1	Initialization, set $iter = 1$.
2	Generate initial solution S_a using 3 initial solution algorithm.
3	Set the current solution $S_i = S_a$, and the optimal solution $S^* = S_a$.
4	Set $S = S_a$.
5	Select a destructor and a repair operator pair $d_i + r_i$ by DRL.
6	Getting a new solution S using $d_i + r_i$.
7	$if\ \ cost(S) < cost(S_i)$:
8	$S_i = S$.
9	$end\ \ if$.
10	$if\ \ cost(S) < cost(S^*)$:
11	$S^* = S$.
12	$end\ \ if$.
13	$if\ \ accept(S_i, S)$:
14	$S_i = S$.
15	$iter = iter + 1$.
16	$if\ \ iter > n^{max}$:
17	return S^*.
18	$end\ \ if$.
19	Go to 5.
Output:	S^*.

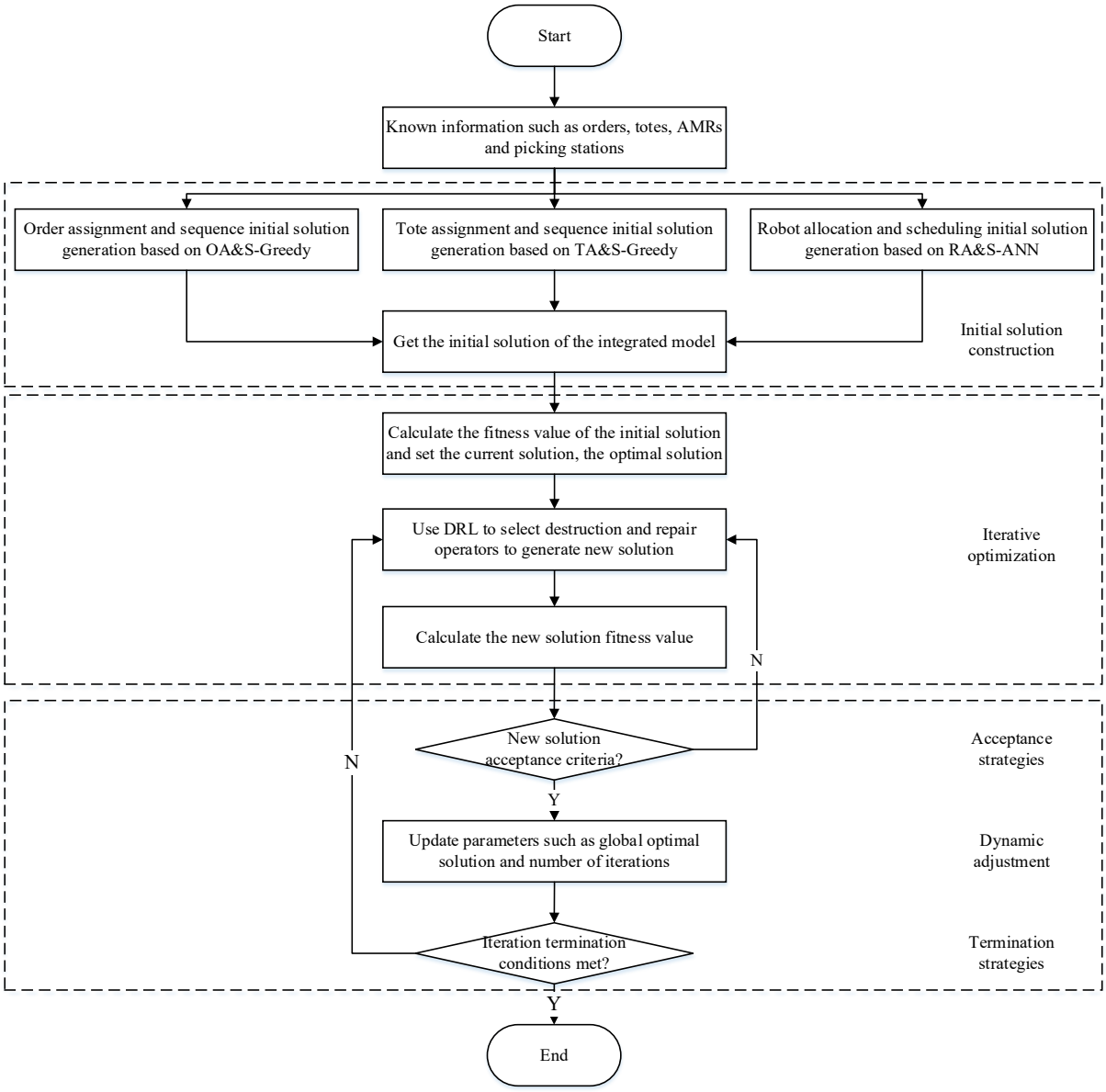


Figure 5. A framework for H-ALNS algorithm.

4.1 Initial Solution

4.1.1 Initial Solution of the Order Assignment and Sequence

The assignment and sorting of orders serve two main purposes. Firstly, to allocate orders with similar structures to the same picking station, and secondly, to ensure that orders before and after in execution sequence at the same picking station are similar to maximize the fulfillment of orders with the least number of totes. Therefore, the initial solution construction for order assignment and sorting involves two parts. The first part evaluates the similarity of orders using Equation (4-1) and categorizes them into several classes for assignment to picking stations. The second part employs a greedy algorithm

proposed by Wang et al. (2022) to sequence the orders assigned to each picking station. The complete algorithm pseudocode is presented in Algorithm 2.

$$\psi_{i_1,i_2} = \frac{O_{i_1}^T O_{i_2}}{O_{i_1}^T O_{i_1} + O_{i_2}^T O_{i_2} - O_{i_1}^T O_{i_2}} \quad (4-1)$$

We define the similarity of two orders as the proportion of the number of common SKUs to the total number of SKUs contained in both 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 contains SKU k. Therefore, $O_{i_1} = \left(O_{set_{i1}}, O_{set_{i2}}, \dots, O_{set_{iK}}\right)^T$.

Algorithm 2: 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: $z_{o,p}, \alpha_{o_1,o_2}, \beta_{o_1,o_2}, \gamma_{o_1,o_2}, \delta_{o_1,o_2}$.

4.1.2 Initial Solution of the Tote Assignment and Sequence

The purpose of assigning and sorting totes is primarily to determine which picking stations the totes need to go to and the sequence in which totes are to be retrieved from storage. After determining the tote assignments based on the assignment of orders, we prioritize the retrieval of each tote from storage based on its current urgency. The current urgency of a tote is defined as the frequency with which all orders currently assigned to the picking station slot are associated with it, as shown in Equation (4-2). The higher the number of orders requiring the tote, the greater its urgency for retrieval, and thus it should be retrieved first. The complete algorithm pseudocode is presented in Algorithm 3.

$$\omega_i = \sum_{p=1}^{P} OT_{i,o} \cdot z_{o,p}, \forall o \in O_{on} \quad (4-2)$$

Where O_{on} represents the orders currently occupying the slots at each picking station.

Algorithm 3: 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 totedings per tote: 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: $y_{i,p}, I_i, f_{i,j,p}$.

4.1.3 Initial solution of the Robot Allocation and Scheduling

The purpose of robot task assignment and scheduling is to determine which robot executes which tote tasks and the specific routing of these tasks by the robots. Due to its similarity to path problems, we adopt the Adapted Nearest Neighbor (ANN) method (Goutham & Stockar, 2023) for initial solution construction. ANN is a fast and simple greedy selection strategy that performs well in path-related problems. Starting from a defined or randomly specified initial node, ANN assigns the nearest unvisited node as the next node for the robot until all nodes are included in the path. For the problem proposed in this paper, we only need to ensure that each node insertion satisfies the priority and ACRs' capacity constraints, ensuring that the constructed initial solution is always feasible. The complete pseudocode of the algorithm is shown in Algorithm 4.

Algorithm 4: 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: $x_{i,j}, u_i^k, Q_i$.
$n = 0,\ q_k = 0$.
while $n \leq P_1 + P_2$:
  Select a P_1 or P_2, and assign it to an ACRs.
  $for\ k \in K$:
    The current location of ACRs is p.
    $if\ q_k \leq Q$:
      When the load of the ACRs is less than Q, a P_1 or P_2 closest to the current position of the ACRs is selected to join in the path of that ACR.
      $x_{p,p_1}\ or\ x_{p,p_2} = 1, u_{p_1}^k\ or\ u_{p_2}^k = 1, Q_{p_1}\ or\ Q_{p_2} = q_k$.
      $n = n + 1, q_k = q_k + 1$.
    $else$:
      Add D_1 or D_2 that already has a P_1 or P_2 counterpart in the path of the ACRs.

12	$q_k = 0.$
13	$end\ \ if.$
14	$end\ \ for.$
15	$end\ \ while.$
Output:	$x_{i,j}, u_i^k, Q_i.$

4.2 Operators Construction

4.2.1 Destructor Operators

The purpose of disruption operators is to remove parts of the current solution. For order assignment and sorting, tote assignment and sorting, the items to be removed are orders or totes, and for robot task assignment and scheduling, it's the task points that need to be removed. For each layer of encoding, three disruption operators are designed in this paper: random disruption operator, worst disruption operator, and Shaw disruption operator. They are described as follows: 1) Random Disruption Operator, this operator randomly selects a certain number of encodings from the current solution for removal. It helps in diversifying the search and prevents getting trapped in local optima. 2) Worst Disruption Operator, this operator removes the encodings with the highest insertion cost and then reinserts them into other positions to obtain a better solution. 3) Shaw Disruption Operator, this operator removes similar encodings simultaneously because similar encodings are easier to rearrange, thus leading to better new solutions.

4.2.2 Repair Operators

The role of repair operators is to reintroduce unassigned encodings back into the current solution. This paper adopts three repair operators: random repair operator, greedy repair operator, and best repair operator. They are described as follows: 1) Random Repair Operator, this operator operates in a serial manner. During each iteration, it randomly selects an encoding from the set of unassigned encodings, finds an insertion position that satisfies the constraints, and inserts the encoding. This process continues until all encodings are inserted. The random insertion operator diversifies the search and prevents falling into local optima. 2) Greedy Repair Operator, during each iteration, this operator selects only one encoding for insertion. The selected encoding is the one with the least global insertion cost. By calculating the insertion cost for each encoding and selecting the one with the minimum cost, it determines the encoding to be inserted greedily. 3) Best Repair Operator, similar to the greedy repair operator, this operator operates in a serial manner. During each iteration, it finds the position with the

minimum insertion cost for each encoding and inserts the encoding accordingly.

4.3 DRL for Operators Selection

In ALNS, the selection of operators directly impacts both the quality and speed of the solution. Particularly, in the context of our study involving three sets of encoding, totaling 18 operators including 9 demolition and 9 repair operators, there theoretically exist 27 combinations of demolition and repair operators. Each combination offers varying degrees of improvement toward the ultimate objective. Hence, the weights and selection probabilities associated with different operator combinations should ideally differ. However, operationalizing this process presents considerable challenges. Consequently, we opt to employ DRL to optimize this aspect. Our deep DRL model assumes responsibility for making operator selection decisions at each iteration, thus replacing traditional methods of operator weight adjustment and selection within ALNS.

Our DRL model, based on a multilayer perceptron, is tasked with extracting features and states from each iteration. We employ the Proximal Policy Optimization (PPO) algorithm to train this model. Within our proposed hyper-heuristic framework, the DRL model leverages effective information at each step to make better decisions in operator selection, thereby enhancing both the quality and speed of ALNS solutions. Subsequently, we will provide a detailed overview of our DRL model, covering aspects such as the environment, multilayer perceptron, and training algorithm.

4.3.1 Environment

1. State

State refers to a set of informative features that guide the intelligent agent in DRL to select the best or optimal heuristic operator combination during each iteration of the ALNS algorithm's search process. In this study, we have chosen a total of 12 features to represent the system's state, as illustrated in Equation (4-3). The specific descriptions of each feature are detailed in Table 4.

$$s_i = (ic_i, ic_i^*, it_i, gc_i, tp_i, cs_i, nt_i, rc_i, wc_i, us_i, bs_i, la_i) \quad (4-3)$$

Table 4. Features for the environment.

Symbol	Description
ic_i	The cost of the current solution.
ic_i^*	The cost of the best solution found so far.
it_i	Number of iterations.
gc_i	Difference in cost between the current solution and the best solution.
tp_i	Current temperature.

cs_i	Cooling time.
nt_i	Number of iterations since the last improvement.
rc_i	Difference in cost between the previous and current solutions.
wc_i	Whether the current solution has the same cost as the previous solution, denoted as 1 if true, and 0 otherwise.
us_i	Whether a solution has been found in the previous search, denoted as 1 if true, and 0 otherwise.
bs_i	Whether a better solution has been found in the previous step of the search, denoted as 1 if true, and 0 otherwise.
la_i	The pair of destruction and repair operators used in the previous iteration.

2. Action

The action of the DRL agent is defined as selecting one of the 27 pairs of disruption and repair operators. In other words, during each iteration of the DRL heuristic, the agent selects a pair of disruption and repair operators, denoted as h, and applies it to solve for a new solution. Therefore, the policy function π is defined as shown in Equation (4-4).

$$\pi(h|s,\theta) = Pr\{A_t = h|S_t = s, \theta_t = \theta\} \quad (4-4)$$

3. Reward

In DRL, a good reward function needs to balance the demands for progressive and incremental rewards, while also preventing agents from exploiting the reward function without optimizing the objective function. Therefore, for our ALNS algorithm framework, we propose a reward function R_t^{5310} with the aforementioned properties, as shown in equation (4-5).

$$R_t^{5310} = \begin{cases} 5, if \ \ f(S_i) < f(S^*) \\ 3, if \ \ f(S_i) < f(S\ ) \\ 2, if \ \ aceept(S_i, S\ ) \\ 1, \quad otherwise \end{cases} \quad (4-5)$$

4.3.2 Feed-forward Neural Network

The feedforward neural network is a fundamental artificial neural network model inspired by the information transmission in biological neural systems. In a feedforward neural network, information can only propagate in one direction, from the input layer through a series of hidden layers, ultimately reaching the output layer, without any feedback connections. The most important components in a feedforward neural network are the input layer, hidden layers, and output layer. Below, we will provide a detailed explanation, and the feedforward neural network used in this paper is illustrated in Figure 6.

Input Layer: The input layer of the feedforward neural network receives the features of the system state X. We represent the 12 features as $x_1 \sim x_{12}$, as shown in Equation (4-6).

$$X = (x_1, x_2, \dots, x_{12}) \quad (4-6)$$

Hidden Layer: The feedforward neural network designed in this paper consists of two hidden layers, each containing 256 neurons.

Output Layer: The output layer of the feedforward neural network is responsible for outputting the probability p_i of selecting a pair of destruction and repair operators in each iteration process. Its calculation is shown in formula (4-7).

$$p_i = \frac{e^{u_i}}{\sum_i e^{u_i}} \tag{4-7}$$

Here, u_i represents the value of each pair of destruction and repair operators output by the feedforward neural network.

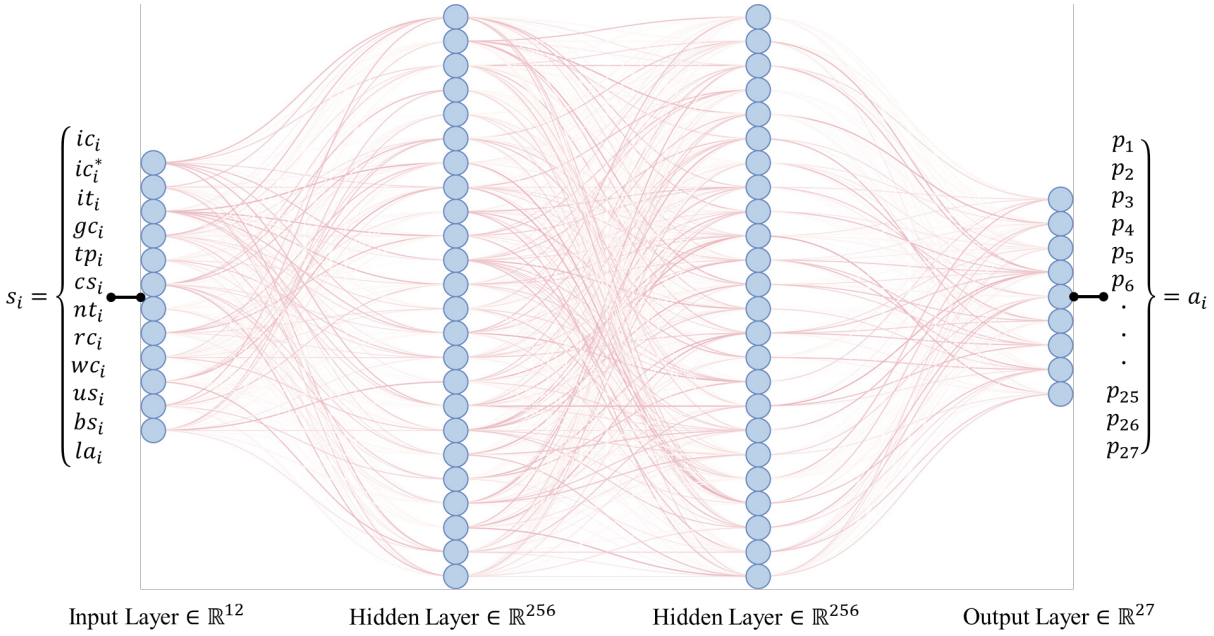


Figure 6. Feed-forward neural network.

4.3.3 Proximal Policy Optimization

For training in DRL, we employ the PPO algorithm, the pseudocode for the algorithm is provided in Algorithm 5. As for training the weight values of the feedforward neural network, we follow the design and specifications outlined in the PPO policy gradient algorithm proposed by Schulman et al. (2017).

Algorithm 5: Proximal policy optimization.	
Input:	Training data.
Output:	Optimal policy π^*.
1	Start with random setting of θ for a random policy π.
2	$for\ e \in E$:
3	Receive initial state S_1.
4	$for\ t \in T$:
5	Choose and perform action $a \in A_t$ according to $\pi(a\|s,\theta)$.

6	Receive $R_t = v$ and $s \in S_{t+1}$ from the environment.
7	$end\ \ for.$
8	Optimize the policy parameters θ according to PPO.
9	$end\ \ for.$
Output:	π^*.

4.4 Acceptance and Termination Strategies

For determining the acceptance probability of a new solution in each iteration, we employ simulated annealing as a criterion. The probability of accepting the new solution S_{new} obtained in the k iteration can be calculated using Equation (4-8).

$$p = min\ \ (e^{\frac{S_{new}-S_{cur}}{T_k}}, 1) \qquad (4-8)$$

Here, T_k represents the current temperature, which is defined as $T_k = T_0 c^k$, where T_0 is the initial temperature and c is the cooling rate.

The termination strategy for the ALNS algorithm in this paper is based on reaching a specified number of iterations or when the number of iterations with no change in the solution reaches a predefined value.

5. Numerical Experiments

We conducted extensive numerical experiments to investigate the performance of the proposed H-ALNS method. Firstly, we provided an explanation of the test case generation, which includes small, medium, and large-scale instances. Next, on these three types of instances, we compared our proposed H-ALNS method with the Gurobi solver, as well as other rule-based algorithms and greedy algorithms. Then, we analyzed the impact of DRL on the performance of H-ALNS. Finally, we conducted sensitivity analysis on the proposed end-to-end algorithm concerning system layout, number of robots, and other factors.

All numerical experiments were conducted on a Windows operating system with a CPU running at 2.90 GHz and 16 GB of RAM (the MIP model was solved using Gurobi 10.0.1). The hyperparameters used during the training process of DRL were as follows: learning rate of $1e^{-5}$, batch size of 64, and discount factor of 0.5.

5.1 Experiment Settings

We randomly generated a series of standard instances for various experiments, including small-

scale, medium-scale, and large-scale instances, as shown in Tables 5, 6, and 7, respectively. Each standard instance is characterized by five main elements: the number of blocks in the length or width direction of the warehouse layout ($W * L$), the total number of pending outbound containers (N), the total number of pending order tasks (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, so each block contains 360 container storage locations.

Table 5. Information on small scale instances.

Small scale instance ID	$W * L$	N	O	R	P
I-1	2*4	2	3	3	2
I-2	2*4	4	3	3	2
I-3	2*4	6	3	3	2
I-4	4*4	8	6	6	4
I-5	4*4	10	6	6	4
I-6	4*4	12	6	6	4
I-7	6*4	14	9	9	6
I-8	6*4	16	9	9	6
I-9	6*4	18	9	9	6

Table 6. Information on medium scale instances.

Medium scale instance ID	$W * L$	N	O	R	P
II-1	6*8	20	10	4	2
II-2	6*8	25	15	6	3
II-3	6*8	30	20	8	4
II-4	8*8	35	25	10	5
II-5	8*8	40	30	12	6
II-6	8*8	45	35	14	7
II-7	10*8	50	40	16	8
II-8	10*8	55	45	18	9
II-9	10*8	60	50	20	10

Table 7. Information on large scale instances.

Large scale instance ID	$W * L$	N	O	R	P
III-1	10*12	60	50	10	5
III-2	10*12	80	60	15	5
III-3	10*12	100	70	20	5
III-4	12*12	120	80	25	10
III-5	12*12	140	90	30	10
III-6	12*12	160	100	35	10
III-7	14*12	180	110	40	15
III-8	14*12	200	120	45	15
III-9	14*12	220	130	50	15

5.2 Comparison of the Fast Decision Method and the Integrated Decision Method

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

5.2.1 Small-scale Instances

The results of the small-scale instances are shown in Table 8 and Table 9, with 10 instances in each category. We took the average as the final result. The experimental results indicate that for small-scale problems, both Gurobi and H-ALNS can achieve optimal solutions. However, in terms of solution time, H-ALNS demonstrates a significant speed advantage, solving problems in milliseconds. Meanwhile, the R* and G* algorithms, which optimize the three stages separately, show noticeable gaps compared to H-ALNS, with differences of 15.78% and 12.97% from the optimal solution, respectively. This highlights the advantages and necessity of jointly optimizing multiple decisions.

Table 8. Gurobi and H-ALNS results for small-scale instances.

Instance ID	Gurobi				H-ALNS					
	UB	LB	$Time_{Gurobi}$	$Gurobi^*$	Obj_{ALNS}	$Time_{ALNS}$	Δ_{UB}^{Obj}	Δ_{LB}^{Obj}	Δ_{g}^{Time}	$ALNS^*$
I-1	572.00	572.00	0.06	10/10	572.00	5.63	0.00%	0.00%	-5.57	10/10
I-2	588.00	588.00	0.47	10/10	588.00	6.67	0.00%	0.00%	-6.19	10/10
I-3	596.50	596.50	28.46	10/10	596.50	7.56	0.00%	0.00%	20.89	10/10
I-4	689.70	689.70	10.88	10/10	689.70	8.84	0.00%	0.00%	2.04	10/10
I-5	694.30	694.30	184.12	10/10	694.30	9.21	0.00%	0.00%	174.91	10/10
I-6	699.40	699.40	340.97	10/10	699.40	9.93	0.00%	0.00%	331.04	10/10
I-7	684.30	684.30	157.35	10/10	684.30	11.11	0.00%	0.00%	146.24	10/10
I-8	684.00	684.00	323.12	10/10	684.00	11.63	0.00%	0.00%	311.49	10/10
I-9	688.70	688.70	392.96	10/10	688.70	12.47	0.00%	0.00%	380.49	10/10
Average	655.21	655.21	159.82	10/10	655.21	9.23	0.00%	0.00%	150.59	10/10

Note. $Gurobi^*$*(*$ALNS^*$*): the number of optimal solutions obtained by Gurobi (ALNS);* $\Delta_{UB}^{Obj} = (UB - OBJ_{ALNS})/UB$; $\Delta_{LB}^{Obj} = (LB - Obj_{ALNS})/UB$; $\Delta_{g}^{Time} = Time_{Gurobi} - Time_{ALNS}$.

Table 9. R* and G* Results for small-scale instances.

Instance ID	Integrated algorithm: H-ALNS		3 stages rule-based algorithm: R*		3 stages greedy algorithm: G*		Gap	
	Obj_{ALNS}	$Time_{ALNS}$	Obj_{R^*}	$Time_{R^*}$	Obj_{G^*}	$Time_{G^*}$	$\Delta_{R^*}^{Obj}$	$\Delta_{G^*}^{Obj}$
I-1	572.00	5.63	578.00	0.00	575.00	0.00	-1.05%	-0.52%
I-2	588.00	6.67	675.60	0.00	662.00	0.00	-14.90%	-12.59%
I-3	596.50	7.56	675.60	0.00	652.80	0.00	-13.26%	-9.44%
I-4	689.70	8.84	753.00	0.00	828.00	0.00	-9.18%	-20.05%
I-5	694.30	9.21	854.60	0.00	828.00	0.00	-23.09%	-19.26%
I-6	699.40	9.93	854.60	0.00	824.80	0.00	-22.19%	-17.93%
I-7	684.30	11.11	819.00	0.00	762.00	0.00	-19.68%	-11.35%
I-8	684.00	11.63	819.00	0.00	763.00	0.00	-19.74%	-11.55%
I-9	688.70	12.47	819.00	0.00	785.20	0.00	-18.92%	-14.01%
Average	655.21	9.23	760.93	0.00	742.31	0.00	-15.78%	-12.97%

Note. $\Delta_{R^*}^{Obj} = (Obj_{ALNS} - Obj_{R^*})/Obj_{ALNS}$; $\Delta_{G^*}^{Obj} = (Obj_{ALNS} - Obj_{G^*})/Obj_{ALNS}$.

5.2.2 Medium-scale Instances

As with the small-scale problems, we used Gurobi, H-ALNS, R*, and G* to solve the medium-scale problems, and the results are shown in Table 10. We were pleasantly surprised to find that even with a large number of decision variables and constraints, Gurobi was still able to provide upper and lower bounds for the problem, with the gap between them being relatively small, around 10%. This

indirectly reflects the efficiency of our model. Meanwhile, H-ALNS also performed well, obtaining better results than the upper bound provided by Gurobi in a very short time and approaching Gurobi's lower bound. We can even reasonably speculate that the result obtained by H-ALNS is the optimal solution. Similarly to the results observed in the small-scale experiments, the results of the three-stage joint optimization were significantly better than those obtained by separately optimizing the stages, as shown in Table 11.

Table 10. Gurobi and H-ALNS results for medium-scale instances

Instance ID	Gurobi				H-ALNS				
	UB	LB	$Time_{Gurobi}$	$Gurobi^*$	Obj_{ALNS}	$Time_{ALNS}$	Δ_{UB}^{Obj}	Δ_{LB}^{Obj}	$ALNS^+$
II-1	1084.90	837.00	>3600	0/10	954.50	17.54	12.02%	-14.04%	10/10
II-2	1134.33	837.00	>3600	0/10	932.40	22.11	17.80%	-11.40%	10/10
II-3	1235.70	837.00	>3600	0/10	922.70	32.69	25.33%	-10.24%	10/10
II-4	1198.10	841.00	>3600	0/10	938.50	32.22	21.67%	-11.59%	10/10
II-5	1236.60	841.00	>3600	0/10	934.00	39.12	24.47%	-11.06%	10/10
II-6	1234.20	841.00	>3600	0/10	930.30	43.17	24.62%	-10.62%	10/10
II-7	1303.40	825.00	>3600	0/10	929.90	48.27	28.66%	-12.72%	10/10
II-8	1200.00	825.00	>3600	0/10	933.30	52.70	22.23%	-13.13%	10/10
II-9	1220.40	825.00	>3600	0/10	928.80	54.37	23.89%	-12.58%	10/10
Average	1205.29	834.33	>3600	0/10	933.82	38.02	22.30%	-11.93%	10/10

Note. $ALNS^+$: the number of better solutions obtained by ALNS.

Table 11. R* and G* results for medium -scale instances

Instance ID	Integrated algorithm: H-ALNS		3 stages rule-based algorithm: R*		3 stages greedy algorithm: G*		Gap	
	Obj_{ALNS}	$Time_{ALNS}$	Obj_{R^*}	$Time_{R^*}$	Obj_{G^*}	$Time_{G^*}$	$\Delta_{R^*}^{Obj}$	$\Delta_{G^*}^{Obj}$
II-1	954.50	17.54	2168.00	0.00	1361.00	0.00	-127.13%	-42.59%
II-2	932.40	22.11	2111.00	0.00	1253.20	0.00	-126.40%	-34.41%
II-3	922.70	32.69	1621.00	0.00	1251.90	0.00	-75.68%	-35.68%
II-4	938.50	32.22	1560.00	0.00	1199.00	0.00	-66.22%	-27.76%
II-5	934.00	39.12	1767.00	0.00	1231.00	0.00	-89.19%	-31.80%
II-6	930.30	43.17	2127.00	0.00	1237.20	0.00	-128.64%	-32.99%
II-7	929.90	48.27	2025.00	0.00	1304.00	0.00	-117.77%	-40.23%
II-8	933.30	52.70	1531.00	0.00	1201.00	0.00	-64.04%	-28.68%
II-9	928.80	54.37	1494.00	0.00	1224.10	0.01	-60.85%	-31.79%
Average	933.82	38.02	1822.67	0.00	1251.38	0.00	-95.10%	-33.99%

5.2.3 Large-scale Instances

Due to the significantly larger number of decision variables and constraints in the large-scale problems compared to what Gurobi can handle, we did not use Gurobi for solving these problems. Instead, we tested H-ALNS, R*, and G* only, and the results are shown in Table 12. H-ALNS still managed to handle problems of various scales well, providing solutions far superior to those obtained by separately optimizing the three-stage decisions within an acceptable time frame.

Table 12. R* and G* results for large-scale instances

Instance ID	Integrated algorithm: H-ALNS		3 stages rule-based algorithm: R*		3 stages greedy algorithm: G*		Gap	
	Obj_{ALNS}	$Time_{ALNS}$	Obj_{R^*}	$Time_{R^*}$	Obj_{G^*}	$Time_{G^*}$	$\Delta_{R^*}^{Obj}$	$\Delta_{G^*}^{Obj}$
III-1	1210.20	65.65	3816.00	0.00	1927.00	0.01	-215.32%	-59.23%
III-2	1195.90	101.02	3705.00	0.00	1858.00	0.01	-209.81%	-55.36%
III-3	1199.00	127.14	3346.00	0.00	1752.00	0.02	-179.07%	-46.12%

III-4	1232.00	174.25	4492.00	0.00	1960.00	0.03	-264.61%	-59.09%
III-5	1243.50	232.70	3848.00	0.00	1749.00	0.04	-209.45%	-40.65%
III-6	1265.70	291.67	3432.00	0.00	1908.00	0.05	-171.15%	-50.75%
III-7	1332.80	315.54	4050.00	0.00	1843.00	0.06	-203.87%	-38.28%
III-8	1399.90	386.86	3573.00	0.00	1894.00	0.08	-155.23%	-35.30%
III-9	1413.80	403.27	3889.00	0.00	1859.00	0.12	-175.07%	-31.49%
Average	1276.98	233.12	3794.56	0.00	1861.11	0.05	-198.18%	-46.25%

5.3 Impact of DRL on ALNS Performance

During the training of DRL, we recorded the rewards and losses throughout the process, as shown in Figure 7 and Figure 8. From the results, we can observe that as the training progresses, both the rewards and losses of the DRL action selection gradually stabilize, indicating that the action selection strategy of DRL tends to stabilize.

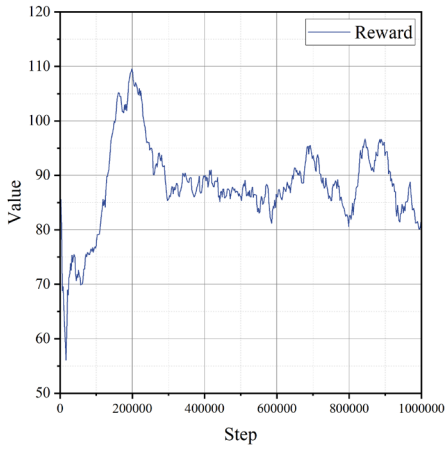


Figure 7. Reward for reinforcement learning

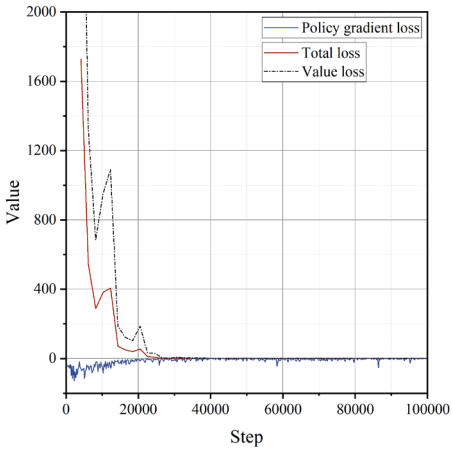


Figure 8. Loss for reinforcement learning

Table 13 Instances for analyzing the impact of DRL on ALNS performance.

Instance ID	$W * L$	N	O	R	P
IV-1	5*5	30	30	10	6
IV-2	6*6	40	40	15	7
IV-3	7*7	50	50	20	8
IV-4	8*8	60	60	25	9
IV-5	9*9	70	70	30	11

Meanwhile, to assess the impact of DRL on ALNS, we conducted tests on it, using the traditional ALNS method for operator selection and weight adjustment as a control. We compared them on five test cases, as shown in Table 13. Our testing mainly consisted of two parts: one part controlled the same number of iterations to compare the differences between traditional ALNS and H-ALNS, while the other part compared their differences in objective values and solution times. The results are depicted in Figure 9 and Figure 10.

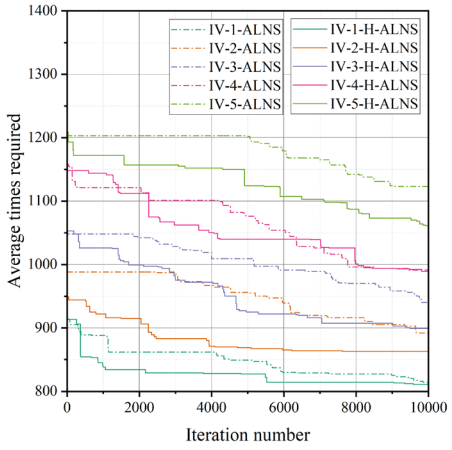


Figure 9. Comparison of iteration number for ALNS and H-ALNS

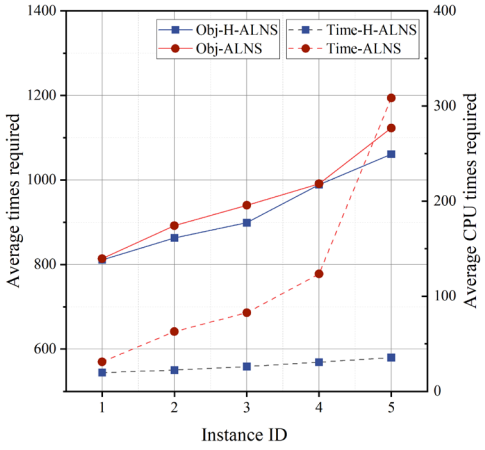


Figure 10. Comparison of ATRs and CPU times for ALNS and H-ALNS

From the results, we observe that the H-ALNS with reinforcement learning converges faster compared to traditional ALNS under the same number of iterations. Moreover, in terms of the final objective function value and solution time, H-ALNS obtains better results and requires less solving time. This demonstrates the effectiveness of reinforcement learning in the operator selection process.

5.4 Sensitivity Analyses on the End-to-end Solution Methods

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 11, 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 12, 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 13, 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.

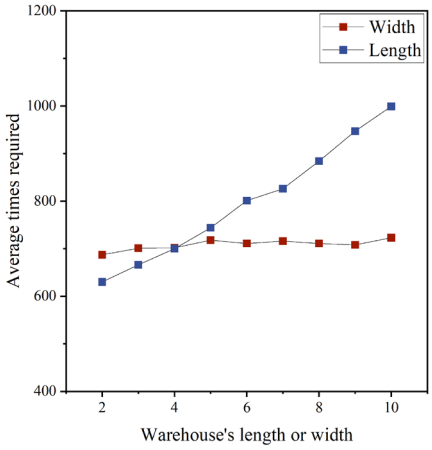
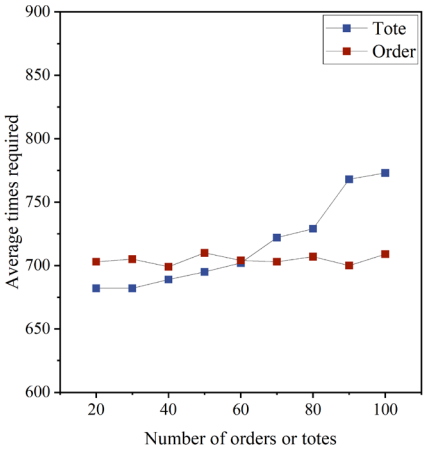


Figure 11. ATRs as the warehouse's length or width change



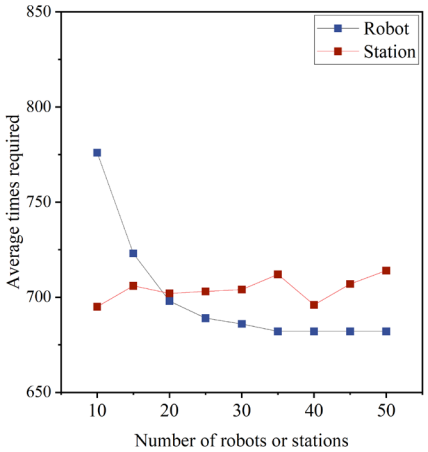


Figure 12. ATRs as the order or tote numbers change

Figure 13. ATRs as the robot or station numbers change

6. Conclusion and Future Research

In this study, we model and analyze an MTSR system with a circular conveyor, jointly optimizing multiple decisions related to orders, totes, and robots. We develop a comprehensive integrated MIP model and design a specialized ALNS algorithm for model solution. Given the numerous decision points that require optimization and the corresponding large number of operators needed, we introduce deep reinforcement learning for operator selection, achieving excellent results. Ultimately, we derive the following conclusions:

Our proposed model and algorithm effectively handle large-scale problems encountered in practical production applications, providing high-quality results for multiple decisions rapidly.

1) Joint optimization of decisions across the three stages of orders, totes, and robots leads to significant improvements in both objectives and efficiency compared to separately optimizing each stage, highlighting the importance and necessity of joint optimization.

2) Integrating deep reinforcement learning into the ALNS algorithm for joint optimization of multiple decisions brings substantial and effective improvements. It addresses the challenge of adjusting operator weights and selecting operators among multiple disruption and repair operators, providing a reliable reference for ALNS to tackle more complex joint optimization problems involving multiple decisions.

3) Our research provides a comprehensive perspective on addressing and solving multiple decisions related to orders, totes, and robots in MTSR, enhancing operational efficiency and offering higher-quality solutions. Additionally, it brings greater convenience to warehouse operators.

In future research, we can explore more precise solutions for the corresponding problem models, including the application of exact or approximate algorithms to further improve solution quality. Furthermore, we can investigate whether machine learning or deep reinforcement learning methods can further accelerate and enhance the speed and quality of problem solving, which is also a worthy research direction.

References

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

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Borovinšek, M., Ekren, B. Y., Burinskienė, A., & Lerher, T. (2017). Multi-objective optimisation model of shuttle-based storage and retrieval system. *Transport*, *32*(2), 120-137.

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

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., & Rousseau, L.-M. (2018). Learning heuristics for the tsp by policy gradient. Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15,

Drori, I., Kharkar, A., Sickinger, W. R., Kates, B., Ma, Q., Ge, S., . . . Udell, M. (2020). Learning to solve combinatorial optimization problems on real-world graphs in linear time. 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA),

Duan, L., Zhan, Y., Hu, H., Gong, Y., Wei, J., Zhang, X., & Xu, Y. (2020). Efficiently solving the practical vehicle routing problem: A novel joint learning approach. Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining,

Gong, Y., Jin, M., & Yuan, Z. (2021). Robotic mobile fulfilment systems considering customer classes. *International Journal of Production Research*, *59*(16), 5032-5049.

Goutham, M., & Stockar, S. J. a. p. a. (2023). Greedy Heuristics Adapted for the Multi-commodity Pickup and Delivery Traveling Salesman Problem.

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),

Joshi, C. K., Laurent, T., & Bresson, X. (2019). An efficient graph convolutional network technique for

the travelling salesman problem. *arXiv preprint arXiv:1906.01227*.

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

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, *30*.

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.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

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, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Lu, H., Zhang, X., & Yang, S. (2020). A learning-based iterative method for solving vehicle routing problems. International conference on learning representations,

Ma, Q., Ge, S., He, D., Thaker, D., & Drori, I. (2019). Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *arXiv preprint arXiv:1911.04936*.

MA, Y., CHEN, H., & YU, Y. (2022). An efficient heuristic for minimizing the number of moves for the retrieval of a single item in a puzzle-based storage system with multiple escorts. *European Journal of Operational Research*, *301*(1), 51-66.

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.

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),

Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, *31*.

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

Qin, Z., Yang, P., Gong, Y., De Koster, R.B.M., 2024. Performance Analysis of Multi-Tote Storage and Retrieval Autonomous Mobile Robot Systems. *Transportation Science* trsc.2023.0397. https://doi.org/10.1287/trsc.2023.0397

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.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

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

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, *28*.

Wang, B., Yang, X., Qi, M. J. F. S., & Journal, M. (2022). 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

Weidinger, F., Boysen, N., & Briskorn, D. (2018). Storage assignment with rack-moving mobile robots in KIVA warehouses. *Transportation Science*, *52*(6), 1479-1495.

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.

Yetkin Ekren, B. (2021). A multi-objective optimisation study for the design of an AVS/RS warehouse. *International Journal of Production Research*, *59*(4), 1107-1126.

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.

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*.