Computers & Industrial Engineering 195 (2024) 110410

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Dynamic unbalanced task allocation of warehouse AGVs using integrated adaptive large neighborhood search and Kuhn–Munkres algorithm

Jianbin Xin [a], Quan Yuan [a], Andrea D'Ariano [b], Guanqin Guo [c], Yanhong Liu [a], Yanjie Zhou [d,*]

[a] School of Electrical Engineering, Zhengzhou University, Science Road 100, 450001, Zhengzhou, China
[b] Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, 00146 Roma, Italy
[c] School of Advanced Technology, Xi'an Jiaotong Liverpool University, 111 Ren'ai Road, 215028, Suzhou, China
[d] School of Management, Zhengzhou University, Science Road 100, 450001, Zhengzhou, China

ARTICLE INFO

Keywords:
Unbalanced task allocation
Automated guided vehicles
Rolling horizon strategy
Integrated metaheuristic

ABSTRACT

Dynamic task allocation poses a complex and challenging decision problem for automated guided vehicles operating within warehouse environments. In this study, we investigate a new method for dynamically allocating unbalanced tasks to a multi-AGV system, with a focus on real-time task arrivals. The research treats this problem as a dynamic vehicle routing problem with pickups and deliveries and proposes the use of a rolling horizon strategy to periodically reallocate tasks by iteratively solving mixed integer programming. To enhance the computational efficiency, a novel metaheuristic is developed, which integrates adaptive large neighborhood search and the Kuhn–Munkres algorithm. Comprehensive numerical experiments are conducted to demonstrate the potential of the proposed approach, in comparison with state-of-the-art heuristics and metaheuristic algorithms, providing insight into the efficiency and effectiveness of the proposed dynamic unbalanced task allocation method for multi-AGV systems in warehouse environments.

1. Introduction

The advancement of automation technology has significantly driven the evolution of Automated Guided Vehicles (AGVs), enabling their utilization for efficiently transporting materials horizontally in diverse scenarios. These scenarios include but are not limited to automotive manufacturing (Cai, Li, Luo, & He, 2023; Xin, Wu, D'Ariano, Negenborn, & Zhang, 2023), warehouse operations (Niu, Wu, Xing, Wang, & Zhang, 2023), container terminals (Xin, Meng, D'Ariano, Wang, & Negenborn, 2022), and other diverse applications. In the warehouse application, materials need to be transported from the starting point to the endpoint, and different materials constitute different tasks that need to be assigned to the AGVs on time, while tasks arriving at different moments and unknown task information bring challenges to task assignment. As the number of tasks and AGVs increases significantly, the solution space becomes vast, making it impractical for any effective algorithm to provide an exact solution within a reasonable timeframe (De Ryck, Versteyhe, & Debrouwere, 2020).

Dynamic unbalanced task allocation stems from the real-world nature of warehousing, where tasks arrive dynamically and the number of available AGVs in a warehouse is often less than the number of goods waiting to be handled. The dynamic nature of the warehouse environment, characterized by the need to assign customer orders that arrive in real-time to AGVs, presents significant challenges for optimizing task allocation among multiple AGVs. This is particularly evident when a new task emerges while an AGV is preoccupied with a specific transportation task. In such a circumstance, a novel modeling methodology and algorithm for addressing dynamic unbalanced task allocation must be developed, whereas the existing literature focuses on static decision-making.

Driven by the above challenge, our paper focuses on enhancing the operational efficiency of dynamically allocating unbalanced tasks to multiple AGVs within a warehouse application. We treat the transport process as a dynamic vehicle routing problem involving pickups and deliveries and devise a rolling horizon strategy to address it. To address the computational efficiency required for real-time decision-making, we decompose the unbalanced allocation into sequencing and balanced assignment and develop a novel integrated algorithm that combines the adaptive large neighborhood search (ALNS) with the Kuhn–Munkres (KM) algorithm. This integrated approach provides significant value for making real-time decisions in the warehouse environment.

* Corresponding author.
E-mail addresses: j.xin@zzu.edu.cn (J. Xin), 1978281592@qq.com (Q. Yuan), andrea.dariano@uniroma3.it (A. D'Ariano), Guanqin.Guo21@student.xjtlu.edu.cn (G. Guo), liuyh@zzu.edu.cn (Y. Liu), ieyjzhou@zzu.edu.cn (Y. Zhou).

https://doi.org/10.1016/j.cie.2024.110410
Received 7 February 2024; Received in revised form 28 June 2024; Accepted 19 July 2024
Available online 27 July 2024

The main contributions of this paper are summarized as follows:

- An investigation into a new task allocation problem for multiple AGVs within a warehouse environment is conducted. Unbalanced tasks are dynamically generated in this scenario and exhibit a periodic pattern. Interestingly, the pertinent literature on warehouse applications focuses on static decision-making. Moreover, the investigated task allocation integrates sequencing and assignment, whereas the prevailing research treats them as two separate stages.
- The investigated dynamic task allocation is treated as a dynamic vehicle routing with pickup and delivery, and a rolling horizon strategy is proposed to reschedule the ongoing and new arrival tasks by solving small-scale mixed integer programming (MIP) iteratively.
- A new algorithm combining ALNS and the KM algorithm is proposed to improve computational efficiency. ALNS provides one overall sequence and KM then breaks the overall sequence for each AGV by solving a matching problem. This new paradigm outperforms the ALNS and other metaheuristics.

The structure of the remaining sections of the paper is outlined as follows: Section 2 reviews the related literature on task allocation of AGVs. Section 3 introduces the dynamic unbalanced task allocation problem and the mathematical model in a rolling horizon manner. Section 4 proposes the integrated ALNS-KM task allocation algorithm. Section 5 discusses numerical experiments on the proposed methodology and conducts a comparative analysis. Finally, Section 6 summarizes this paper and suggests future research directions.

2. Related work

This part reviews the literature concerning task allocation of AGVs in the warehouse and production applications and related vehicle routing problems with pickup and deliveries.

2.1. Task allocation of warehouse AGVs

Task allocation is a fundamental challenge that warehouse AGVs must overcome for autonomous material handling. Prior research on AGV task allocation in warehouses can be classified into static and dynamic task allocation.

In addressing the static task allocation issue, one approach concentrates on the attributes of the warehouse environment, such as extremely narrow aisles and specific order details. Another approach tailors the problem to the characteristics of vehicles, including multi-loading and multi-type capabilities. For the former, He, Aggarwal, and Nof (2018) examined differentiated services for various order types and devised a differential probability queuing strategy to optimize the average total delay of task completion within the warehousing setting. Polten and Emde (2021) delved into the access-constrained nature of narrow aisle warehouses, proposing two access strategies alongside a large neighborhood search algorithm to tackle the AGV scheduling challenge in such environments. Turning to the latter approach, Jiang, Zhang, and Wang (2023) employed a genetic algorithm coupled with a penalty function to facilitate task allocation. Research on AGV scheduling has been conducted based on the multi-loading feature of multi-loading AGVs; Huo, He, Xiong, and Wu (2024) suggested a non-dominated sorting genetic algorithm to address this issue. Similarly, AGV scheduling has been explored considering the variable speed attribute of AGVs, with Liu, Ji, Su and Guo (2019) introducing a multi-adaptive genetic algorithm as a solution. Furthermore, some studies have taken into account both warehouse and AGV traits, exemplified by Maoudj, Kouider, and Christensen (2023), which crafted a mixed integer linear programming model and established behavioral rules for assigning tasks amidst conflicting tasks and multiple loading scenarios.

Regarding the warehouse application, dynamic task allocation has been less studied than static allocation. The available dynamic task allocation is a distributed auction algorithm for the multi-robot task allocation problem (Bai, Fielbaum, Kronmüller, Knoedler, & Alonso-Mora, 2022), to deal with the transportation problem for dynamically occurring packages by introducing the absolute value of the difference between the lowest bid value and the next lowest bid value as a metric.

Table 1 summarizes the above-related work on task allocation for warehouse AGVs. It can be observed that current studies concentrate on static decision-making processes, where allocation is typically divided into separate sequencing and assignment phases. This separation often results in missed opportunities for enhancing performance. In contrast, our paper adopts an integrated approach to sequencing and assignment, exploring the dynamic and unbalanced task allocation scenario.

2.2. Task allocation of production AGVs

In addition to the warehouse, the application of AGVs can also be found in the production system for material transport. AGV's task allocation in the production systems is also categorized into static and dynamic methods.

Regarding the static task allocation, the scheduling of AGVs in a matrix workshop layout with diverse constraints such as loading, charging, and maintenance has been investigated (Zhang, Sang, Li, Han, & Duan, 2022; Zou, Pan, Meng, Gao, & Wang, 2020; Zou et al., 2023). To enhance computational efficiency, a variety of metaheuristic approaches, including the discrete artificial bee colony and adaptive iterative greedy algorithm, have been devised to tackle the customized scheduling challenges individually. Furthermore, the multi-AGV dispatching problem has been integrated with conflict-free path planning, and a time-space network model has been introduced to manage the complexity by leveraging commercial solvers with valid inequalities (Murakami, 2020).

Dynamic AGV task allocation in the production systems comprises market-based (auction-based), learning-based, and optimization methods. Market-based (auction-based) allocation is a key strategy for assigning tasks to AGVs. AGVs bid for tasks, with the highest bidder winning, making it efficient for parallel computing with multiple tasks. For instance, De Ryck, Pissoort, Holvoet, and Demeester (2021, 2022) develop an allocation algorithm based on sequential single-item auctions, which includes resource and routing constraints and can solve allocation schemes that do not violate the constraints. Reinforcement learning optimizes actions based on experience, often selecting from heuristic rules (Yin, Liu, & Wang, 2022). For instance, deep Q networks were used by Hu, Jia, He, Fu, and Liu (2020) for real-time AGV allocation in the workshops. Deep reinforcement learning has also been applied to AGV scheduling, considering battery constraints (Singh, Akcay, Dang, Martagan, & Adan, 2024; Zhang, Yan and Hu, 2023), and a tailored reinforcement learning algorithm was developed for dynamic task allocation in integrated logistics, ideal for small-batch individualized orders (Lei, Hui, Chang, Dassari, & Ding, 2023). Optimization-based methods typically build up a mathematical model using knowledge from operations research to precisely present a relationship between the objective (e.g., makespan) and the allocation decision variables. The static allocation of AGVs for the matrix workshop has been extended to a dynamic version by reassigning AGVs for new tasks and special cases within the planning horizon (Li et al., 2023). A similar

Table 1
Summary of the related work on the task allocation of warehouse AGVs.

Literatures	New mission arrivals	Unbalance	Strategy	Objective function	Noval constraints	Model	Approach
He et al. (2018)	–	✓	ABS	Weighted completion time	–	MINLP	DPQ
Polten and Emde (2021)	–	✓	SAA	Makespan	–	MILP	LNS
Jiang et al. (2023)	–	–	–	Makespan and energy consumption	Time window and energy	MINLP	GA
Huo et al. (2024)	–	✓	SBA	Delay and energy consumption	Time window and energy	MIP	NSGA-II
Liu, Ji et al. (2019)	–	✓	ABS	makespan and energy consumption	Speed and energy	LP	MAGA
Maoudj et al. (2023)	–	✓	ABS	Maximum traveling distance	Capacity and product	MILP	BR
Bai et al. (2022)	✓	✓	SBA	Minimum total travel time	Time window and capacity	MIP	AA
Our paper	✓	✓	SAA	Total completion time and makespan	–	MIP	ALNS-KM

SAA(Sequencing and assignment), ABS(Assignment before sequencing), SBA(Sequencing before assignment) DPQ(Differentiated Probabilistic Queuing), LNS(Large Neighborhood Search), NSGA-II(Non-dominated Sorting GA), MAGA(Multi-Adaptive Genetic Algorithm), BR(Behavioral Rules), AA(Auction Algorithm).

scenario is considered by inserting more customers into the static allocation to include pickup and delivery requests (Zhang, Sang, Li, Zhang and Meng, 2023).

2.3. VRP with pickup and delivery

In a warehouse setting, a fleet of AGVs complete a series of materials transport requests, each with a designated pickup location and a certain delivery point, and the necessity to navigate between these sites. This scenario is identified as a Vehicle Routing Problem with Pickup and Delivery (VRPPD), in the domain of operations research (Desaulniers, Desrosiers, Erdmann, Solomon, & Soumis, 2002).

VRPPD has been extensively employed in service sectors requiring point-to-point pickups and deliveries, including courier services, food delivery, and passenger transport. For instance, Jiang, Dai, Yang, and Ma (2024) analyzed a multi-point access VRPPD, integrating trucks and drones to meet pickup and delivery demands in rural areas, proposing a tailored large-scale search algorithm. Teng et al. (2021) explored optimizing merchant-customer satisfaction in food delivery, considering delivery mileage and time window constraints, and enhancing the genetic algorithm to reduce delivery costs. Zhang, Chen, Yu and Wang (2021) researched route optimization for suburban demand-responsive transit services under exceptional circumstances causing order fluctuations, designing a branch-and-price algorithm to cut operational costs. Chen, Wang, Wang, Qu, and Ma (2021) studied the multi-trip, multi-pickup, and delivery problem with time windows for customized buses, generating initial solutions based on a modified sweep method and improving them using adaptive variable neighborhood search.

2.4. Summary

As a summarizing remark, dynamic task allocation of AGVs in the warehouse application has not been sufficiently investigated. The current literature focuses on static task allocation, requiring a new methodology to deal with new arrivals of tasks computationally efficiently.

3. Problem description and modeling

This section first describes the task allocation problem studied in this article and establishes a mathematical model.

3.1. Problem description

The research considers a warehouse system featuring multiple Input/Output (I/O) stations and narrow aisle shelf rows (Polten & Emde, 2021). Goods handling tasks are performed by multiple AGVs, with each AGV responsible for transporting goods from a specific pickup point to a designated delivery point, as depicted in Fig. 1. The AGVs under consideration are similar to single-reach trucks (often used in narrow aisle applications) in that they can only pick or place goods at one depth of the shelf. These pickup and delivery points can correspond to a particular I/O station or a specific shelf location. The primary objective of task assignments is to minimize the combined total completion time and makespan.

In the background of large-scale goods storage and e-commerce, customer orders arrive dynamically and periodically in real-time (Ghassemi & Chowdhury, 2022). Typically, the number of tasks arriving each time equals or exceeds the number of available AGVs. For instance, for a scenario consisting of 2 available AGVs, when 4 tasks arrive, they could be assigned to 2 AGVs, as depicted in Fig. 1. The corresponding real-time scenario is illustrated in Fig. 2. This paper addresses the dynamic version of the unbalanced assignment problem, where the number of dynamically arriving tasks surpasses the number of AGVs. As new tasks arrive at different time points, AGVs may need to transport ongoing tasks before completion.

We refer to Leeckm et al. (2019) and Zhang, Wu, Zhang, Peng and Zheng (2021) to make some important assumptions as follows:

- The actual collision of AGV in the warehousing environment is not considered;
- The speed of the AGV is fixed at 1 grid per second, and it takes 1 s to go to the adjacent point;
- At any time, only one AGV can be assigned to each task;
- At any time, each AGV can be assigned multiple tasks, but can only complete one task at a time, and cannot execute another task without completing the current task;
- The initial position of the AGV is randomly provided;
- The AGV roadmap allows bi-directional movement;
- Warehouse AGVs are single-reach trucks that can only transport materials to the nearest shelf.

3.2. Rolling horizon framework

Rolling horizon policy is suitable for recurring, dynamic, or multi-periodic problems requiring immediate decisions with updated data (Cuisinier, Lemaire, Penz, Ruby, & Bourasseau, 2022). It relaxes global planning requirements, enabling local optimal planning (Ji et al., 2022). In dynamic environments, it is difficult to achieve globally optimal planning when future system characteristics cannot be predicted correctly and completely. Rolling horizon policy that enables locally optimal planning is particularly useful in warehousing environments with dynamic task arrivals and limited advance information for which long-term prediction of task information is unreliable (Gkiotsalitis & Van Berkum, 2020). In addition, the rolling horizon framework allows the model to adapt to dynamic environments. This framework involves setting the planning start time and horizon, rolling tasks between

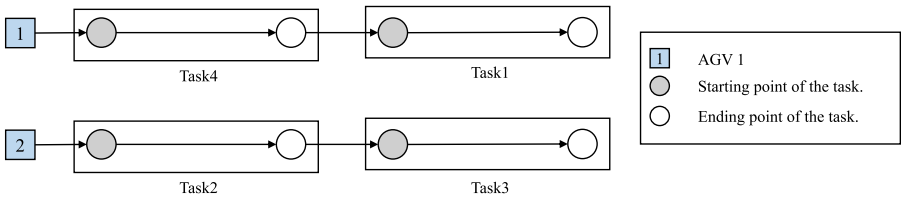

Fig. 1. Introduction to task allocation.

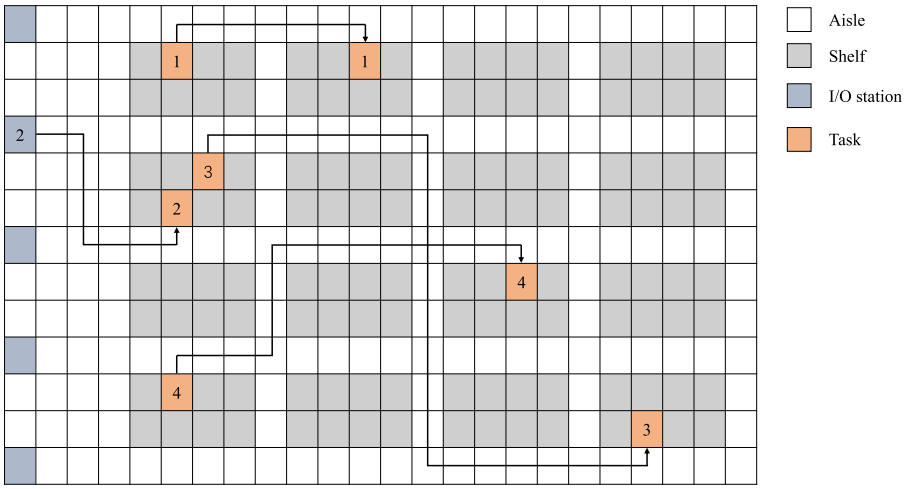

Fig. 2. Task allocation of the multiple pickup-delivery pairs in the warehouse environment.

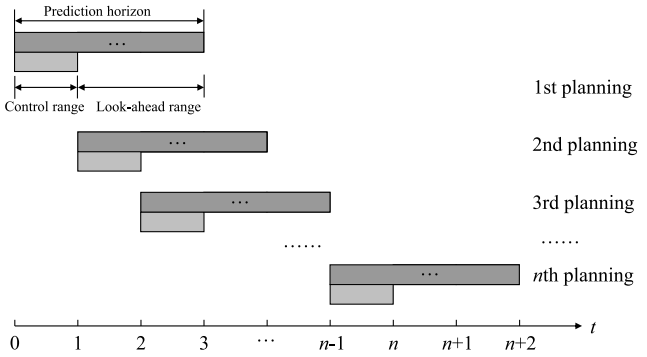

Fig. 3. Schematic diagram of rolling horizon policy.

planning stages, and implementing a stopping rule when all tasks are completed.

Initially, set the planning start time and rolling planning horizon using the initial task arrival moment as the start, the task arrival period as the control horizon, and the task's makespan as the prediction horizon. Upon arrival of new tasks, transition from the previous planning stage to the next, using the output of the previous stage and new tasks as input for the next stage. In this phase, executing tasks enable AGVs to start the next phase's allocated tasks at varying times, and the makespan of tasks executed within the previous phase serves as inputs for the next. Finally, implement a rolling planning stopping rule to halt the process upon completion of all tasks.

The dynamic optimization principle of the rolling horizon policy uses current output as input for the next sampling moment, handling dynamically arriving tasks by adjusting allocations through a mathematical model. This approach enhances the static model's capability to manage dynamic events.

Fig. 3 depicts a schematic of the rolling horizon policy for task arrivals per sampling time, with the predicted time domain representing the makespan of task batches and the control range as the time interval between batch arrivals (set at 1 min for instance). This study introduces a task assignment method based on a time-driven rolling horizon policy, where each moment t signifies specific task assignments. AGVs forecast future assignments within the rolling planning period based on current moment t allocations. Through iterative resolution at different t moments, the assignment scheme for all tasks is ultimately obtained.

3.3. Modeling

This subsection describes the required task allocation model within the rolling horizon framework. It begins by defining the parameters and decision variables and then presents a detailed mathematical model

In our considered problem, each task consists of a starting point and an endpoint, which have different locations in space. To facilitate the construction of the model, we convert the tasks as points, as shown in Fig. 4, which contain the positional information of the starting points and endpoints. Therefore, the distances between two successive tasks are asymmetric due to the different locations of the starting points and endpoints between them.

In warehouse environments, task allocation is commonly analyzed from the perspective of the assignment problem. However, the sequence in which tasks are completed also significantly impacts transportation efficiency. When the sequencing is included, the unbalanced allocation problem can be viewed as a Vehicle Routing Problem (VRP) (Liu, Song, Bucknall and Zhang, 2019).

3.3.1. Mathematical model

Before introducing the mathematical model, Table 2 describes the relevant parameters and decision variables.

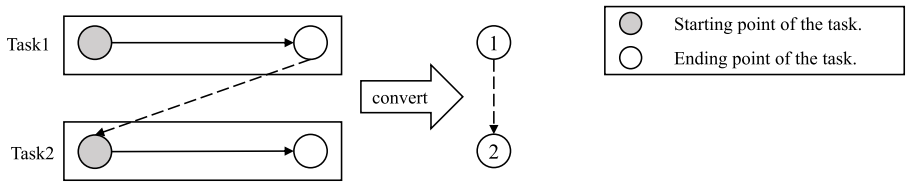

Fig. 4. Convert tasks into task points.

Table 2
Notation for the mathematical models.

Set	Description
V	Set of all the AGVs
Q	Set of all the tasks
k	Index of a particular AGV, $k \in V$
i, j, j', h	Indices of tasks, $i, j, j', h \in Q$
Parameter	
v	Number of AGVs
q	Number of tasks
$d_{i,j}$	Distance between the task i and the task j
$c_{k,i}$	Distance between the AGV k and the task i
E_j	Distance from the starting point to the endpoint of the jth task
a_j	Arrival time of task j
w_1	Weighting factor for the total completion time
w_2	Weighting factor for the makespan
Decision variable	
$x_{i,j}^k \in \{0,1\}$	If AGV k goes from i to j to complete the corresponding task, $x_{i,j}^k$=1, otherwise it is 0; Meanwhile $x_{0,j}^k$=1 means that AGV k goes from the initial position to execute task j
$s_{j,k} \in N$	The time when AGV k starts executing task j. When AGV k does not execute task j, $s_{j,k}$ is meaningless.
T	Makespan

The goal is to minimize the sum of the total completion time and makespan defined as T. The objective function defined as J uses a weighted form for these two objectives. The total completion time originally contains three parts. The first part is the time it takes for all AGVs to go from the initial position to the starting point of their first task. The second part is the time it takes for the AGV to go from the starting point of the task to the endpoint of the task. The third part is the time it takes for the AGV to go from the endpoint of the previous task to the starting point of the next task. When the starting point and endpoint of the task are determined, the value of the second part is constant. Therefore, only the first and third parts are considered for the total completion time in the objective function.

The detailed objective function is described as follows:

$$J = w_1\left(\sum_{j=1}^{q}\sum_{k=1}^{v} c_{k,j}x_{0,j}^k + \sum_{i=1}^{q}\sum_{j=1}^{q} d_{i,j}\sum_{k=1}^{v} x_{i,j}^k\right) + w_2 T \tag{1}$$

where w_1 and w_2 are weighting factors for the total completion time and makespan.

The constraints of the model are as follows:

$$\sum_{i=0}^{q}\sum_{k=1}^{v} x_{i,j}^k = 1, \forall j \in Q \tag{2}$$

$$\sum_{j=1}^{q} x_{0,j}^k \le 1, \forall k \in V \tag{3}$$

$$\sum_{i=0}^{q} x_{i,h}^k - \sum_{j=0}^{q} x_{h,j}^k = 0, \forall k \in V, \forall h \in Q \tag{4}$$

$$s_{j,k} + d_{j,j'} + E_j - M(1 - x_{j,j'}^k) \le s_{j',k}, \forall k \in V, \forall j, j' \in Q \tag{5}$$

$$a_j + d_{k,j} \le s_{j,k}, \forall k \in V, \forall j \in Q \tag{6}$$

$$s_{j,k} \le T, \forall k \in V, \forall j \in Q \tag{7}$$

The constraint (2) indicates that each task can only be completed once by a specific AGV, and also indicates the order in which the AGV completes the task. The constraint (3) indicates that the AGV starts from the initial position and goes to the first task that needs to be completed. The constraint (4) is a continuity constraint, which means that the AGV needs to go from point i to point h before completing the task from point h to point j. The constraint (5) and the constraint (6) limit $s_{j,k}$ to denote the moment when the AGV k arrives at the starting point of the task j. The constraint (5) ensures that the arrival time of an AGV at the next task's starting point is the sum of the AGV's arrival time at the current task's starting point, the time taken by the AGV to travel from the current task's starting point to its endpoint, and the time taken by the AGV to travel from the current task's endpoint to the next task's starting point. The constraint (6) restricts the AGV to be assigned a task only when it has been reached. The constraint (7) restricts the variable T to be the sum of the maximum moment when the AGV reaches the start of a task and the time it takes to complete that task, i.e., the makespan.

Based on the above models and constraints, our task allocation model can be described as p, as shown below.

$$(p) \quad \min\ J$$
$$\text{s.t. } (2) - (7).$$

The problem p is a mixed-integer programming problem typically tackled using commercial solvers like CPLEX. However, it is crucial to understand that similar problems, such as the vehicle routing problem with pickups and deliveries, have been proven to be NP-hard (Desaulniers et al., 2002), demonstrating substantial computational complexity. As a result, relying solely on commercial solvers may not be feasible for addressing complex environments or large-scale instances. Therefore, we propose a tailored metaheuristic approach, complemented by the rolling horizon strategy, to facilitate real-time task allocation.

4. ALNS-KM based algorithm

In this section, we propose a rolling horizon version of ALNS algorithm based on the model constructed in the previous section to allocate real-time batch tasks in a dynamic environment. The first part decomposes the problem, and the second part improves the ALNS algorithm based on the decomposition.

4.1. Decomposition

For the problem p, considering both the assignment of tasks and the task orders makes it difficult to solve. While traditional metaheuristics consider optimizing them simultaneously, this paper further decomposes the unbalanced task allocation problem into a sequencing problem and a balanced task assignment problem.

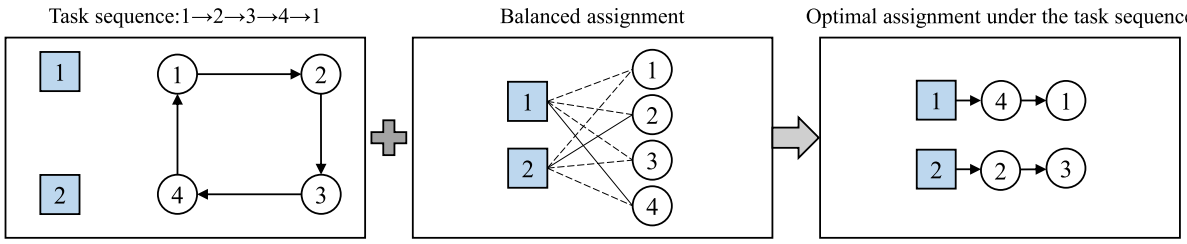

Fig. 5. Schematic diagram of the proposed strategy for task allocation.

4.1.1. Sequencing

Each task consists of a starting point and an endpoint, and the positions of these starting and endpoints vary among tasks. The order in which tasks are completed affects the objective function. A reasonable sequencing method is a prerequisite for obtaining the optimal solution. Traditionally, each task is abstracted as a point, and the conventional sequencing approach connects these points into a directed line. The starting point of this line represents the task that must be assigned as the first task at that moment. However, this approach overlooks the relationship between the starting and endpoints of the line. Therefore, this paper proposes a more flexible sequencing method that connects these points into a directed circular structure. With this approach, AGVs can choose any point on the circular structure as the starting point for a task.

4.1.2. Assignment

For the circular structure composed of q tasks, v tasks ($v < q$) are selected and the circle can be divided into v parts. Then, if these v parts are assigned to the same number of AGVs, the assignment is called a balanced assignment problem (Kuhn, 1955). If v exceeds the number of AGVs, it is a problem of unbalanced allocation, as shown in Fig. 5. The leftmost graph in the figure represents the task sequencing, the middle graph shows the balanced assignment (matching) problem and the right graph shows the final allocation result.

Fig. 5 depicts the allocation of 4 tasks to 2 AGVs, where squares represent AGVs and circles represent tasks. The tasks are first concatenated into a directed task circle$\{1 \to 2 \to 3 \to 4 \to 1\}$ that represents the task sequencing, and then different points in the task order are selected to match the AGVs. If AGV1 matches task 4 and AGV2 matches task 2, then task 4 is disconnected from task 3 in the task order, and task 2 is disconnected from task 1. AGV1 goes to complete the section of task order where task 4 is located, and AGV2 goes to complete the section of task order where task 2 is located.

4.2. ALNS-KM algorithm

Following the decomposed structure, this part proposes an integrated ALNS-KM algorithm to solve the task allocation problem involving task sequencing and task assignment together.

ALNS can be used to efficiently explore the space of sequencing combinations based on neighborhood search (Mara, Norcahyo, Jodiawan, Lusiantoro, & Rifai, 2022). It has been widely applied to transportation and logistics optimization problems such as TSP and VRP. The task ordering problem in the unbalanced allocation problem needs to be solved for a reasonable task sequencing, and ALNS can fulfill its needs. The KM algorithm can be used to solve the balanced assignment problem (Delaram, Houshamand, Ashtiani, & Valilai, 2021; Rabbani, Khan, & Quddoos, 2019) and can find the optimal solution in finite time, so the KM algorithm is chosen to solve the balanced assignment problem in the unbalanced allocation problem. The developed ALNS algorithm incorporating KM evaluation is denoted as ALNS-KM. The essential steps of ALNS-KM are depicted in Fig. 6.

Table 3
Notation for algorithm.

Notation	Description
C	Encoding
M_{AT}	Distance matrix between AGVs and tasks
M_{TT}	Distance matrix between tasks
M_W	Weight matrix between AGVs and tasks
S, s	Solution
T_s	Simulation task set
T_{min}	Initial temperature
α	Temperature cooling rate
$iter_m$	Maximum iterations
S_{best}	Optimal solution
C_{best}	Encoding of the optimal solution
J_{best}	Objective value of the optimal solution
ω_d	Selection weight of the destroy operator
ω_r	Selection weight of the repair operator
$iter$	Iterations
$C_0, (S_0, J_0)$	Initial encoding(solution,objective value)
$C_{new}, (S_{new}, J_{new})$	New encoding(solution,objective value) for each iteration
T	Temperature
T_L	Rolling horizon length
T_w	Collection of tasks waiting to be assigned
T_a	Allocation results obtained at each rolling planning
T_f	First task to be executed in T_a
M_a	Completion time matrix for executing tasks

Before introducing the algorithm, Table 3 gives the relevant symbols and meanings.

4.2.1. Encoding and decoding

Designing a clear and efficient encoding is a key part of heuristic design. ALNS is used to search the space of sequencing combinations, so the encoding is used to represent the sequencing of tasks. The encoding is represented by the list $C = \{1, 2, 3, 4, \ldots, 1\}$. Each element in the list represents a task index, and the last element is the same as the first element. This encoding represents the sequence between tasks as $\{1 \to 2 \to 3 \to 4 \to \cdots \to 1\}$.

The decoding scheme relies on the KM algorithm, with detailed procedures outlined in Algorithm 1. This algorithm constructs a matrix of matching weights between every task point and each AGV, then employs the KM algorithm to determine the optimal matching solution within this matrix. Given the established matching, the task sequence is partitioned into v segments. Tasks allocated to an AGV are distanced from the preceding task, thereby minimizing the travel time from the previous task to the current one and enhancing the efficiency of the AGV's journey to the current task. Consequently, the weight associated with a task point and an AGV can be defined as the disparity between the distance that the AGV must travel to reach the task point and the distance it would have traveled from the previous task point to the current task point.

The decoding process is shown in the example encoded as $\{1, 2, 3, 4, 1\}$. If the KM algorithm assigns the AGV to task 4, it needs to disconnect

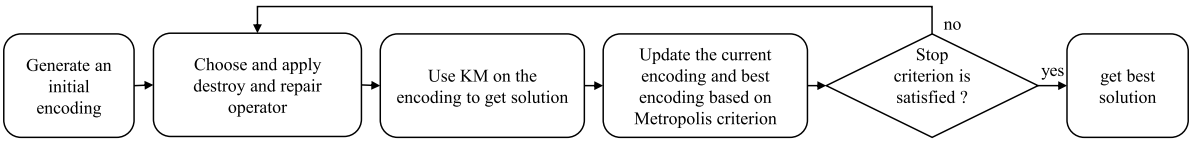

Fig. 6. Flowchat of the ALNS-KM.

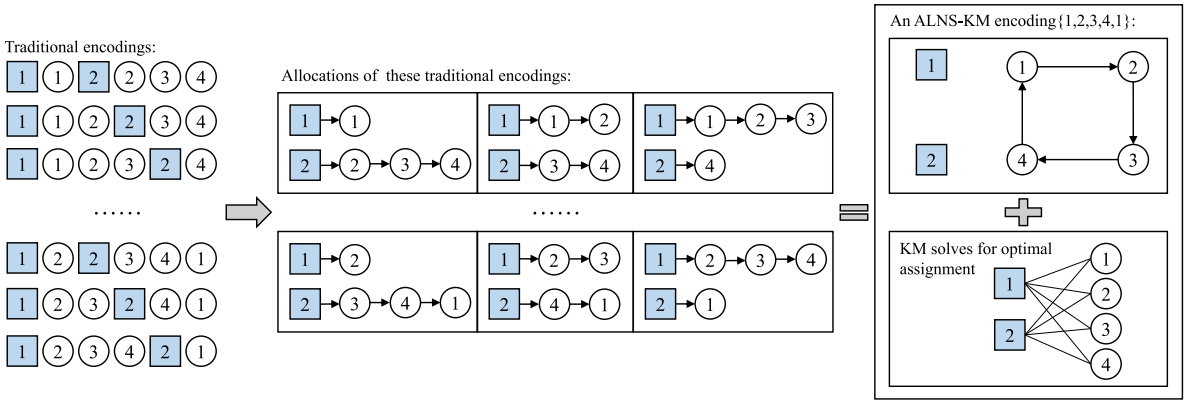

Fig. 7. Difference between the proposed encoding and the traditional encoding.

the connection between task 3 and task 4. Each additional AGV divides the task sequencing circle by one more segment. The first task assigned by each AGV is the point with which it is directly matched, and the rest of the tasks are the segmented section where that point is located.

Algorithm 1 Decode using KM

Input: C, M_{AT}, M_{TT}
1: Initialize M_{W}, S, s
2: **for** $i = 1$ to $|V|$ **do**
3: **for** $j = 2$ to $|C|$ **do**
4: $M_{\mathrm{W}}[i][C[j]] = M_{\mathrm{AT}}[i][C[j]] - M_{\mathrm{TT}}[C[j-1]][C[j]]$
5: **end for**
6: **end for**
7: $s \leftarrow$ Kuhn_Munkres(M_{W})
8: **for** $k = 1$ to $|s|$ **do**
9: $S \leftarrow s[k]$ to the previous element of $s[k+1]$ in C
10: **end for**
Output: S

The encoding represents a task sequence, while the decoding is the assignment process based on the task sequence. Fig. 7 depicts the difference between the traditional encodings and the encodings in this paper in a scenario consisting of 2 AGVs and 4 tasks, with multiple traditional encodings on the left, the solutions corresponding to these encodings in the center, and the solution resulting from the combination of an ALNS-KM encoding and decoding on the right (see Fig. 5 for details). With this ALNS-KM coding, 2 tasks out of 4 must be selected to match with 2 AGVs so that the coding can constitute A_4^2 allocation schemes. For the scenario of v vehicles and q tasks, one encoding in this encoding method corresponds to A_q^v encodings in the conventional encoding.

Selecting the best allocation scheme among A_q^v possible schemes means choosing v tasks from q tasks to assign to v AGVs in such a way that J is optimized. This is a balanced assignment problem, and the KM algorithm can be used to compute the optimal solution within a finite time (Chopra, Notarstefano, Rice, & Egerstedt, 2017). The process is shown in Algorithm 1.

4.2.2. Operator competition

The ALNS algorithm selects the search direction based on the competition of different operators so that the weight of operator i being selected is ω_i. When ALNS iterates, the operator selection weight ω_i will change with the performance of the corresponding operator. The weight of the operator with good performance will become larger and larger, while the weight of the operator with poor performance will decrease relatively. Small, to choose a better operator. The performance score of this operator needs to be updated with iterations. When the new solution obtained by using the destruction and repair operators is better than the optimal solution, the score of the corresponding operator is increased by μ_1; when the new solution is better than the current solution and worse than the optimal solution, the score of the corresponding operator increases by μ_2; when the new solution is worse than the current solution, but is accepted by the Metropolis acceptance criterion, the score of the corresponding operator is increased by μ_3; when the new solution is worse than the current solution, and is not accepted by the Metropolis acceptance criterion, the corresponding The operator's score increases by 0.

The update formula of the selection weight ω_i of operator i is:

$$\omega_i = (1-b)\omega_i' + b\frac{P_i}{N_i}, \tag{8}$$

where ω_i' represents the weight before the update, b is the reaction parameter, P_i is the performance score of the operator i, N_i is the operator number of uses of the operator i.

4.2.3. Destruction operator and repair operator

The destruction operator breaks the original structure of the encoding and is combined with the repair operator to generate a new encoding in the neighborhood of the original encoding. As can be seen from Fig. 7, one encoding corresponds to multiple allocations, and the KM algorithm can be used to obtain the optimal allocation among these allocations, which greatly improves search efficiency. Therefore, this article only uses two damage operators and two repair operators. to search for a better solution. (1) Random destruction operator, which randomly removes a task from the current encoding; (2) Maximum impact factor destruction operator, which removes the task that has

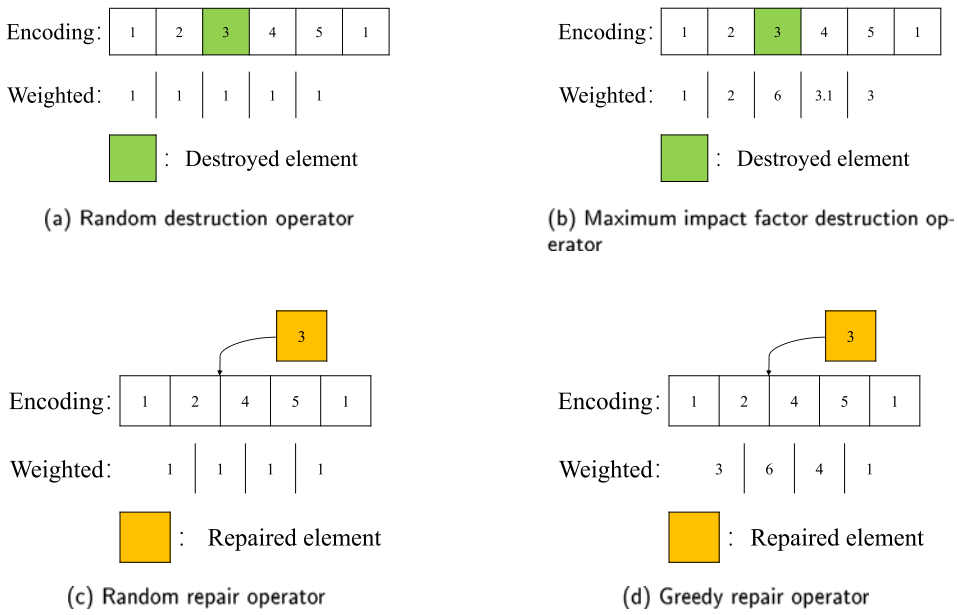

Fig. 8. Schematic diagram of destruction operator and repair operator.

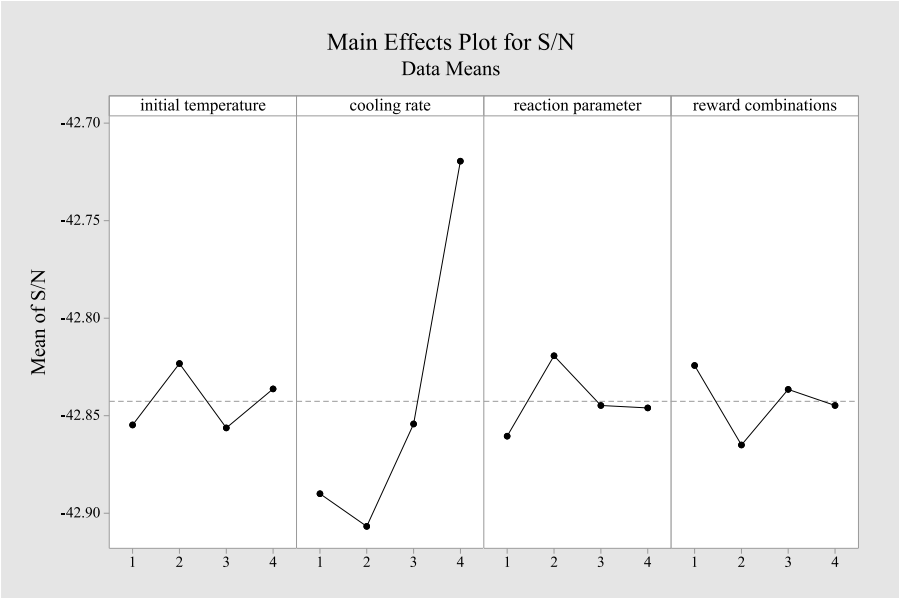

Fig. 9. The mean S/N plot for different levels of the parameters for T20A3I1-A1.

the greatest impact on decoding in the encoding; (3) Random repair operator, this operator randomly inserts the removed tasks into the encoding; (4) Greedy repair operator: This operator traverses the removed operators and inserts them into the encoding, and selects the one with the best KM algorithm evaluation position insert, as shown in Fig. 8, where the weighted indicate the weighted that select that location for destruction (repair). The weighted of the random destruction (repair) operator are all the same, the weight of the greedy repair operator is the optimal evaluation value of traversing each location, and the weight of the maximum impact factor destruction operator is calculated as:

$$W_m = M_{\mathrm{TT}}[C[m-1]][C[m]] + M_{\mathrm{TT}}[C[m]][C[m+1]] - M_{TT}[C[m-1]][C[m+1]] \quad (9)$$

where W_m denotes the weight of the mth position of the encoding to be destroyed.

4.2.4. Basic ALNS-KM framework

The ALNS algorithm improves the solution obtained through the competition of different algorithms. Algorithm 2 gives the basic framework of the ALNS algorithm based on KM evaluation. The encoding in Algorithm 2 does not directly represent the allocation scheme; instead, it needs to be encoded to get the allocation scheme, and then the evaluation value of that encoding is obtained. After initialization, Algorithm 2 selects the appropriate destruction and repair operators based on the operator weights, which break the construction of the initial solution and generate a new solution. The new solution needs to be decoded to get the evaluated value, and the acceptance criterion for the new solution uses the Metropolis criterion for simulated annealing (Ropke & Pisinger, 2006), i.e., the probability of accepting the current solution is $\exp\left(\frac{-(J_{\mathrm{new}}-J)}{T}\right)$. The probability of accepting the current solution

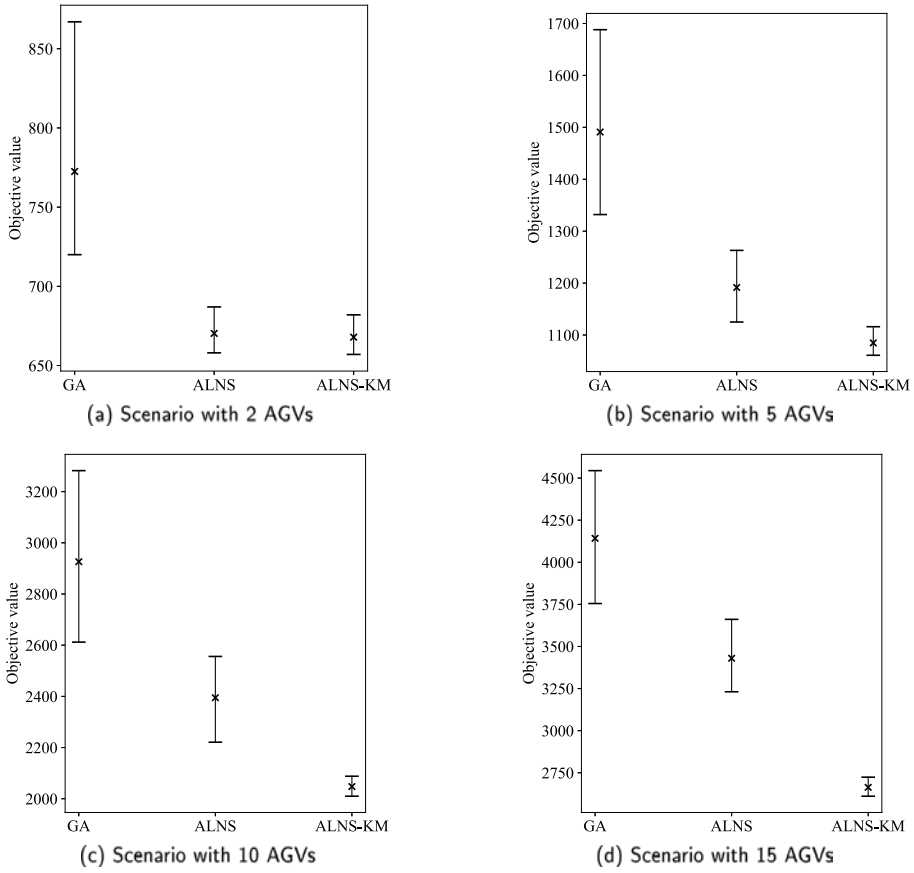

Fig. 10. Comparison of the maximum and minimum values and average results corresponding to the experimental results.

is higher when the current temperature is high and decreases as the temperature decreases.

4.2.5. Rolling horizon version of ALNS-KM

The ALNS-KM algorithm presented earlier is not directly applicable to solving the dynamic unbalanced allocation problem. To address this, a rolling horizon policy adapted to the dynamic environment is proposed in this section. The policy is similar to Section 3.2, and the specific framework is shown in Algorithm 3.

The framework aims to obtain the allocation by periodically solving at the arrival of each new task. In Algorithm 3, the third line reveals that T_w represents the task of each arrival, while the sixth line indicates that tasks executed within the control range of the rolling planning are assigned to the AGVs. The remaining tasks, along with the upcoming arrivals, serve as inputs for the subsequent planning. At the beginning of each planning phase, tasks may already be in the process of being executed by the AGVs. Consequently, the start time of executing the tasks allocated for this phase varies among the different AGVs. Therefore, the completion time of each executed task is recorded in the seventh line.

5. Results and discussions

In this section, we compare the computational results of the proposed ALNS-KM algorithm with existing heuristic rules and metaheuristics. This section first introduces the warehousing scenarios and related parameters and then analyzes the computation example results.

Algorithm 2 ALNS-KM

Input: T_s,T_{min},α, $iter_m$
1: Initialize S_{best}, C_{best}, ω_d, ω_r , $iter$, T
2: Generate initial encoding C_0={1, 2, 3, ..., $|T_s|$, 1}, $C \leftarrow C_0$
3: $S_0 \leftarrow$ Decode(C_0), $S \leftarrow S_0$
4: Generate objective function J_0 based on S_0 and Equation(1), $J_{best} \leftarrow J_0$
5: **repeat**
6: Choose destroy operator $d_i()$ and repair operator $r_i()$ based on ω_d and ω_r
7: $C_{new} \leftarrow r_i(d_i(C))$
8: $S_{new} \leftarrow$ Decode(C_0)
9: Generate objective function J_{new} based on S_{new} and Equation(1)
10: **if** $J_{new} < J$ **then**
11: **if** $J_{new} < J_{best}$ **then**
12: $C_{best} \leftarrow C_{new}$, $J_{best} \leftarrow J_{new}$
13: **end if**
14: $C \leftarrow C_{new}$,$J \leftarrow J_{new}$
15: **else** {Random() < $\exp\left(\frac{-(J_{new}-J)}{T}\right)$}
16: $C \leftarrow C_{new}$,$J \leftarrow J_{new}$
17: **end if**
18: update $\omega_d[i]$ and $\omega_r[i]$ based on Equation(8)
19: $T \leftarrow \alpha T$
20: **until** $T < T_{min}$ or $iter > iter_m$
21: $S_{best} \leftarrow$ Decode(C_{best})
Output: S_{best}

Algorithm 3 Rolling horizon framework of ALNS-KM

Input: T_s, V, T_L
1: Initialize T_f, T_w, T_a, M_a, $iter$
2: **repeat**
3: $T_w \leftarrow T_s[iter * T_L, (iter+1) * T_L - 1]$
4: Generate M_{AT}, M_{TT} based on T_w and M_a
5: $T_a \leftarrow \text{ALNS} - \text{KM}(T_w)$
6: $T_f \leftarrow T_a[iter * T_L, (iter+1) * T_L - 1]$, $T_w \leftarrow T_a \backslash T_f$
7: Update M_a based on the task completion time in the T_f
8: $iter \leftarrow iter + 1$
9: **until** $|T_f| = |T_s|$
Output: T_f

5.1. Scenario settings

To validate the effectiveness of the proposed algorithm, this section presents various task scenarios in a warehouse environment to analyze the proposed method. The warehouse environment is defined in Huo, Zheng, Zhang, and Liu (2022) and Polten and Emde (2021). These examples are categorized based on the number of tasks arriving each time and the total number of task arrivals and are denoted as T10A6I2. Here, T10 signifies that 10 tasks arrive each time, and A6 indicates that these tasks arrive 6 times (if the number of tasks arriving each time is twice the number of vehicles, the arrival time is 30 s; if the number of tasks arriving each time is four times the number of vehicles, the arrival time is 60 s.), and I2 represents the second instance. The tasks' interval arrival is referenced from Ghassemi and Chowdhury (2022).

The proposed ALNS-KM algorithm is compared with the First-come-first-serve (FCFS) heuristic and two classic metaheuristic algorithms: the traditional ALNS and the genetic algorithm (GA) in Jiang et al. (2023). The GA searches for solutions by applying crossover, mutation, and selection operations to the population, and obtains improved solutions after several iterations.

In all the scenarios, We use the objective function (J) and computation time (CT) as performance indicators for all methods. In the objective function, the values of w_1 and w_2 are both set to 1 aiming for a minimal total completion time while minimizing the makespan. For the metaheuristic algorithm, each computation example calculates the mean, maximum, and minimum values 20 times, and the fitness evaluation times are set to 10,000. For the solver GUROBI, the maximum computation time is set to 600 s. All methods are implemented using Python 3.8 programming on Windows 10 (64-bit) system. The computer configuration is Intel's i7-11700 (2.5GHZ) processor with 16 GB of memory.

5.2. Parameter selection

In metaheuristic algorithms, parameters play a crucial role in determining performance (Shi, Liu, & Zhou, 2023). In the designed metaheuristic algorithm, four key parameters require calibration: initial temperature T_{max}, temperature cooling rate α, reaction parameter b, and reward combinations $[\mu_1, \mu_2, \mu_3]$. To obtain the suitable values of these four parameters, Taguchi's method was used (Zhou & Lee, 2020). Four value levels were set for each parameter to determine a suitable value for subsequent experiments, and the range of parameter selection is shown in Table 4.

In this paper, the objective function can be used as a performance metric to evaluate the effect of different parameters. The smaller the objective value obtained by this algorithm indicates that the parameter is more effective. Therefore the smaller-the-better type characteristic S/N is used in Taguchi's method with the following formula:

$$S/N = -10\log_{10}(\frac{1}{n}\sum_{i=1}^{n}(J_i)^2), \tag{10}$$

where J_i denotes the objective value of the ith experiment solved by metaheuristic and n denotes the number of experiments.

The L_{16} orthogonal array of the Taguchi method is adopted to design the experimentation. We conducted experiments for instance T10A3I1-A1 and repeated the test 20 times for each parameter combination of the orthogonal array, and the S/N results are shown in Table 5. The L_{16} orthogonal array and their corresponding parameter settings and S/N are presented in Table 5.

According to the S/N results in Table 5, the results of the response values of each parameter level for each parameter were averaged separately, and the trend of the effect of each parameter level on the results can be obtained, as shown in Fig. 9. Fig. 9 shows how each parameter affects S/N. The slopes of the folded lines in the graph indicate the relative amount of S/N for each parameter; the greater the slope, the greater the effect of the parameter on S/N. The larger the S/N, the better the effect produced by the parameter. So, the parameter values are finally set as: $T_{max} = 200, \alpha = 0.99999, b = 0.4, [\mu_1, \mu_2, \mu_3] = [0.45, 0.3, 0.15]$. Based on preliminary tests, the population size and mutation rate of GA is 50 and 0.01, respectively.

Table 4
Parameter levels for ALNS-KM.

Parameter level	1	2	3	4
T_{max}	100	200	300	400
α	0.8	0.9	0.999	0.99999
b	0.1	0.4	0.6	0.9
$[\mu_1, \mu_2, \mu_3]$	[0.45,0.3,0.15]	[33,13,9]	[45,15,3]	[100,10,1]

Table 5
Taguchi L_{16} orthogonal array and S/N.

No.	T_{max}	α	b	$[\mu_1, \mu_2, \mu_3]$	S/N
1	100	0.8	0.1	[0.45,0.3,0.15]	−42.883
2	100	0.9	0.4	[33,13,9]	−42.918
3	100	0.999	0.6	[45,15,3]	−42.867
4	100	0.99999	0.9	[100,10,1]	−42.751
5	200	0.8	0.4	[45,15,3]	−42.855
6	200	0.9	0.1	[100,10,1]	−42.912
7	200	0.999	0.9	[0.45,0.3,0.15]	−42.820
8	200	0.99999	0.6	[33,13,9]	−42.706
9	300	0.8	0.6	[100,10,1]	−42.908
10	300	0.9	0.9	[45,15,3]	−42.899
11	300	0.999	0.1	[33,13,9]	−42.922
12	300	0.99999	0.4	[0.45,0.3,0.15]	−42.696
13	400	0.8	0.9	[33,13,9]	−42.914
14	400	0.9	0.6	[0.45,0.3,0.15]	−42.898
15	400	0.999	0.4	[100,10,1]	−42.808
16	400	0.99999	0.1	[45,15,3]	−42.725

5.3. Performance comparison

5.3.1. Single-round task scenarios

Table 6 compares the objective values (J i.e., the weighted sum of makespan and the total completion time) and computation time (CT) for single-round task scenarios of two AGVs. The metaheuristic algorithm provides the average objective value over 20 experiments for every instance, along with the standard deviation. It is important to note that a single round of tasks serves as the foundation for multiple rounds of tasks. Δ_{Obj} indicates the performance between the average objective value obtained between ALNS-KM and the benchmark

Table 6
Compared performances of single-round task scenarios. (2 AGVs, UNIT: SECOND).

Instance	Gurobi		GA		ALNS		ALNS-KM		vs. Gurobi	vs. GA	vs. ALNS
	Obj	CT	Obj	CT	Obj	CT	Obj	CT	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T4A1I1	99	0.02	109.3 ± 7.3	0.18 ± 0.01	99.0 ± 0.0	0.31 ± 0.27	99.0 ± 0.0	0.84 ± 0.01	0.0%	−9.4%	0.0%
T4A1I2	116	0.04	124.1 ± 9.9	0.18 ± 0.01	116.0 ± 0.0	0.21 ± 0.00	116.0 ± 0.0	0.83 ± 0.02	0.0%	−6.5%	0.0%
T4A1I3	105	0.02	113.5 ± 4.5	0.18 ± 0.01	105.0 ± 0.0	0.22 ± 0.01	105.0 ± 0.0	0.84 ± 0.02	0.0%	−7.5%	0.0%
T4A1I4	103	0.02	114.8 ± 11.6	0.18 ± 0.01	103.0 ± 0.0	0.22 ± 0.01	103.0 ± 0.0	0.85 ± 0.02	0.0%	−10.3%	0.0%
T4A1I5	105	0.02	112.3 ± 8.2	0.18 ± 0.01	105.0 ± 0.0	0.21 ± 0.01	105.0 ± 0.0	0.85 ± 0.02	0.0%	−6.5%	0.0%
T8A1I1	240	4.1	267.9 ± 14.6	0.24 ± 0.01	242.5 ± 7.0	0.29 ± 0.01	241.0 ± 1.8	1.36 ± 0.19	0.4%	−10.0%	−0.6%
T8A1I2	188	2.3	217.1 ± 19.0	0.24 ± 0.01	189.1 ± 0.8	0.29 ± 0.01	188.6 ± 2.4	1.39 ± 0.20	0.3%	−13.1%	−0.3%
T8A1I3	202	1.5	241.0 ± 13.4	0.24 ± 0.01	203.2 ± 2.0	0.29 ± 0.01	202.2 ± 0.7	1.31 ± 0.02	0.1%	−16.1%	−0.5%
T8A1I4	182	1.6	222.5 ± 13.5	0.24 ± 0.01	186.1 ± 4.6	0.29 ± 0.01	182.0 ± 0.0	1.29 ± 0.04	0.0%	−18.2%	−2.2%
T8A1I5	188	0.4	237.3 ± 17.9	0.24 ± 0.01	191.8 ± 9.0	0.30 ± 0.01	188.2 ± 1.1	1.26 ± 0.02	0.1%	−20.7%	−1.9%
T12A1I1	305	600	352.1 ± 14.3	0.29 ± 0.01	308.1 ± 2.8	0.36 ± 0.01	307.1 ± 2.4	1.72 ± 0.01	0.7%	−12.8%	−0.3%
T12A1I2	312	600	358.6 ± 15.2	0.30 ± 0.01	316.5 ± 3.2	0.37 ± 0.03	316.0 ± 4.1	1.75 ± 0.04	1.3%	−11.9%	−0.2%
T12A1I3	274	600	323.2 ± 14.2	0.30 ± 0.01	279.5 ± 4.6	0.37 ± 0.02	280.3 ± 6.6	1.88 ± 0.14	2.3%	−13.3%	0.3%
T12A1I4	282	600	340.6 ± 19.9	0.30 ± 0.01	285.6 ± 3.7	0.36 ± 0.01	284.0 ± 2.5	1.90 ± 0.28	0.7%	−16.6%	−0.6%
T12A1I5	255	600	303.1 ± 16.5	0.30 ± 0.01	260.1 ± 5.1	0.36 ± 0.01	260.0 ± 4.9	1.81 ± 0.04	2.0%	−14.2%	0.0%
T16A1I1	395	600	440.6 ± 17.4	0.34 ± 0.01	399.6 ± 4.8	0.48 ± 0.16	399.3 ± 4.0	2.32 ± 0.04	1.1%	−9.4%	−0.1%
T16A1I2	395	600	450.8 ± 16.0	0.36 ± 0.02	403.1 ± 9.1	0.43 ± 0.01	402.7 ± 5.7	2.33 ± 0.07	1.9%	−10.7%	−0.1%
T16A1I3	343	600	410.2 ± 18.0	0.34 ± 0.01	348.9 ± 6.1	0.43 ± 0.01	348.1 ± 4.8	2.27 ± 0.09	1.5%	−15.1%	−0.2%
T16A1I4	384	600	452.9 ± 17.9	0.34 ± 0.01	392.1 ± 8.6	0.44 ± 0.01	391.3 ± 9.0	2.29 ± 0.03	1.9%	−13.6%	−0.2%
T16A1I5	319	600	377.4 ± 15.3	0.35 ± 0.01	325.3 ± 4.8	0.44 ± 0.03	324.8 ± 5.5	2.28 ± 0.03	1.8%	−13.9%	−0.2%
Average	239.6	300.5	278.5 ± 118.5	0.27 ± 0.06	243.0 ± 102.3	0.33 ± 0.11	242.2 ± 102.3	1.57 ± 0.55	0.8%	−12.5%	−0.4%

method, as shown in the following formula:

$$\Delta_{\text{Obj}} = \frac{(Obj_{ALNS-KM} - Obj_Z)}{Obj_Z} \tag{11}$$

where Obj_Z and $Obj_{ALNS-KM}$ denote the average objective value obtained by the benchmark method Z and ALNS-KM, respectively and a negative value of Δ_{Obj} indicates that ALNS-KM can obtain a lower objective function value than the benchmark method due to the benchmark method.

Table 6 shows that the computation efficiencies of ALNS-KM and ALNS are close to that of GUROBI for a small number of tasks. As the number of AGVs grows considerably, the minimum values solved by ALNS-KM and ALNS are close to the results of GUROBI solving for 600 s. The average values of ALNS-KM are slightly better than those of ALNS and are significantly better than those of GA. Table 6 also indicates that Gurobi solves poorly for a large number of tasks. Gurobi is poorly solved and unsuitable for dynamic environments. Under the multi-round task scenarios, Gurobi is not recommended due to its computation inefficiency and only heuristic algorithms and meta-heuristic algorithms are discussed under the multi-round task scenarios.

5.3.2. Multiple-round task scenarios

Tables 7–10 compare the computational performance of the four methods for multiple-round task scenarios with different numbers of AGVs. The meta-heuristic algorithm gives the results of 20 experiments - average objective values and standard deviation for every instance. In addition, FCFS has a very short computation time in all scenarios and is therefore denoted by "-". These tables also include the average objective value and the average computation time for the various methods for all scenarios at the bottom. In addition, to show more clearly the differences in the objective values of these metaheuristics, Fig. 10 is used to show the average of the objective mean, maximum, and minimum values corresponding to each table.

Comparison of Tables 7–10 shows that the proposed ALNS-KM is superior to the other three methods. Overall, ALNS-KM can compute the minimum average objective values and standard deviation in the vast majority of cases with reasonable computation time. In Table 7, where the number of AGVs is small, ALNS-KM slightly outperforms ALNS and significantly outperforms GA. This indicates that the solution effectiveness of ALNS-KM and ALNS is similar at this scale. In Tables 8–10, where the number of AGVs is high, the advantage of ALNS-KM, which is indistinguishable in Table 7, is more obvious. In addition, we observe that FCFS has high computational speed and poor computational effectiveness under all the arithmetic cases, which indicates that FCFS does not apply to the unbalanced task allocation problem.

Fig. 10 compares the maximum, minimum, and average objective values for different instances. It is apparent from Fig. 10 that as the problem size increases, the gap between the maximum, minimum, and average objective values of ALNS-KM, ALNS, and GA becomes more obvious. In addition, in Fig. 10(b), (c), and (d), the maximum objective value of ALNS-KM is significantly smaller than the minimum objective value of the other two methods. This indicates that ALNS-KM is significantly better than the other three methods in terms of the objective function. The proposed method shows strong robustness to the objective values, and the gap between the maximum objective value and the minimum objective is significantly smaller than the other two methods.

5.3.3. Convergence curve comparison

Fig. 11 depicts the convergence trends of three metaheuristic algorithms (GA, ALNS, ALNS-KM) during the initial round of task arrivals across different instances: T4A3I1, T4A6I1, T10A3I1, T10A6I1, T20A3I1, and T20A6I1. The data represents the average results obtained from 20 experiments for these instances.

In particular, subfigures (a) and (b) focus on the small-scale scenarios. Subfigure (a) highlights that ALNS-KM achieves convergence at an earlier stage compared to ALNS and GA, with all three algorithms converging to the same value. Conversely, subfigure (b) showcases the scenario where the task count is four times the number of vehicles, demonstrating that ALNS-KM converges earlier than the other two algorithms, with a lower convergence value. Additionally, ALNS-KM is capable of reaching the convergence values of GA and ALNS within a shorter number of iterations. In small-scale scenarios, the computational efficiencies of ALNS-KM and ALNS are close to each other due to the small problem size.

Sub Fig. 11(c)–(f) correspond to medium-scale and large-scale instances. These four subfigures show that ALNS-KM outperforms GA and ALNS consistently in both the convergence speed (in fewer fitness

Table 7
Compared performances of multi-round task scenarios. (2 AGVs, UNIT: SECOND).

Instance	FCFS		GA		ALNS		ALNS-KM		vs. FCFS	vs. GA	vs. ALNS
	Obj	CT	Obj	CT	Obj	CT	Obj	CT	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T4A3I1	553	–	373.9 ± 10.2	0.20 ± 0.02	340.7 ± 1.7	0.25 ± 0.06	340.3 ± 3.1	1.00 ± 0.09	−38.5%	−9.0%	−0.1%
T4A3I2	494	–	297.0 ± 11.8	0.18 ± 0.01	260.0 ± 0.0	0.21 ± 0.00	260.0 ± 0.0	0.93 ± 0.02	−47.4%	−12.5%	0.0%
T4A3I3	549	–	346.6 ± 16.6	0.19 ± 0.01	326.2 ± 3.5	0.22 ± 0.00	323.0 ± 0.0	0.92 ± 0.06	−41.2%	−6.8%	−1.0%
T4A3I4	503	–	327.8 ± 11.5	0.18 ± 0.01	299.1 ± 11.9	0.22 ± 0.01	308.9 ± 16.0	0.92 ± 0.02	−38.6%	−5.8%	3.3%
T4A3I5	584	–	367.3 ± 7.1	0.19 ± 0.03	349.0 ± 0.0	0.22 ± 0.00	347.1 ± 4.9	0.91 ± 0.01	−40.6%	−5.5%	−0.5%
T8A3I1	1045	–	648.6 ± 18.4	0.25 ± 0.01	602.9 ± 4.0	0.31 ± 0.03	609.2 ± 7.5	1.40 ± 0.05	−41.7%	−6.1%	1.0%
T8A3I2	1123	–	687.4 ± 32.4	0.28 ± 0.02	621.2 ± 7.6	0.29 ± 0.01	615.4 ± 7.2	1.44 ± 0.08	−45.2%	−10.5%	−0.9%
T8A3I3	1035	–	671.5 ± 21.0	0.28 ± 0.02	602.7 ± 7.1	0.31 ± 0.01	602.4 ± 6.6	1.48 ± 0.09	−41.8%	−10.3%	0.0%
T8A3I4	984	–	621.1 ± 26.2	0.27 ± 0.01	566.5 ± 6.8	0.30 ± 0.01	567.5 ± 7.3	1.44 ± 0.07	−42.3%	−8.6%	0.2%
T8A3I5	972	–	604.0 ± 23.2	0.26 ± 0.02	550.0 ± 5.9	0.28 ± 0.01	549.0 ± 7.0	1.50 ± 0.19	−43.5%	−9.1%	−0.2%
T4A6I1	1008	–	644.2 ± 21.4	0.23 ± 0.01	585.0 ± 6.9	0.26 ± 0.02	582.9 ± 11.1	1.17 ± 0.05	−42.2%	−9.5%	−0.4%
T4A6I2	1022	–	650.1 ± 20.2	0.22 ± 0.01	612.9 ± 6.4	0.26 ± 0.01	595.2 ± 6.2	1.18 ± 0.04	−41.8%	−8.4%	−2.9%
T4A6I3	1074	–	690.8 ± 20.5	0.23 ± 0.01	625.4 ± 6.1	0.27 ± 0.02	622.0 ± 7.3	1.17 ± 0.04	−42.1%	−10.0%	−0.5%
T4A6I4	1048	–	627.9 ± 22.9	0.23 ± 0.01	555.1 ± 13.3	0.27 ± 0.01	556.5 ± 7.8	1.20 ± 0.03	−46.9%	−11.4%	0.3%
T4A6I5	1156	–	708.0 ± 16.7	0.23 ± 0.01	650.9 ± 8.3	0.28 ± 0.01	650.9 ± 6.5	1.29 ± 0.08	−43.7%	−8.1%	0.0%
T8A6I1	1908	–	1202.8 ± 38.4	0.34 ± 0.02	1064.4 ± 11.9	0.38 ± 0.01	1060.3 ± 10.1	2.05 ± 0.05	−44.4%	−11.8%	−0.4%
T8A6I2	2137	–	1344.9 ± 34.0	0.35 ± 0.01	1228.3 ± 17.8	0.40 ± 0.03	1225.5 ± 13.8	2.17 ± 0.08	−42.7%	−8.9%	−0.2%
T8A6I3	1872	–	1182.0 ± 24.2	0.34 ± 0.02	1071.0 ± 9.3	0.38 ± 0.01	1064.9 ± 5.8	2.06 ± 0.08	−43.1%	−9.9%	−0.6%
T8A6I4	2241	–	1403.8 ± 30.2	0.35 ± 0.01	1306.3 ± 11.1	0.39 ± 0.02	1309.0 ± 11.5	2.16 ± 0.03	−41.6%	−6.8%	0.2%
T8A6I5	2064	–	1296.0 ± 36.9	0.35 ± 0.02	1182.3 ± 11.4	0.39 ± 0.02	1173.2 ± 8.4	2.10 ± 0.09	−43.2%	−9.5%	−0.8%
Average	1168.6	–	734.8 ± 347.6	0.26 ± 0.06	670.0 ± 316.2	0.29 ± 0.06	668.2 ± 314.8	1.42 ± 0.44	−42.6%	−8.9%	−0.2%

Table 8
Compared performances of multi-round task scenarios (5 AGVs, UNIT: SECOND).

Instance	FCFS		GA		ALNS		ALNS-KM		vs. FCFS	vs. GA	vs. ALNS
	Obj	CT	Obj	CT	Obj	CT	Obj	CT	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T10A3I1	948	–	564.1 ± 43.4	0.34 ± 0.02	472.2 ± 19.2	0.42 ± 0.07	430.7 ± 8.9	1.96 ± 0.12	−54.6%	−23.6%	−8.8%
T10A3I2	972	–	576.2 ± 44.5	0.35 ± 0.02	489.9 ± 16.1	0.42 ± 0.02	466.8 ± 6.9	2.05 ± 0.04	−52.0%	−19.0%	−4.7%
T10A3I3	953	–	558.2 ± 31.1	0.36 ± 0.02	460.4 ± 12.3	0.42 ± 0.01	437.8 ± 11.3	2.04 ± 0.05	−54.1%	−21.6%	−4.9%
T10A3I4	1020	–	594.6 ± 42.3	0.36 ± 0.02	504.0 ± 11.9	0.43 ± 0.01	485.4 ± 9.7	2.06 ± 0.04	−52.4%	−18.4%	−3.7%
T10A3I5	1049	–	592.0 ± 34.2	0.37 ± 0.02	516.4 ± 11.6	0.43 ± 0.02	490.8 ± 7.5	2.05 ± 0.04	−53.2%	−17.1%	−5.0%
T20A3I1	1883	–	1128.5 ± 66.8	0.52 ± 0.04	996.1 ± 37.6	0.61 ± 0.03	898.2 ± 14.4	3.49 ± 0.12	−52.3%	−20.4%	−9.8%
T20A3I2	1947	–	1222.8 ± 65.8	0.52 ± 0.03	1021.8 ± 31.0	0.60 ± 0.03	925.3 ± 14.2	3.31 ± 0.17	−52.5%	−24.3%	−9.4%
T20A3I3	2007	–	1276.2 ± 102.4	0.54 ± 0.04	1048.1 ± 38.8	0.60 ± 0.02	958.5 ± 11.1	3.40 ± 0.08	−52.2%	−24.9%	−8.5%
T20A3I4	2058	–	1237.5 ± 85.7	0.54 ± 0.04	1044.9 ± 36.3	0.61 ± 0.02	955.8 ± 18.8	3.51 ± 0.09	−53.6%	−22.8%	−8.5%
T20A3I5	2128	–	1264.9 ± 86.3	0.54 ± 0.04	1082.2 ± 38.0	0.63 ± 0.02	996.0 ± 14.0	3.75 ± 0.08	−53.2%	−21.3%	−8.0%
T10A6I1	2190	–	1293.1 ± 39.4	0.46 ± 0.02	1126.7 ± 27.6	0.55 ± 0.01	1062.8 ± 9.4	2.85 ± 0.06	−51.5%	−17.8%	−5.7%
T10A6I2	2048	–	1192.9 ± 61.6	0.47 ± 0.03	1039.0 ± 21.3	0.53 ± 0.02	963.1 ± 8.7	2.74 ± 0.05	−53.0%	−19.3%	−7.3%
T10A6I3	1965	–	1130.8 ± 48.5	0.47 ± 0.02	971.1 ± 25.8	0.53 ± 0.01	906.2 ± 12.0	2.72 ± 0.08	−53.9%	−19.9%	−6.7%
T10A6I4	2030	–	1159.2 ± 48.7	0.45 ± 0.04	1012.8 ± 22.7	0.53 ± 0.02	929.1 ± 13.3	2.68 ± 0.06	−54.2%	−19.8%	−8.3%
T10A6I5	2080	–	1176.0 ± 49.1	0.46 ± 0.03	1032.2 ± 25.1	0.51 ± 0.02	951.1 ± 13.2	2.57 ± 0.05	−54.3%	−19.1%	−7.9%
T20A6I1	4066	–	2437.8 ± 125.7	0.84 ± 0.07	2156.7 ± 95.6	0.91 ± 0.04	1897.1 ± 19.4	5.32 ± 0.09	−53.3%	−22.2%	−12.0%
T20A6I2	4207	–	2593.1 ± 132.9	0.86 ± 0.05	2248.8 ± 68.0	0.90 ± 0.03	2021.7 ± 20.8	5.73 ± 0.14	−51.9%	−22.0%	−10.1%
T20A6I3	4094	–	2464.8 ± 96.5	0.87 ± 0.05	2203.5 ± 84.1	0.92 ± 0.03	1930.3 ± 27.8	5.47 ± 0.14	−52.9%	−21.7%	−12.4%
T20A6I4	4179	–	2557.8 ± 158.6	0.85 ± 0.05	2191.2 ± 59.4	0.92 ± 0.03	1985.2 ± 22.9	5.82 ± 0.13	−52.5%	−22.4%	−9.4%
T20A6I5	4199	–	2575.1 ± 114.8	0.83 ± 0.05	2213.3 ± 47.0	0.93 ± 0.02	2004.8 ± 24.1	6.06 ± 0.20	−52.3%	−22.1%	−9.4%
Average	2301.2	–	1379.8 ± 717.3	0.55 ± 0.19	1191.6 ± 628.5	0.62 ± 0.19	1084.8 ± 550.4	3.48 ± 1.39	−53.0%	−21.0%	−8.0%

evaluation times) and the convergence value (with a smaller fitness value). Meanwhile, when the number of tasks increases, the advantage of ALNS-KM becomes more considerable in comparison with GA and ALNS. It is also observed that the initial objective values of ALNS-KM are not the same as the other two methods, which is because KM can solve for the optimal allocation scheme under the initial task arrangement.

6. Conclusions and future work

In conclusion, the efficient allocation of tasks for AGVs in a warehouse setting presents a complex and intricate decision problem. This study delves into the dynamic unbalanced task allocation within a multi-AGV system, focusing on real-time task arrivals. By framing this as a dynamic vehicle routing problem with pickups and deliveries, a rolling horizon strategy is proposed to reallocate tasks through iterative solutions of an MIP problem periodically. An innovative algorithm integrating adaptive large neighborhood search and the Kuhn–Munkres algorithm is developed to enhance computational efficiency. Through comprehensive numerical experiments, the potential of this approach is demonstrated, showcasing its competitiveness against existing state-of-the-art heuristics and metaheuristic algorithms. These findings underscore the significance of the proposed dynamic task allocation approach in enhancing the operational effectiveness of multi-AGV systems in warehouse environments.

It can be seen from the numerical experiments that the solution obtained by the ALNS-KM method is close to the exact solution on a small scale, and is significantly better than the other three methods on a large scale. ALNS-KM combines the adaptive search of ALNS with the optimal matching of KM. ALNS searches different task rankings, and KM generates the optimal matching based on the ranking and feeds it back to ALNS as an evaluation value. One task sequencing can correspond to multiple actual allocations, and KM can select the optimal allocation from these actual allocations, greatly improving computing efficiency.

Table 9
Compared performances of multi-round task scenarios. (10 AGVs, UNIT: SECOND).

Instance	FCFS		GA		ALNS		ALNS-KM		vs. FCFS	vs. GA	vs. ALNS
	Obj	CT	Obj	CT	Obj	CT	Obj	CT	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T20A3I1	1719	–	1296.2 ± 79.0	0.50 ± 0.03	1178.5 ± 51.7	0.61 ± 0.07	1118.7 ± 6.5	4.21 ± 0.03	−34.9%	−13.7%	−5.1%
T20A3I2	1766	–	1271.6 ± 67.6	0.48 ± 0.03	1164.8 ± 31.9	0.57 ± 0.02	1103.6 ± 8.7	4.21 ± 0.04	−37.5%	−13.2%	−5.3%
T20A3I3	1903	–	1281.5 ± 62.1	0.48 ± 0.04	1152.0 ± 33.5	0.58 ± 0.02	1065.3 ± 12.4	4.20 ± 0.04	−44.0%	−16.9%	−7.5%
T20A3I4	1914	–	1250.8 ± 89.1	0.49 ± 0.03	1118.3 ± 41.9	0.58 ± 0.02	1059.1 ± 12.9	4.20 ± 0.03	−44.7%	−15.3%	−5.3%
T20A3I5	1952	–	1269.1 ± 56.9	0.50 ± 0.03	1099.2 ± 32.5	0.58 ± 0.01	1025.7 ± 9.7	4.21 ± 0.03	−47.5%	−19.2%	−6.7%
T40A3I1	3745	–	2381.8 ± 111.0	1.12 ± 0.09	2017.0 ± 66.5	1.20 ± 0.05	1685.9 ± 24.5	8.94 ± 0.17	−55.0%	−29.2%	−16.4%
T40A3I2	3748	–	2320.1 ± 124.4	1.13 ± 0.10	1979.2 ± 96.1	1.24 ± 0.08	1646.3 ± 32.2	9.19 ± 0.36	−56.1%	−29.0%	−16.8%
T40A3I3	3698	–	2308.9 ± 128.8	1.11 ± 0.09	1973.5 ± 85.2	1.22 ± 0.06	1607.5 ± 22.0	8.86 ± 0.19	−56.5%	−30.4%	−18.5%
T40A3I4	3774	–	2342.7 ± 136.8	1.11 ± 0.09	1957.0 ± 102.4	1.24 ± 0.04	1659.3 ± 20.5	9.27 ± 0.28	−56.0%	−29.2%	−15.2%
T40A3I5	3645	–	2324.6 ± 150.8	1.07 ± 0.08	1987.2 ± 97.6	1.19 ± 0.05	1623.8 ± 23.1	8.55 ± 0.17	−55.5%	−30.1%	−18.3%
T20A6I1	3746	–	2541.1 ± 130.3	0.55 ± 0.02	2339.8 ± 76.8	0.61 ± 0.05	2192.7 ± 5.6	4.15 ± 0.02	−41.5%	−13.7%	−6.3%
T20A6I2	3854	–	2592.3 ± 131.3	0.57 ± 0.04	2357.8 ± 50.3	0.58 ± 0.01	2163.0 ± 11.1	4.15 ± 0.01	−43.9%	−16.6%	−8.3%
T20A6I3	3743	–	2517.2 ± 68.7	0.58 ± 0.03	2291.6 ± 77.5	0.58 ± 0.02	2160.4 ± 7.3	4.36 ± 0.31	−42.3%	−14.2%	−5.7%
T20A6I4	3845	–	2548.8 ± 75.0	0.60 ± 0.05	2274.6 ± 43.7	0.58 ± 0.02	2118.2 ± 14.8	4.14 ± 0.18	−44.9%	−16.9%	−6.9%
T20A6I5	3917	–	2540.6 ± 103.6	0.59 ± 0.04	2247.9 ± 88.1	0.60 ± 0.02	2049.1 ± 15.1	4.13 ± 0.15	−47.7%	−19.3%	−8.8%
T40A6I1	7485	–	4608.9 ± 205.3	1.87 ± 0.20	4177.6 ± 126.9	1.74 ± 0.06	3342.6 ± 27.0	14.97 ± 0.43	−55.3%	−27.5%	−20.0%
T40A6I2	7531	–	4707.9 ± 304.8	1.77 ± 0.14	4180.2 ± 176.7	1.75 ± 0.06	3330.4 ± 49.7	14.45 ± 0.46	−55.8%	−29.3%	−20.3%
T40A6I3	7705	–	4819.8 ± 288.3	1.79 ± 0.16	4148.6 ± 136.0	1.75 ± 0.07	3419.5 ± 34.7	15.54 ± 0.41	−55.6%	−29.1%	−17.6%
T40A6I4	7469	–	4716.1 ± 201.8	1.79 ± 0.15	4063.5 ± 154.6	1.74 ± 0.07	3265.4 ± 40.2	14.99 ± 0.48	−56.3%	−30.8%	−19.6%
T40A6I5	7477	–	4682.9 ± 283.1	1.80 ± 0.15	4188.1 ± 194.6	1.76 ± 0.07	3316.2 ± 45.9	14.75 ± 0.41	−55.6%	−29.2%	−20.8%
Average	4231.8	–	2716.1 ± 1258.5	1.00 ± 0.53	2394.8 ± 1105.4	1.05 ± 0.49	2047.6 ± 835.2	8.07 ± 4.44	−49.3%	−22.6%	−12.5%

Table 10
Compared performances of multi-round task scenarios. (15 AGVs, UNIT: SECOND).

Instance	FCFS		GA		ALNS		ALNS-KM		vs. FCFS	vs. GA	vs. ALNS
	Obj	CT	Obj	CT	Obj	CT	Obj	CT	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T30A3I1	2522	–	1658.6 ± 112.3	0.86 ± 0.06	1310.2 ± 47.0	0.94 ± 0.04	1065.7 ± 17.0	7.46 ± 0.21	−57.7%	−35.7%	−18.7%
T30A3I2	2657	–	1688.0 ± 72.0	0.84 ± 0.05	1377.5 ± 41.5	0.96 ± 0.03	1104.0 ± 18.7	7.84 ± 0.18	−58.4%	−34.6%	−19.9%
T30A3I3	2724	–	1718.0 ± 86.9	0.86 ± 0.05	1317.6 ± 30.3	0.98 ± 0.04	1168.8 ± 18.3	8.47 ± 0.17	−57.1%	−32.0%	−11.3%
T30A3I4	2670	–	1732.6 ± 84.7	0.84 ± 0.04	1363.2 ± 41.4	0.95 ± 0.03	1130.2 ± 17.2	7.81 ± 0.28	−57.7%	−34.8%	−17.1%
T30A3I5	2925	–	1797.9 ± 72.4	0.88 ± 0.07	1419.5 ± 48.8	0.98 ± 0.03	1184.0 ± 20.9	8.12 ± 0.27	−59.5%	−34.1%	−16.6%
T60A3I1	5478	–	3578.2 ± 195.4	1.51 ± 0.09	3010.2 ± 118.9	1.59 ± 0.08	2410.6 ± 40.3	17.71 ± 0.50	−56.0%	−32.6%	−19.9%
T60A3I2	5584	–	3699.2 ± 197.4	1.54 ± 0.13	2937.6 ± 93.5	1.63 ± 0.11	2444.3 ± 42.8	17.96 ± 0.28	−56.2%	−33.9%	−16.8%
T60A3I3	5468	–	3542.2 ± 163.2	1.48 ± 0.14	2997.1 ± 90.1	1.62 ± 0.13	2393.1 ± 43.9	16.85 ± 0.43	−56.2%	−32.4%	−20.2%
T60A3I4	5384	–	3545.1 ± 233.6	1.51 ± 0.10	3046.5 ± 99.7	1.63 ± 0.09	2368.1 ± 27.4	17.03 ± 0.37	−56.0%	−33.2%	−22.3%
T60A3I5	5357	–	3568.3 ± 113.5	1.54 ± 0.08	2926.1 ± 108.9	1.59 ± 0.05	2339.1 ± 40.6	16.78 ± 0.36	−56.3%	−34.4%	−20.1%
T30A6I1	5663	–	3605.8 ± 194.0	1.42 ± 0.16	2836.4 ± 93.1	1.36 ± 0.05	2284.7 ± 19.9	12.68 ± 0.22	−59.7%	−36.6%	−19.5%
T30A6I2	5386	–	3466.1 ± 123.0	1.42 ± 0.13	2830.2 ± 72.8	1.38 ± 0.09	2213.1 ± 19.5	11.49 ± 0.20	−58.9%	−36.2%	−21.8%
T30A6I3	5538	–	3583.6 ± 200.9	1.38 ± 0.09	2828.7 ± 108.9	1.36 ± 0.05	2292.6 ± 17.6	12.36 ± 0.31	−58.6%	−36.0%	−19.0%
T30A6I4	5711	–	3559.2 ± 194.2	1.31 ± 0.08	2879.4 ± 58.6	1.38 ± 0.04	2362.2 ± 16.6	12.62 ± 0.25	−58.6%	−33.6%	−18.0%
T30A6I5	5525	–	3711.4 ± 223.9	1.38 ± 0.12	2875.2 ± 107.3	1.37 ± 0.04	2345.5 ± 25.4	12.33 ± 0.28	−57.5%	−36.8%	−18.4%
T60A6I1	10 755	–	7529.2 ± 411.7	2.98 ± 0.23	6361.2 ± 198.4	2.74 ± 0.10	4615.3 ± 38.8	27.66 ± 0.68	−57.1%	−38.7%	−27.4%
T60A6I2	11 007	–	7778.4 ± 402.9	2.70 ± 0.17	6570.0 ± 226.8	2.86 ± 0.19	4842.9 ± 50.7	29.47 ± 0.67	−56.0%	−37.7%	−26.3%
T60A6I3	11 200	–	7639.8 ± 382.1	2.70 ± 0.22	6526.4 ± 206.4	2.79 ± 0.12	4897.7 ± 49.2	30.33 ± 0.71	−56.3%	−35.9%	−25.0%
T60A6I4	11 208	–	7740.1 ± 384.2	2.70 ± 0.23	6627.2 ± 211.1	2.87 ± 0.16	4945.1 ± 34.3	29.64 ± 0.58	−55.9%	−36.1%	−25.4%
T60A6I5	11 130	–	7706.7 ± 379.7	2.74 ± 0.25	6559.4 ± 215.7	2.78 ± 0.10	4867.1 ± 50.5	29.58 ± 0.83	−56.3%	−36.8%	−25.8%
Average	6194.6	–	4142.4 ± 2195.8	1.63 ± 0.72	3430.0 ± 1906.8	1.67 ± 0.69	2663.7 ± 1351.6	16.71 ± 8.04	−57.3%	−35.1%	−20.5%

Future research directions could explore the application of machine learning and artificial intelligence techniques to enhance further the performance of dynamic task allocation for multi-AGV systems. Additionally, investigating the integration of real-time environmental data and sensor information could lead to more adaptive and responsive task allocation strategies. Moreover, exploring the implications of this research in broader contexts, such as smart city logistics or automated transportation systems, could offer valuable insights into the scalability and versatility of the proposed methodologies.

CRediT authorship contribution statement

Jianbin Xin: Writing – review & editing, Writing – original draft, Supervision, Project administration, Conceptualization. **Quan Yuan:** Writing – original draft, Software, Methodology, Data curation. **Andrea D'Ariano:** Writing – review & editing, Methodology, Conceptualization. **Guanqin Guo:** Data curation, Visualization. **Yanhong Liu:** Supervision, Writing – original draft, Writing – review & editing. **Yanjie Zhou:** Writing – review & editing, Supervision, Funding acquisition.

Data availability

Data will be made available on request.

Acknowledgments

This research is supported in part by the National Natural Science Foundation of China under Grant 62173311, 72201252, and U23A20340, and in part by the College Youth Backbone Teacher Project of Henan Province under Grant 2021GGJS001.

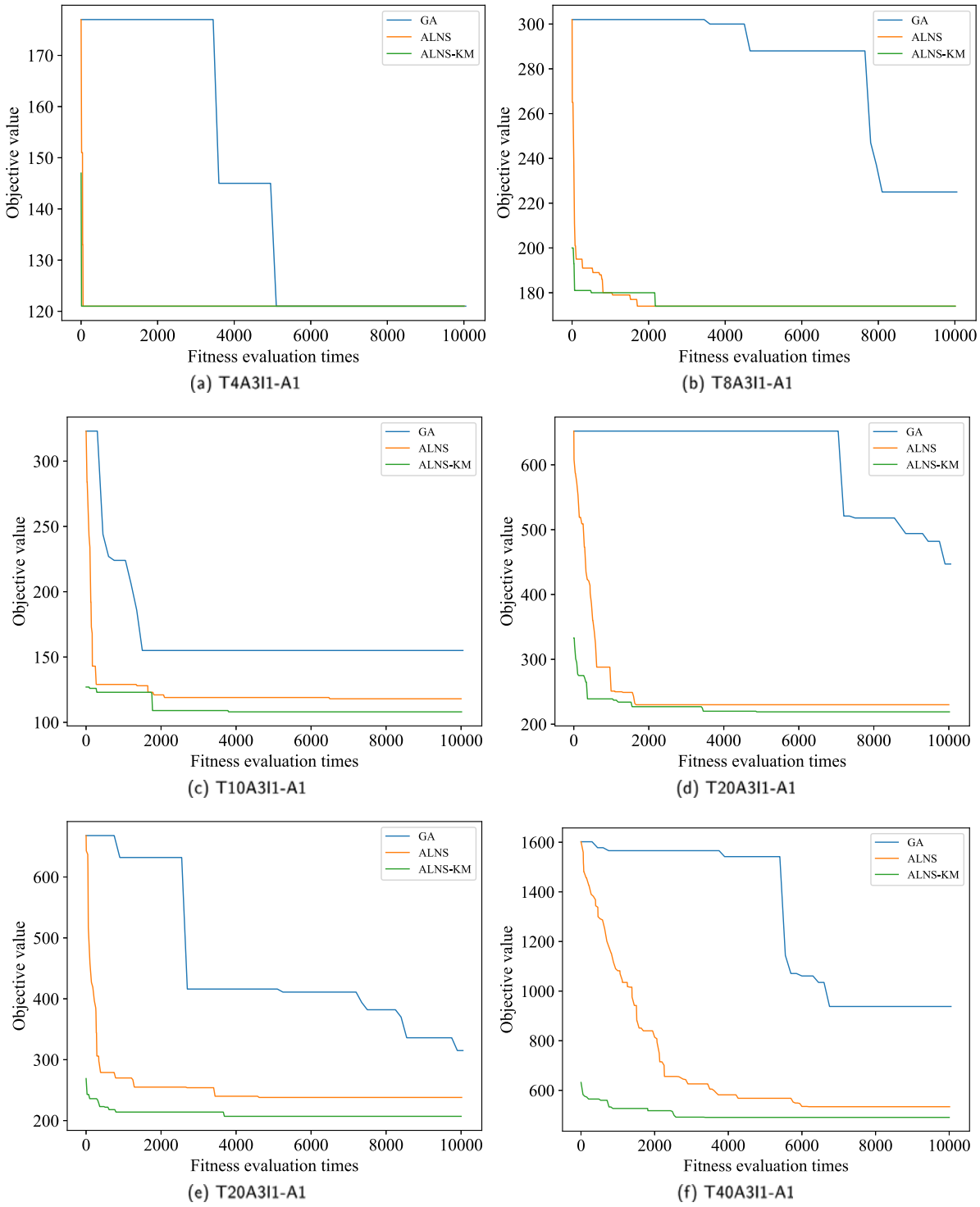

Fig. 11. Comparison of convergence curves of experimental methods.

References

Bai, X., Fielbaum, A., Kronmüller, M., Knoedler, L., & Alonso-Mora, J. (2022). Group-based distributed auction algorithms for multi-robot task assignment. *IEEE Transactions on Automation Science and Engineering*, *20*(2), 1292–1303.

Cai, L., Li, W., Luo, Y., & He, L. (2023). Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment. *International Journal of Production Research*, *61*(5), 1373–1393.

Chen, X., Wang, Y., Wang, Y., Qu, X., & Ma, X. (2021). Customized bus route design with pickup and delivery and time windows: Model, case study and comparative analysis. *Expert Systems with Applications*, *168*, Article 114242.

Chopra, S., Notarstefano, G., Rice, M., & Egerstedt, M. (2017). A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics*, *33*(4), 932–947.

Cuisinier, É., Lemaire, P., Penz, B., Ruby, A., & Bourasseau, C. (2022). New rolling horizon optimization approaches to balance short-term and long-term decisions: An application to energy planning. *Energy*, *245*, Article 122773.

De Ryck, M., Pissoort, D., Holvoet, T., & Demeester, E. (2021). Decentral task allocation for industrial AGV-systems with resource constraints. *Journal of Manufacturing Systems*, *59*, 310–319.

De Ryck, M., Pissoort, D., Holvoet, T., & Demeester, E. (2022). Decentral task allocation for industrial AGV-systems with routing constraints. *Journal of Manufacturing Systems*, *62*, 135–144.

De Ryck, M., Versteyhe, M., & Debrouwere, F. (2020). Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, *54*, 152–173.

Delaram, J., Houshamand, M., Ashtiani, F., & Valilai, O. F. (2021). A utility-based matching mechanism for stable and optimal resource allocation in cloud manufacturing platforms using deferred acceptance algorithm. *Journal of Manufacturing Systems*, *60*, 569–584.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., & Soumis, F. (2002). VRP with pickup and delivery. *The Vehicle Routing Problem*, *9*, 225–242.

Ghassemi, P., & Chowdhury, S. (2022). Multi-robot task allocation in disaster response: Addressing dynamic tasks with deadlines and robots with range and payload constraints. *Robotics and Autonomous Systems*, *147*, Article 103905.

Gkiotsalitis, K., & Van Berkum, E. (2020). An exact method for the bus dispatching problem in rolling horizons. *Transportation Research Part C (Emerging Technologies)*, *110*, 143–165.

He, Z., Aggarwal, V., & Nof, S. Y. (2018). Differentiated service policy in smart warehouse automation. *International Journal of Production Research*, *56*(22), 6956–6970.

Hu, H., Jia, X., He, Q., Fu, S., & Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers & Industrial Engineering*, *149*, Article 106749.

Huo, X., He, X., Xiong, Z., & Wu, X. (2024). Multi-objective optimization for scheduling multi-load automated guided vehicles with consideration of energy consumption. *Transportation Research Part C (Emerging Technologies)*, *161*, Article 104548.

Huo, J., Zheng, R., Zhang, S., & Liu, M. (2022). Dual-layer multi-robot path planning in narrow-lane environments under specific traffic policies. *Intelligent Service Robotics*, *15*(4), 537–555.

Ji, B., Tang, M., Wu, Z., Samson, S. Y., Zhou, S., & Fang, X. (2022). Hybrid rolling-horizon optimization for berth allocation and quay crane assignment with unscheduled vessels. *Advanced Engineering Informatics*, *54*, Article 101733.

Jiang, J., Dai, Y., Yang, F., & Ma, Z. (2024). A multi-visit flexible-docking vehicle routing problem with drones for simultaneous pickup and delivery services. *European Journal of Operational Research*, *312*(1), 125–137.

Jiang, Z., Zhang, X., & Wang, P. (2023). Grid-map-based path planning and task assignment for multi-type AGVs in a distribution warehouse. *Mathematics*, *11*(13), 2802.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97.

Leeckm, L., et al. (2019). Smart robotic mobile fulfillment system with dynamic conflict-free strategies considering cyber-physical integration. *Advanced Engineering Informatics*, *42*, Article 100998.

Lei, J., Hui, J., Chang, F., Dassari, S., & Ding, K. (2023). Reinforcement learning-based dynamic production-logistics-integrated tasks allocation in smart factories. *International Journal of Production Research*, *61*(13), 4419–4436.

Li, Z., Sang, H., Pan, Q., Gao, K., Han, Y., & Li, J. (2023). Dynamic AGV scheduling model with special cases in matrix production workshop. *IEEE Transactions on Industrial Informatics*, *19*(6), 7762–7770.

Liu, Y., Ji, S., Su, Z., & Guo, D. (2019). Multi-objective AGV scheduling in an automatic sorting system of an unmanned (intelligent) warehouse by using two adaptive genetic algorithms and a multi-adaptive genetic algorithm. *PLoS One*, *14*(12), Article e0226161.

Liu, Y., Song, R., Bucknall, R., & Zhang, X. (2019). Intelligent multi-task allocation and planning for multiple unmanned surface vehicles (USVs) using self-organising maps and fast marching method. *Information Sciences*, *496*, 180–197.

Maoudj, A., Kouider, A., & Christensen, A. L. (2023). The capacitated multi-AGV scheduling problem with conflicting products: Model and a decentralized multi-agent approach. *Robotics and Computer-Integrated Manufacturing*, *81*, Article 102514.

Mara, S. T. W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., & Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, *146*, Article 105903.

Murakami, K. (2020). Time-space network model and MILP formulation of the conflict-free routing problem of a capacitated AGV system. *Computers & Industrial Engineering*, *141*, Article 106270.

Niu, H., Wu, W., Xing, Z., Wang, X., & Zhang, T. (2023). A novel multi-tasks chain scheduling algorithm based on capacity prediction to solve AGV dispatching problem in an intelligent manufacturing system. *Journal of Manufacturing Systems*, *68*, 130–144.

Polten, L., & Emde, S. (2021). Scheduling automated guided vehicles in very narrow aisle warehouses. *Omega*, *99*, Article 102204.

Rabbani, Q., Khan, A., & Quddoos, A. (2019). Modified hungarian method for unbalanced assignment problem with multiple jobs. *Applied Mathematics and Computation*, *361*, 493–498.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

Shi, Y., Liu, W., & Zhou, Y. (2023). An adaptive large neighborhood search based approach for the vehicle routing problem with zone-based pricing. *Engineering Applications of Artificial Intelligence*, *124*, Article 106506.

Singh, N., Akcay, A., Dang, Q.-V., Martagan, T., & Adan, I. (2024). Dispatching AGVs with battery constraints using deep reinforcement learning. *Computers & Industrial Engineering*, *187*, Article 109678.

Teng, R., Hong-bo, X., Kang-ning, J., Tian-yu, L., Ling, W., & Li-ning, X. (2021). Optimisation of takeaway delivery routes considering the mutual satisfactions of merchants and customers. *Computers & Industrial Engineering*, *162*, Article 107728.

Xin, J., Meng, C., D'Ariano, A., Wang, D., & Negenborn, R. R. (2022). Mixed-integer nonlinear programming for energy-efficient container handling: formulation and customized genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems*, *23*(8), 10542–10555.

Xin, J., Wu, X., D'Ariano, A., Negenborn, R., & Zhang, F. (2023). Model predictive path planning of AGVs: Mixed logical dynamical formulation and distributed coordination. *IEEE Transactions on Intelligent Transportation Systems*, *27*(7), 6943–6954.

Yin, Z., Liu, J., & Wang, D. (2022). Multi-AGV task allocation with attention based on deep reinforcement learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *36*(09), Article 2252015.

Zhang, L., Chen, T., Yu, B., & Wang, C. (2021). Suburban demand responsive transit service with rental vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *22*(4), 2391–2403.

Zhang, X.-j., Sang, H.-y., Li, J.-q., Han, Y.-y., & Duan, P. (2022). An effective multi-AGVs dispatching method applied to matrix manufacturing workshop. *Computers & Industrial Engineering*, *163*, Article 107791.

Zhang, X., Sang, H., Li, Z., Zhang, B., & Meng, L. (2023). An efficient discrete artificial bee colony algorithm with dynamic calculation method for solving the AGV scheduling problem of delivery and pickup. *Complex & Intelligent Systems*, 1–21.

Zhang, Z., Wu, L., Zhang, W., Peng, T., & Zheng, J. (2021). Energy-efficient path planning for a single-load automated guided vehicle in a manufacturing workshop. *Computers & Industrial Engineering*, *158*, Article 107397.

Zhang, L., Yan, Y., & Hu, Y. (2023). Deep reinforcement learning for dynamic scheduling of energy-efficient automated guided vehicles. *Journal of Intelligent Manufacturing*, 1–14. http://dx.doi.org/10.1007/s10845-023-02208-y.

Zhou, Y., & Lee, G. M. (2020). A bi-objective medical relief shelter location problem considering coverage ratios. *International Journal of Industrial Engineering: Theory, Applications and Practice*, *27*(6), 971–988.

Zou, W.-Q., Pan, Q.-K., Meng, T., Gao, L., & Wang, Y.-L. (2020). An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop. *Expert Systems with Applications*, *161*, Article 113675.

Zou, W.-q., Pan, Q.-k., Meng, L.-l., Sang, H.-y., Han, Y.-y., & Li, J.-q. (2023). An effective self-adaptive iterated greedy algorithm for a multi-agvs scheduling problem with charging and maintenance. *Expert Systems with Applications*, *216*, Article 119512.