Computers & Industrial Engineering 208 (2025) 111355

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Routing fast-charging AGVs for non-stop manufacturing: MILP model and adaptive large neighborhood search

Jianbin Xin [a], Shilong Guo [a], Yanhong Liu [a], Yanjie Zhou [b,*], Andrea D'Ariano [c]

[a] School of Electrical and Information Engineering, Zhengzhou University, Science Road 100, 450001, Zhengzhou, China
[b] School of Management, Zhengzhou University, Science Road 100, 450001, Zhengzhou, China
[c] Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University, 00146 Roma, Italy

ARTICLE INFO

Keywords:
Automated guided vehicles
Routing with fast charging
Non-stop manufacturing
Adaptive large neighborhood search

ABSTRACT

Automated guided vehicles (AGVs) are essential for manufacturing logistics. However, despite their reliance on electric power, the coordination of their routes with charging operations has received little attention, and the productivity remains to be improved. To address this issue, this paper proposes a new methodology for routing AGVs that incorporates fast-charging operations within the production cycle to guarantee (non-stop production. We establish a new mixed-integer linear programming (MILP) model and develop a customized adaptive large neighborhood search (ALNS) to tackle its computational complexity. Specifically, we design a charging insertion heuristic to optimize the recharging strategy and construct feasible solutions. We compare our computational results with those obtained from several state-of-the-art heuristics and metaheuristic algorithms. Our findings demonstrate that the proposed ALNS algorithm outperforms existing methods in terms of both stability and solution quality when addressing this problem.

1. Introduction

Automated Guided Vehicles (AGVs) are autonomous robots used to transport materials, thereby enhancing production efficiency and reducing manual intervention. Exemplified by Industry 4.0 (Mehami, Nawi, & Zhong, 2018), AGVs play a crucial role in this setting and have been extensively deployed in various industrial applications such as container terminals (Angeloudis & Bell, 2010; Cai, Li, Luo, & He, 2023), warehouses (Luo, Zhao, Zhu, & Sun, 2023; Niu, Wu, Xing, Wang, & Zhang, 2023), and manufacturing (Xin, Lu, Wang, & Deng, 2025; Xin, Wu, D'Ariano, Negenborn, & Zhang, 2023). In manufacturing applications, AGVs are essential for logistics, ensuring the timely delivery of materials required during the manufacturing process. With the expansion of the production scale, the scheduling system must effectively allocate materials to meet these challenges, leading to an increased frequency of orders and a growing demand for materials.

The production cycle is considered an efficient and vital tool in modern manufacturing. It offers a systematic approach to managing complex manufacturing processes and achieving operational excellence (Li et al., 2022; Zou, Pan, Meng, Gao, & Wang, 2020). Specifically, it decomposes the overall manufacturing process within the workshop into several continuous production phases and addresses order uncertainty to attain operational excellence. Within such production cycles, the necessary materials must be transported by the AGVs promptly to feed the required machines, and the related routing problems and algorithms have been thoroughly investigated (Chi, Sang, Zhang, Duan, & Zou, 2024; Zou, Pan, Wang, Miao, & Peng, 2022).

Yet, in the context of AGV routing for manufacturing applications, research on charging operations has received little attention, and productivity remains to be improved. For each production cycle, the existing literature commonly assumes that AGVs recharge only after completing all assigned tasks, overlooking the impact of energy constraints (Zou et al., 2023). As charging operations require considerable time, this assumption renders some AGVs, whose remaining battery levels upon returning to the warehouse are insufficient for redeployment in subsequent production cycles, unavailable until recharging processes have been fully completed, thereby leading to low AGV utilization in certain production cycles. This limitation occasionally escalates to extreme scenarios where simultaneous charging demands from multiple AGVs disrupt operational continuity due to vehicle shortages, reducing productivity or even halting production.

Thanks to the fast charging technology (Zhang et al., 2022), it is possible to seamlessly incorporate the charging operations during the transportation phase for routing the electric AGVs and maintain a non-stop production pattern to achieve high productivity. Fig. 1

* Corresponding author.
E-mail addresses: j.xin@zzu.edu.cn (J. Xin), gsl666@gs.zzu.edu.cn (S. Guo), liuyh@zzu.edu.cn (Y. Liu), ieyjzhou@zzu.edu.cn (Y. Zhou), andrea.dariano@uniroma3.it (A. D'Ariano).

https://doi.org/10.1016/j.cie.2025.111355
Received 7 December 2024; Received in revised form 16 May 2025; Accepted 29 June 2025
Available online 12 July 2025

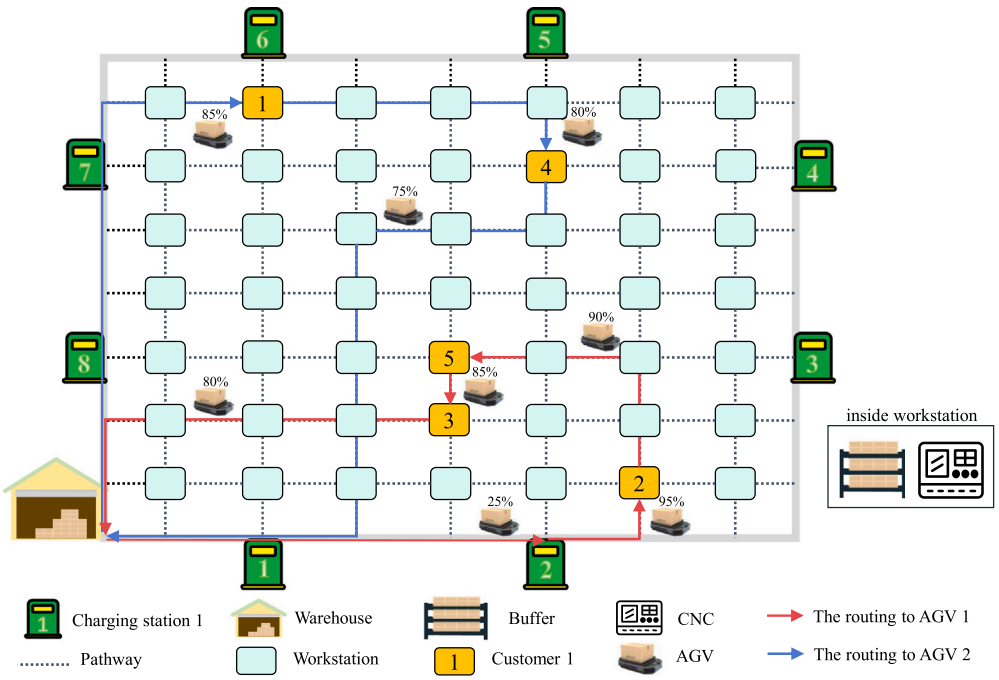

Fig. 1. Graphical illustration of the non-stop production pattern (AGV charges during production, while the AGVs recharge only after completing all assigned tasks in the existing literature).

illustrates this arrangement. In the depicted example, AGV 1 recharges at charging station 2 during the task transport, rapidly restoring its battery level. As a result, all AGVs remain available for redeployment in subsequent production cycles, enhancing production efficiency and ensuring continuous system reliability. In this circumstance, a new mathematical model and an efficient algorithm must be developed to fill this research gap.

To address the above issues, our contributions are summarized as follows:

- We investigate a novel multi-AGV routing problem within a manufacturing context, considering charging operations during the production cycle to facilitate continuous production. Unlike existing methods that either disregard AGV energy constraints entirely or assume charging is conducted only after all tasks are completed, our approach integrates charging activities into the scheduling process, thereby facilitating truly continuous production.
- We develop a new mathematical model that integrates charging-related energy constraints into the multi-AGV routing framework and establishes a mixed-integer linear programming (MILP) formulation. This model captures the interplay between routing decisions and battery limitations, establishing a solid foundation for subsequent optimization.
- To address the computational complexity, we devise a customized Adaptive Large Neighborhood Search (ALNS) algorithm. This algorithm incorporates ten destroy and five repair operators to improve solution quality and search efficiency. Notably, it integrates a charging insertion heuristic that maintains solution feasibility by embedding charging operations directly within the routing process.

The structure of this paper is organized as follows: Section 2 reviews related literature. In Section 3, we describe the multi-AGV routing problem and establish an MILP model. Section 4 proposes the customized ALNS algorithm. Section 5 presents numerical experiments on the proposed methodology and conducts a comparative analysis. Finally, Section 6 summarizes this paper and discusses future directions.

2. Literature review

This part reviews the literature concerning the AGV routing problem in manufacturing applications and logistics applications, as well as its closely related problem, the electric vehicle routing problem (EVRP).

2.1. AGVs in manufacturing systems

AGVs have been widely used in manufacturing systems for transporting materials, and the routing of multiple AGVs involves different types of manufacturing applications. For instance, the multi-AGV routing in an intelligent textile spinning workshop has been investigated, and an improved genetic algorithm has been proposed to meet the real-time application requirement (Farooq, Bao, Raza, Sun, & Ma, 2021). Similarly, the routes of the AGVs are determined by a genetic algorithm for the prefabricated bathroom units manufacturing system to schedule the operation for each workstation and to choose a start time for each unit (Chen, Tiong, & Chen, 2019). For large-scale flexible manufacturing systems, the routing of AGVs is quite challenging. To quickly yield high-quality solutions, an improved Benders decomposition has been proposed using an open-source module theories solver and a commercial constraint programming solver (Riazi & Lennartson, 2021). Moreover, a neural network-based multi-state planning algorithm is developed to make a good trade-off between AGV utilization rate and total processing makespan in flexible manufacturing systems (Wang et al., 2022). To resolve the conflict in the routing problem, a time–space network model has been proposed for a capacitated AGV system that includes splitting and merging manufacturing tasks (Murakami, 2020).

In particular, the multi-AGV routing for computer numerical control machines in the matrix workshop has received considerable attention due to its capacity to enhance space utilization, flexibility, and production efficiency. (Li, Zeng et al., 2018) initially introduced the multi-AGV routing problem under single-load conditions, constructing a mathematical model and designing a harmony search algorithm. Later, Li, Li et al. (2018) proposed an improved harmony search algorithm to tackle the multi-load scheduling problem with a multi-objective model. Building on this foundation, recent research has been developed to

extend to large-scale scenarios. An MILP model is formulated to minimize the transportation cost, including distance cost, penalty cost for violating time constraints, and AGV cost; a discrete artificial bee colony algorithm with new operations has been proposed to solve this problem (Zou et al., 2020). To address dynamic routing, a discrete invasive weed optimization algorithm that incorporates rescheduling strategies and emergency waiting time settings is proposed, enabling the system to adapt to task updates and exceptional cases (Li et al., 2023, 2022). Furthermore, to improve energy efficiency, efficient iterated greedy algorithms have been proposed to deal with different considerations such as charging and maintenance (Zou et al., 2023, 2022).

2.2. AGVs in logistics systems

In addition to manufacturing applications, research on the routing of AGVs is also investigated in logistics applications, particularly those focusing on warehouses and container terminals. Regarding warehouse applications, mathematical programming serves as a fundamental methodology to provide systematic solutions. For instance, the routing of multi-load AGVs for an intelligent sorting center has been investigated, and a multi-objective mixed-integer programming (MIP) model was formulated to find the trade-off among delay, energy consumption, and partial charging (Huo, He, Xiong, & Wu, 2024). Similarly, Xin et al. (2024) formulated the dynamic task allocation problem as a vehicle routing problem (VRP) with real-time task arrivals, introducing a rolling horizon strategy integrated with ALNS and the Kuhn–Munkres algorithm to enable periodic task reallocation. Furthermore, the routing of AGVs integrated with order batching has been explored using a two-commodity network flow formulation (Xie, Li, & Luttmann, 2023). Recently, reinforcement learning has been applied to route AGVs in parts-to-pick warehouse scenarios (Li, Liu et al., 2024).

Container terminals are also places where AGVs are used to automate operations to boost the productivity of global trade logistics. In container terminals, AGVs are operated with other types of equipment, such as quay and yard cranes, and therefore, the routing of AGVs is integrated with the scheduling of these machines. Yang, Zhong, Dessouky, and Postolache (2018) set up a bi-level programming model for quay cranes, AGVs, and stacking cranes to coordinate the tasks and AGVs' paths. The coordination between the AGVs and quay cranes or yard cranes is investigated, especially to optimize the operation speed and reduce energy consumption for greener and sustainable terminal operations (Ma, Yu, Xie, & Yang, 2024; Xin, Meng, D'Ariano, Wang, & Negenborn, 2022). Moreover, the integrated scheduling of these coordinated machines can be decomposed using ADMM using a multi-robot task allocation framework (Chen et al., 2020). Besides these methods using mathematical programming, the coordinated scheduling is carried out by the digital twin-based support framework, which contains a mathematical programming model and Q-learning algorithm to deal with the operations of the complex operating environment (Gao, Chang, Chen, & Sha, 2024; Li, Fan et al., 2024).

2.3. EVRP

Another stream of research focuses on the EVRP, a variant of the traditional VRP that specifically utilizes a fleet of electric vehicles rather than internal combustion engine vehicles. For instance, Conrad and Figliozzi (2011) investigated the EVRP, allowing electric vehicles to recharge either fully or up to 80% of their battery capacity at selected customer locations during service, and an iterative construction and improvement heuristic was proposed. Schneider, Stenger, and Goeke (2014) expanded on this by introducing time windows and developing a foundational model where electric vehicles can recharge at any battery level using a linear charging function, with vehicles always leaving stations fully charged. This was solved using a metaheuristic that combines variable neighborhood search with tabu search. Keskin and Çatay (2018) further examined this problem, focusing on time windows and various recharging configurations. Cortés-Murcia, Prodhon, and Murat Afsar (2019) handled an EVRP variant with time windows, partial recharging, and satellite customers using iterated local search. Dönmez, Çağrı Koç, and Altıparmak (2022) studied the mixed fleet VRP with time windows and partial recharging, developing an MIP model and ALNS. Cai, Wu, and Fang (2024) proposed a double-assistant evolutionary multitasking algorithm for electric vehicle routing with backup batteries and swapping stations. Wang, Hou, and Guo (2024) focused on inventory management for battery swapping and charging stations under demand uncertainty. Qian, Feng, Yu, Hu, and Chen (2024) introduced a method for EVRP with time windows and battery swapping stations.

2.4. Summary

As a summarizing remark, the routing of AGVs for manufacturing and logistics applications remains an active research area. A comparison for the main related works on AGV routing with this study is summarized in Table 1. It can be observed that most existing literatures ideally assume that AGVs recharge after completing all tasks and overlooks the impact of energy constraints (Huo et al., 2024; Li et al., 2022; Zou et al., 2023), resulting in low AGV utilization and reduced productivity. In contrast, integrating fast charging operations within the production cycle represents an emerging research direction that is crucial for non-stop production. However, a comprehensive mathematical framework that explicitly incorporates fast charging during production remains underexplored. Consequently, addressing this gap is crucial for maintaining continuous AGV operation and ensuring high production efficiency in modern manufacturing systems. Furthermore, the development of more effective models and algorithms is required.

3. Problem description and formulation

This section describes the investigated multi-AGV routing integrated with fast charging and provides the mathematical model in detail.

3.1. Problem description

In modern matrix manufacturing workshops, workstations are arranged in a matrix format. The layout of the matrix manufacturing workshop studied in this paper is shown in Fig. 1, which comprises warehouses, workstations, charging stations, AGVs, pathways, and other key components. Once the workshop is fully operational, these components will work in coordination to ensure the efficient operation of the entire system. Specifically, each workstation consists of computer numerical control (CNC) machines and a material buffer. The material buffer is responsible for storing raw materials, while the CNC machine performs processing operations, consuming materials continuously throughout the process. When the material buffer runs low, the workstation sends a signal to replenish the materials in a timely manner. This moment is termed call time, and the corresponding workstation is defined as a customer. Additionally, the planned sequence of tasks assigned to each AGV is defined as a route. To ensure a consistent representation within the routing framework, the warehouse, customers, and charging stations are collectively categorized as tasks. However, due to the uncertainty in the call time of the workstations and the actual material delivery time, the material demand fluctuates dynamically, further increasing the complexity and uncertainty of the problem. AGVs load goods from the warehouse and shuttle between workstations in a predetermined order to ensure continuity and stability in production. After completing material delivery, they return to the warehouse, ready for the next transportation.

Given the numerous workstations in the workshop, multiple customers may be generated simultaneously. If each AGV is assigned to customers strictly in the order of their generation, the transport capacity of the AGVs cannot be fully utilized, resulting in inefficient

Table 1
Summary of related work on AGV routing.

Literatures	Recharging configuration		Policy	Objective function	Model	Algorithm
	Standard	Fast				
Farooq et al. (2021)			–	Completion time	MINLP	GA
Wang et al. (2022)			–	Makespan	MINLP	MSSA
Li et al. (2022)			–	Distance cost, penalty cost and AGV cost	MINLP	DIWO
Zou et al. (2023)	✓		CAD	Distance cost, penalty cost and AGV cost	MILP	SAIG
Xin et al. (2024)			–	Total completion time and makespan	MIP	ALNS-KM
Huo et al. (2024)	✓		CAD	Delay and energy consumption	MIP	NSGA-II
Xin et al. (2022)			–	Makespan and energy consumption	MINLP	GA
Chen et al. (2020)			–	Min total turn time	ILP	ADMM
This paper		✓	CDD	Distance cost, penalty cost and AGV cost	MILP	ALNS

CAD (Charging after delivery), CDD (Charging during delivery), Mixed-integer nonlinear programming (MINLP), GA (Genetic algorithm), MSSA (Multi-state scheduling algorithm), DIWO (Discrete invasive weed optimization), SAIG (Self-adaptive iterated greedy), ALNS-KM (Adaptive large neighborhood search-Kuhn–Munkres algorithm), NSGA-II (Non-dominated sorting GA), ADMM (Alternating direction method of multipliers-based dual decomposition).

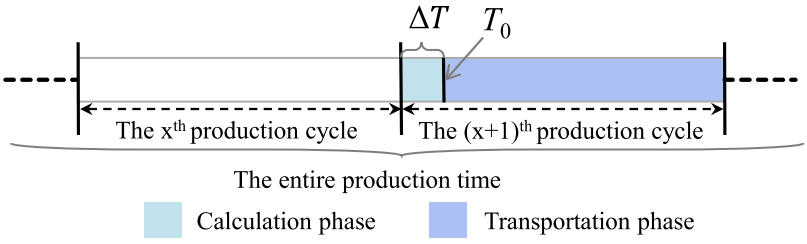

Fig. 2. Illustration of consecutive production cycles with the time-division strategy.

transportation. To address this issue, this study adopts a time-division strategy (Zou et al., 2020), as shown in Fig. 2, by dividing the entire production time into multiple consecutive production cycles. Each production cycle comprises two phases: a calculation phase and a transportation phase. During the calculation phase, the system schedules all customers generated in the previous production cycle and produces an optimal allocation plan. In the transportation phase, the AGVs perform material transport according to the routing plan. This strategy allows AGVs to serve multiple customers within the same production cycle, reducing the number of AGVs required in the workshop and lowering transportation costs.

However, as the AGVs operate, their battery power gradually depletes, which may lead to low battery issues. To ensure that AGVs can continuously transport materials over long periods without delays or interruptions due to insufficient power, the primary challenge in the multi-AGV routing problem becomes how to efficiently plan AGV charging behavior. This study proposes a flexible routing strategy based on fast charging to address the charging issue of AGVs during production. Before each AGV routing is determined, the control system assesses the AGV's battery level. If the remaining battery is predicted to fall below a predefined battery threshold, the system schedules a fast charge for the AGV. The timing of charging and the selection of the charging station are flexibly determined based on the customer sequence, aiming to optimize routing cost efficiency. This strategy ensures that AGVs can complete tasks efficiently while minimizing the impact of charging on task execution. The objective function of this study is to minimize transportation distance costs, early arrival penalty costs, and AGV usage costs.

To ensure the generalizability of this research on matrix manufacturing workshops, we propose the following basic assumptions:

- All equipment operates normally without breakdowns (Li, Li et al., 2018; Zou, Pan, & Wang, 2021);
- All AGVs are of the same specifications (Li, Li et al., 2018; Zou et al., 2021);
- The pathways are dual-lane, allowing AGVs to travel without collision risk during transportation (Zou et al., 2023);
- AGVs maintain a constant speed during transportation (Li, Li et al., 2018; Zou et al., 2021);
- The charging stations are equipped with enough charging piles to support multiple AGVs charging simultaneously.

The assumptions regarding equipment reliability, uniform AGV specifications, dual-lane pathways, and constant AGV speed are adopted from previous studies (Li, Li et al., 2018; Zou et al., 2023, 2021). Additionally, we introduce a novel parallel AGV recharge capacity assumption tailored to this study. Collectively, these assumptions establish a controlled yet realistic environment for modeling AGV operations in matrix manufacturing workshops.

3.2. Modeling

In order to address the aforementioned issues, we have proposed a mathematical model. The notation for the sets is provided in Table 2, while the indices and parameters utilized in the model are detailed in Table 3. Additionally, Table 4 presents the variables used in the model, categorized into decision variables and auxiliary variables.

The problem can be represented by an undirected graph $G = (V, E)$, where V is the set of vertices, and $E = \{(i, j) \mid i, j \in V, i \neq j\}$ is the set of edges, representing the feasible shortest paths between each pair of vertices. Let $C = \{1, 2, \ldots, n\}$ represent the customers, and $D = \{0, n+1, n+2, \ldots, n+n'\}$, where 0 represents the warehouse, and $n+1, n+2, \ldots, n+n'$ represent the charging stations. There are m AGVs available in the workshop, forming the AGV set $K = \{1, 2, \ldots, m\}$. The control system dispatches the optimal number of AGVs based on specific circumstances, and this number is always less than m. The transportation distance between two vertices is based on Manhattan distance, and the time required for an AGV to travel from customer i to customer j is calculated as $t_{ij} = d_{ij}/v$, where d_{ij} is the Manhattan distance and v is the AGV speed.

As the workstation submits a material request, its production operations continue uninterrupted, progressively depleting the buffer stock. To maintain efficient production in the manufacturing workshop, the AGV must accurately replenish the necessary materials based on the consumption level in the buffer at the time of its arrival. Considering the workstation's consumption dynamics and its stock level at the

Table 2
Notation for sets.

Set	Definition
C	Set of customers, and $C = \{1, 2, \dots, n\}$
D	Set of vertices excluding customers, and $D = \{0, n+1, n+2, \dots, n+n'\}$ where 0 represents the warehouse, and others represent charging stations
F	Set of charging stations, and $F = \{n+1, n+2, \dots, n+n'\}$
V	Set of all vertices, and $V = C \cup D$
K	Set of AGVs, and $K = \{1, 2, \dots, m\}$
E	Set of edges, and $E = \{(i, j) \mid i, j \in V, i \neq j\}$

Table 3
Notation for indices and parameters.

Index	Definition
i, j	Indices of tasks
k	Index of AGVs
Parameter	**Definition**
x_i	X-coordinate of task i
y_i	Y-coordinate of task i
n	Total number of customers
m	Number of available AGVs in the workshop
Q	Load capacity of the AGV
H	Total battery capacity of the AGV
N	Initial battery power of the AGV
r_c	Charging rate
r_d	Discharging rate
d_{ij}	Distance between customer i and customer j
t_{ij}	Time required for the AGV to travel from customer i to customer j
t_m	Consumption time for each material
t_u	Unloading time of materials at workstations
S	Capacity of the material buffer zone
S_i^c	Remaining capacity of the material buffer zone when workstation i calls
g	Weight of each unit of material
v	Speed of the AGVs
α	Battery threshold: The minimum required battery level (as a fraction of total battery capacity) upon arriving at the warehouse.
c_t	Weight coefficient for the early arrival penalty cost
c_d	Weight coefficient for transportation distance cost
c_a	Weight coefficient for AGV usage cost
T_i^l	Expiration time for customer i
T_i^c	Call time for customer i
ΔT	Duration of the calculation phase
T_0	Departure time of the AGV from the warehouse

Table 4
Notation for variables.

Decision Variable	Definition
x_{ijk}	Equals 1 if tasks i and j are executed consecutively by AGV k, and 0 otherwise (binary variable).
T_i^r	Delivery time for customer i (non-negative continuous variable).
Auxiliary Variable	**Definition**
z_{ik}	Equals 1 if task i is assigned to AGV k, and 0 otherwise (binary variable).
q_i	Demand quantity of customer i (non-negative continuous variable).
e_{ik}^r	Remaining battery power of AGV k upon arrival at customer i (non-negative continuous variable).
e_{ik}^l	Remaining battery power of AGV k upon departure from customer i (non-negative continuous variable).
c_k	Equals 1 if AGV k requires charging, and 0 otherwise (binary variable).
d_k^p	Predicted travel distance of AGV k (non-negative continuous variable).

moment of the request, the replenishment quantity can be determined using the following formula:

$$q_i = \left(\left(S - S_i^c\right) + \left(T_i^r - T_i^c\right)/t_m\right) \cdot g, \quad \forall i \in C \tag{1}$$

The total distance traveled by all AGVs can be expressed as:

$$C_d = \sum_{k=1}^{m} \sum_{j=0}^{n} \sum_{i=0}^{n} x_{ijk} d_{ij} \tag{2}$$

The total early arrival penalties for all customers can be expressed as:

$$C_t = \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{i=0}^{n} x_{ijk} \left(T_j^l - T_j^r\right) \tag{3}$$

The number of AGVs used can be expressed as:

$$C_a = \sum_{k=1}^{m} \sum_{j=1}^{n} x_{0jk} \tag{4}$$

The objective function of the mathematical model aims to minimize transportation costs. It consists of three elements: C_d (transportation distance cost), C_a (AGV usage cost), and C_t (early arrival penalty cost). The coefficients c_d, c_a, and c_t are weighting parameters that enable the model to adjust the relative importance of these cost elements based on specific operational requirements. The objective function is formulated as follows:

$$\min F = c_d C_d + c_a C_a + c_t C_t \tag{5}$$

To enhance clarity in our model, we divide the constraints into two categories: basic constraints, which establish the standard routing model, and energy constraints, which specifically address the battery limitations and fast charging operations of the AGVs.

Basic constraints:

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1, \quad \forall j \in C \tag{6}$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk} = 1, \quad \forall i \in C \tag{7}$$

$$\sum_{i\in V} x_{ijk} = \sum_{i\in V} x_{jik}, \quad \forall j \in V, \quad \forall k \in K \tag{8}$$

$$\sum_{i\in V} x_{i0k} = \sum_{j\in V} x_{0jk} \le 1, \quad \forall k \in K \tag{9}$$

$$\sum_{j=1}^{n}\sum_{i=0}^{n} x_{ijk}\cdot q_j \le Q, \quad \forall k \in K \tag{10}$$

$$d_k^{\mathrm{p}} = \sum_{i\in V}\sum_{j\in V} d_{ij}\cdot x_{ijk}, \quad \forall k \in K \tag{11}$$

$$x_{ijk}\cdot T_j^r = \begin{cases} (T_0 + t_{ij})\cdot x_{ijk}, & i = 0, \quad \forall j \in V, \quad \forall k \in K, \\ (T_i^r + tu + t_{ij})\cdot x_{ijk}, & \forall i \in C, \quad \forall j \in V, \quad \forall k \in K \\ (T_i^r + r_{\mathrm{c}}\cdot(H - e_{ik}^r) + t_{ij})\cdot x_{ijk}, & \forall i \in F, \quad \forall j \in V, \quad \forall k \in K \end{cases} \tag{12}$$

$$T_i^c \le T_i^r \le T_i^l, \quad \forall i \in V \tag{13}$$

$$z_{ik} = \sum_{j\in V} x_{ijk}, \quad \forall i \in V, \quad \forall k \in K \tag{14}$$

$$z_{ik} \in \{0,1\}, \quad \forall i \in V, \quad \forall k \in K \tag{15}$$

$$c_k \in \{0,1\}, \quad \forall k \in K \tag{16}$$

$$x_{ijk} \in \{0,1\}, \quad \forall i,j \in V, \quad \forall k \in K \tag{17}$$

$$x_{ijk} = 0, \quad \forall i,j \in V \quad \text{and} \quad i = j \tag{18}$$

$$T_i^r = T_0, \quad i = 0 \tag{19}$$

Constraints (6)–(8) ensure that each customer is served by exactly one AGV, maintaining consistent in-degree and out-degree for each customer. Constraint (9) ensures that each AGV is used at most once in a dispatch and that dispatched AGVs start from and return to the warehouse. Constraint (10) guarantees that the total load carried by an AGV does not exceed its maximum load capacity. Constraint (11) specifies the method for calculating the total distance traveled by each AGV along its assigned route. Constraint (12) defines the delivery time of the AGV at vertex j, considering its movement from different types of vertex i. Specifically, if i is the warehouse, the delivery time is determined by the departure time and travel time. When i is a customer, the delivery time is calculated by adding the unloading time and the travel time from i to j to the delivery time at i. Similarly, for a charging station, it is obtained as the sum of the delivery time at i, the charging time at i, and the travel time. Constraint (13) ensures compliance with time window restrictions. Finally, constraints (14)–(19) denote restrictions on the decision variables. After presenting the basic constraints, we then introduce the energy-related constraints.

Energy-related constraints:

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} \le m, \quad \forall i \in F \tag{20}$$

$$e_{0k}^r \ge \alpha\cdot H, \quad \forall k \in K \tag{21}$$

$$e_{ik}^l = \begin{cases} H, & \forall i \in F, \quad \forall k \in K \\ N, & i = 0, \quad \forall k \in K \\ e_{ik}^r, & \forall i \in C, \quad \forall k \in K \end{cases} \tag{22}$$

$$e_{jk}^r \le e_{ik}^l - r_{\mathrm{d}}\cdot d_{ij}\cdot x_{ijk} + H\cdot(1 - x_{ijk}), \quad \forall i,j \in V, \quad \forall k \in K \tag{23}$$

$$e_{jk}^r \ge e_{ik}^l - r_{\mathrm{d}}\cdot d_{ij}\cdot x_{ijk} - H\cdot(1 - x_{ijk}), \quad \forall i,j \in V, \quad \forall k \in K \tag{24}$$

$$c_k \ge (r_{\mathrm{d}}\cdot d_k^{\mathrm{p}} - (N - \alpha\cdot H))/H, \quad \forall k \in K \tag{25}$$

$$c_k \le (r_{\mathrm{d}}\cdot d_k^{\mathrm{p}} - (N - \alpha\cdot H) + H)/H, \quad \forall k \in K \tag{26}$$

$$c_k = \sum_{i\in V}\sum_{j\in F} x_{ijk}, \quad \forall k \in K \tag{27}$$

Constraint (20) states that all AGVs can access charging stations. Constraint (21) ensures that all AGVs arrive at the warehouse with a sufficient battery level to support subsequent tasks, where α denotes the battery threshold. Constraint (22) defines the battery level of the AGV upon exiting different types of vertices. Constraints (23) and (24) employ the big-M method to precisely track the AGV's battery level as it transitions between vertices. Constraints (25) and (26) utilize the big-M technique to accurately represent whether charging is required under specific conditions, with the binary variable c_k indicating the necessity of charging: $c_k = 1$ signifies that charging is needed, while $c_k = 0$ indicates otherwise. The big-M technique operates by imposing upper and lower bounds that link the AGV's battery level to the charging condition, ensuring the binary variable is activated only when charging is required. Building on this, Constraint (27) schedules a charging operation if required ($c_k = 1$); conversely, no charging operation is planned if not needed ($c_k = 0$).

In the model, constraints (10) and (12) are nonlinear, and we linearize them using the big-M method to formulate our problem as an MILP model. Moreover, similar to Zou et al. (2020), the AGV routing problem with integrated fast-charging operations can be viewed as a variant of the VRP, a well-known NP-hard problem. Its complexity is further compounded by task sequencing, time windows, demand uncertainties, and battery-charging strategies, with charging decisions exerting a critical influence on route feasibility and total transit time. Although commercial solvers such as Gurobi can efficiently solve small-scale instances, their computational effort grows exponentially with problem size, rendering them impractical for large-scale, real-world applications. To overcome these limitations, we propose an improved two-stage ALNS algorithm incorporating a heuristic charging-station insertion strategy.

4. Algorithm design

In this section, we detail the proposed ALNS algorithm customized for the AGV routing problem with integrated fast-charging operations. The key steps of the proposed ALNS algorithm are presented in Fig. 3, and a detailed description of the heuristic for the insertion of charging stations is provided in Section 4.3. The ALNS algorithm, originally proposed by Ropke and Pisinger (2006), is widely applied to solve VRP and EVRP. The core idea of ALNS is to explore new solutions by combining various destroy and repair operations, adaptively adjusting their selection probabilities based on the performance during the search process.

During the ALNS search process, any alteration in customer visit sequences can influence charging decisions. To enhance search efficiency, we divide the neighborhood search operation of ALNS into two

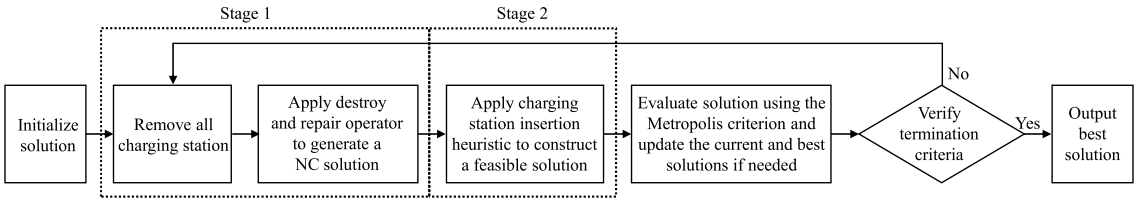

Fig. 3. Flowchart of the Proposed ALNS Algorithm.

Algorithm 1: The main procedure of the Improved ALNS heuristic algorithm with SA

Input: A set of destroy and repair operators (D and R)
Output: The best solution Ψ_{best}

Apply the constructive heuristic to generate an initial solution Ψ_{current};
$\Psi_{\text{best}} \leftarrow \Psi_{\text{current}}$;
Initialize selection probabilities P_d^t for each destroy operator $d \in D$ and P_r^t for each repair operator $r \in R$ in the iteration round t;
Let h be the cooling rate and T be the temperature;
while *termination conditions are not met* **do**
  Remove all charging stations;
  Select a destroy operator d and obtain a partial solution ϕ^*_{new};
  Choose a repair operator r to generate a new NC solution ϕ_{new};
  Apply the charging station insertion heuristic to construct a feasible solution Ψ_{new};
  if $c(\Psi_{new}) < c(\Psi_{current})$ **then**
    $\Psi_{\text{current}} \leftarrow \Psi_{\text{new}}$;
  else
    Let $\nu \leftarrow \exp\left((c(\Psi_{\text{current}}) - c(\Psi_{\text{new}}))/T\right)$;
    Generate a random number $\delta \in [0, 1]$;
    if $\delta < \nu$ **then**
      $\Psi_{\text{current}} \leftarrow \Psi_{\text{new}}$;
    end
  end
  if $c(\Psi_{current}) < c(\Psi_{best})$ **then**
    $\Psi_{\text{best}} \leftarrow \Psi_{\text{current}}$;
  end
  $T \leftarrow h \cdot T$;
  Update the selection probabilities for the operators;
end
return Ψ_{best};

stages. Each neighborhood search operation follows a "search first, insert later" principle, where neighborhood operations modify non-charging (NC) solutions that exclude charging stations, followed by the insertion heuristic to convert it into a feasible solution. Specifically, in the first stage, destroy and repair operations are applied to NC solutions that satisfy capacity and time window constraints without considering battery constraints. In the second stage, when an NC solution is generated after the destroy and repair operations, a charging station insertion heuristic is applied to add charging stations to the NC routes, ensuring the feasibility of the solution. The final feasible solution is accepted under simulated annealing (SA) criterion and is iteratively optimized within the algorithm framework. The algorithm framework is outlined in Algorithm 1.

4.1. Solution representation

In this study, to effectively address the AGV routing problem with charging considerations, we adopt a solution representation based on task permutations, similar to the classical VRP. Each task is represented by a unique identifier, where the warehouse is denoted by D_0, customers by C, and charging stations by S. For example, C_1 represents customer 1, and S_1 represents charging station 1. The solution is represented as a two-dimensional list, where each list corresponds to a complete route of an AGV. In each route, the AGV departs from the warehouse, visits the tasks in sequence to complete the respective tasks, and finally returns to the warehouse. For instance, if five workstations issue replenishment requests, requiring two AGVs, the task sequence for the first AGV could be $[D_0, S_2, C_2, C_5, C_3, D_0]$, and the sequence for the second AGV could be $[D_0, C_1, C_4, D_0]$. Thus, the solution can be expressed as $[[D_0, S_2, C_2, C_5, C_3, D_0], [D_0, C_1, C_4, D_0]]$. The routes of the two AGVs are shown in Fig. 1.

4.2. Initial solution construction

As ALNS commences from a single initial solution, its performance critically depends on the quality of the initial solution. Accordingly, we design a two-phase constructive heuristic algorithm to generate a high-quality initial solution.

In the first phase, we adopt an extended nearest neighbor heuristic (ENNH). Let $V = \{1, \ldots, n\}$ represent the set of unassigned customers. Each AGV departs from the warehouse, and at each iteration, it identifies the three nearest unserved customers relative to its current location. Among these candidates, the customer with the earliest expiration time is selected as the next successor. If adding the customer maintains compliance with both the time window and capacity constraints, the current route is updated. Otherwise, the warehouse is appended, signaling the termination of the current route and initiating the construction of a new route. This iterative process continues until all customers are assigned. Algorithm 2 presents the pseudocode for the ENNH.

Since the routes constructed in the first phase do not consider battery constraints, we introduce a charging station insertion heuristic

Algorithm 2: The extended nearest neighborhood heuristic

Input: the set of unassigned customers $V_{\text{unvisited}}$
Output: An NC solution ϕ_0
Initialize an NC solution ϕ_0 and $j = D_0$;
while $V_{unvisited}$ *is not empty* **do**
  Create a new route γ starting from the j;
  while $V_{unvisited}$ *is not empty* **do**
    Find the three nearest customers to j in $V_{\text{unvisited}}$;
    Choose the customer i with the earliest expiration time;
    Add the customer i into γ;
    if *the route* γ *is NC feasible* **then**
      Let $j = i$ and remove customer i from $V_{\text{unvisited}}$;
    else
      Remove the last added customer from γ;
      Add D_0 at the end of γ and set $j = D_0$;
    end
  end
  if *the route* γ *contains at least one customer* **then**
    Add the route γ to ϕ_0;
  end
end
return ϕ_0;

in the second phase to ensure solution feasibility. Specifically, when the AGV's battery level is insufficient during a route, the heuristic inserts an appropriate charging station to ensure that the final route complies with battery limitations and routing requirements.

4.3. Charging station insertion heuristic

The main steps of the charging station insertion heuristic are outlined in Algorithm 3. Given an NC solution, each route in the solution is sequentially assessed to determine whether charging is required. Routes that meet energy constraints are immediately deemed feasible. Otherwise, if charging is necessary, the route is adjusted by inserting a charging station.

For each route requiring adjustment, the feasible insertions of charging stations are evaluated. An insertion is deemed feasible only if the remaining energy level allows the visit to the charging station and the resulting route satisfies both the time window and capacity constraints. After evaluating all feasible insertions, the insertion with the minimum total cost is selected to ensure the route remains optimal in terms of efficiency. If no viable insertion is available, the customer with the earliest expiration time is removed and added to the removal list. The updated route is then re-evaluated for charging requirements. This process is repeated iteratively until all routes meet the feasibility criteria, either by not requiring charging or by successfully incorporating charging stations.

Finally, once all routes in the current solution have become feasible, the customers in the removal list are addressed. While the removal list is not empty, the ENNH is applied to these customers, generating a new NC solution that may comprise multiple routes. This new NC solution is then combined with the previous solution. Subsequently, the algorithm returns to step 1 to process the updated solution through the aforementioned operations to ensure feasibility. This iterative procedure continues to refine the solution until the removal list is empty at step 14, ultimately yielding a feasible final solution.

4.4. Destroy and repair operators

An initial feasible solution is first given, followed by the generation of NC solutions through the removal of charging stations. Neighborhood search is then performed on the NC solutions using destroy and repair operators. Previous studies on routing problems have introduced a variety of destroy and repair operators (Keskin & Çatay, 2018; Ropke & Pisinger, 2006). Based on the unique characteristics of this problem, we selected 10 effective destroy operators and 5 repair operators specifically tailored to these requirements. Finally, a charging station insertion heuristic is applied to incorporate charging stations into the NC solutions, thereby resulting in feasible solutions.

4.4.1. Destroy operators

The destruction mechanism within the ALNS framework comprises multiple destroy operators, which can be classified into two categories: route destroy and customer destroy. In the route destroy operation, an AGV route is selected, and all customers within that route are removed and added to the unassigned list ζ. In customer destroy, a subset of n customers is selected and added to the unassigned list, where the value of n is determined based on the total number of customers. In this study, we employ several well-known destroy operators including Random Route, Shortest Route, Random, Worst-Distance, Worst-Time, Shaw, Proximity-based, Time-based, and Zone (Keskin & Çatay, 2018), and we also propose the Fewest Customer Route Destroy. Detailed descriptions of all destroy operators are provided below.

Random Route Destroy In this operator, a route is randomly selected from the current solution and deleted, with all its customers added to the unassigned list, denoted as ζ. This random destroy strategy helps to expand the search space, thereby increasing the chances of finding the global optimum. (Keskin & Çatay, 2018)

Shortest Route Destroy This operator removes the route with the shortest distance from the current solution and deletes all customers on that route, thereby helping to minimize the risk of converging to a local optimum (Keskin & Çatay, 2018).

Fewest Customer Route Destroy This operation selects and removes the route containing the fewest customers, aiming to merge routes and reduce the number of AGVs required.

Random Customer Destroy A set of n customers is randomly removed from various routes in the current solution. This random selection aims to diversify the search and expand the solution space (Keskin & Çatay, 2018; Ropke & Pisinger, 2006).

Worst-Distance Customer Destroy Customers with the highest total distances are selected and removed one by one from the current solution. The distance of a customer is defined as the sum of the distances to its preceding and succeeding customers in the route (Keskin & Çatay, 2018; Ropke & Pisinger, 2006).

Algorithm 3: The procedure of the charging station insertion heuristic

Input: An infeasible NC solution ϕ_0
Output: A feasible solution Ψ_{fea}
Let $V_{\text{unvisited}} \leftarrow \emptyset$;
foreach *AGV route* γ *in* ϕ_0 **do**
  while *route* γ *requires charging* **do**
    Find feasible insertions for charging stations;
    if *no insertion point exists* **then**
      Remove the customer with the earliest expiration time;
      Add the removed customer to $V_{\text{unvisited}}$;
    else
      Select the insertion with the minimum cost;
      Generate a feasible route γ_{fea};
      $\gamma \leftarrow \gamma_{\text{fea}}$;
    end
  end
end
while $V_{unvisited} \neq \emptyset$ **do**
  Apply the ENNH for $V_{\text{unvisited}}$ and update ϕ_0;
  Return to step 1;
end
return Ψ_{fea};

Worst-Time Customer Destroy This operator calculates the difference between the delivery time and the expiration time for each customer i, then removes the customer with the largest difference. The goal is to maximize the number of customers whose time windows are satisfied (Keskin & Çatay, 2018).

Shaw Destroy The Shaw Destroy operation involves removing a group of customers based on specific criteria. Initially, the operator randomly selects and removes a customer i. Then, it selects the customer $j = \arg\min(\Phi_1 \cdot d_{ij} + \Phi_2 \cdot |T_j^l - T_i^l| + \Phi_3 \cdot l_{ij} + \Phi_4 \cdot |q_j - q_i|)$, where Φ_1 to Φ_4 are the weights of different parameters. Here, d_{ij} represents the distance between customers i and j, T^l denotes the expiration time, and q indicates the material demand (Keskin & Çatay, 2018; Ropke & Pisinger, 2006).

Proximity-based Customer Destroy In this operator, the first customer is selected randomly, and the next customer is chosen based on proximity to the previously selected one. At each step, the customer closest to the last selected customer is added to the unassigned list. This process continues until n customers are selected and removed sequentially (Keskin & Çatay, 2018).

Time-based Customer Destroy Since the expiration times are determined by the call times, all customers will have identical time windows in terms of length. Therefore, similar to the Proximity-based Customer Destroy, this operator removes n customers based on their similar call times (Keskin & Çatay, 2018).

Zone Destroy This operation begins by randomly selecting a region of predefined size in the Cartesian coordinate system. A random set of n customers within this region is then chosen for removal. If fewer than n customers are found, another region of the same size is selected, and the process is repeated until n customers are removed (Keskin & Çatay, 2018).

4.4.2. Repair operators

In the algorithm, five repair operators are introduced to reinsert removed customers into the routes, thereby generating an NC solution. Specifically, following the approach in Keskin and Çatay (2018), we adopt the Greedy, Time-based, Regret-2, and Regret-3 repair operators, and we additionally introduce a Random repair operator. During this insertion process, only time window and capacity constraints are considered, while battery capacity is not taken into account. Furthermore, it is important to note that since destroy operations may reduce the actual number of AGVs dispatched below the minimum required, if no feasible insertion is available for any customer during the repair process, a new route will be initialized to accommodate the customer and facilitate the insertion of any remaining customers.

Greedy Repair The operator evaluates the insertion cost for each unassigned customer at the most appropriate position within the current solution. It then iteratively inserts the customer with the minimum insertion cost into a feasible position. The insertion cost is defined as the increase in total travel distance caused by adding the customer (Keskin & Çatay, 2018; Ropke & Pisinger, 2006).

Time-based Repair The Time-based Repair, based on the Greedy Repair method, primarily differs in how insertion cost is defined. Instead of measuring the increase in travel distance, it calculates the cost as the change in route completion time after adding a customer. The objective is to minimize the increase in completion time for each route (Keskin & Çatay, 2018).

Random Repair The Random Repair operator inserts unassigned customers into existing routes by randomly selecting both the route and the insertion position. This operator introduces stochasticity to increase solution diversity and help avoid local optima.

Regret-2 Repair Let Δf_i denote the change in the objective function value after inserting customer i into the current solution. The customer to be inserted, i, is selected as: $i = \arg\max_{i \in \zeta}(\Delta f_{i2} - \Delta f_{i1})$, where Δf_{i1} represents the increase in the objective function value after inserting customer i at the best feasible position, and Δf_{i2} is the increase after inserting the same customer at the second-best feasible position (Keskin & Çatay, 2018; Ropke & Pisinger, 2006).

Regret-3 Repair Similar to the Regret-2 Repair, the Regret-3 Repair compares the best and third-best feasible positions. Specifically, The chosen customer i is determined as: $i = \arg\max_{i \in \zeta}(\Delta f_{i3} - \Delta f_{i1})$, where Δf_{i1} denotes the change from inserting customer i in the best feasible position, and Δf_{i3} represents the change from inserting the same customer in the third-best position (Ropke & Pisinger, 2006).

4.5. Roulette wheel selection and operator weight update mechanism

This study employs a roulette wheel mechanism to probabilistically select destroy and repair operators during the search process. Initially, the weight of each operator is uniformly set to 1. Assume there are k destroy operators and m repair operators. For a destroy operator j, its weight is denoted as ω_j^d, and the probability of its selection under the roulette wheel mechanism is calculated as $p_j^d = \omega_j^d / \left(\sum_{i=1}^{k} \omega_i^d\right), j \in$

$\{1, 2, \ldots, k\}$ Similarly, for a repair operator j, its weight is denoted as ω_j^r, and the probability of its selection is given by $p_j^r = \omega_j^r / \left(\sum_{i=1}^{m} \omega_i^r\right), j \in \{1, 2, \ldots, m\}$.

During the search, the weights of selected operators are dynamically updated based on their performance. The weight update rules are defined as follows:

$$\omega_d^{t+1} = \theta \omega_d^t + (1 - \theta) s_j \tag{28}$$

$$\omega_r^{t+1} = \theta \omega_r^t + (1 - \theta) s_j \tag{29}$$

Here, ω_d^{t+1} and ω_r^{t+1} represent the weights of the selected destroy and repair operators at iteration $t + 1$, respectively. The parameter θ serves as a decay factor controlling the influence of previous iterations on the current weight update, and s_j is the score assigned to operator j in iteration t, quantifying its effectiveness based on the solution's performance. Higher scores correspond to more effective operators. The scores are assigned according to the following criteria: when a new global best solution is found, $s_j = \mu_1$; if the solution improves upon the current solution but does not surpass the global best, $s_j = \mu_2$; if the solution is worse than the current one but is accepted under the SA criterion, $s_j = \mu_3$; and if the solution is rejected, $s_j = \mu_4$. This adaptive mechanism guarantees that the algorithm remains flexible by gradually prioritizing operators with better performance, thereby enhancing the overall search process.

4.6. Acceptance criterion

This study adopts an SA mechanism as the acceptance criterion in the ALNS algorithm to balance exploration and exploitation. SA, a probabilistic optimization technique, prevents the algorithm from becoming trapped in local optima and facilitates its progression toward the global optimum. The acceptance of new solutions is regulated by a "temperature" parameter that diminishes over time. Initially, at higher temperatures, the algorithm can accept suboptimal solutions with a certain probability, encouraging exploration and reducing the likelihood of stagnation in suboptimal regions. As the temperature declines, the probability of accepting worse solutions decreases, steering the algorithm toward convergence. In each iteration, ALNS generates a new solution through destroy-and-repair operations, and the acceptance decision is based on the SA criterion. If the new solution enhances the objective value, it is directly accepted; otherwise, it is accepted with a probability calculated as:

$$P = \exp\left(\left(f_{\text{current}} - f_{\text{new}}\right) / T\right) \tag{30}$$

Here, f_{current} and f_{new} denote the objective values of the current solution and new solution, respectively, while T denotes the current temperature. This mechanism allows the algorithm to accept suboptimal solutions during the early stages of the search process, promoting broader exploration of the solution space. As the temperature declines, the acceptance criteria become stricter, focusing on solution refinement and guiding the search toward optimality.

5. Computational experiments

In this section, comprehensive statistical experiments are conducted to assess the effectiveness of the proposed ALNS algorithm. First, the test methods, analysis methods, and data settings for the experiments are described. Next, the experimental environment is presented. Subsequently, the relevant parameters are selected. Then, we compare the computational results of the proposed algorithm with those of existing heuristic rules and metaheuristics. Finally, sensitivity analysis experiments on large-scale instances are conducted, and managerial insights are provided.

Table 5
Parameter settings.

Items	Values	Items	Values
ΔT	10 s	t_m	30 s/slice
T_0	350 s	t_u	15 s
Q	250 kg	g	0.75 kg/slice
v	1 m/s	S	48 slice
H	72 Wh	c_d	1
N	33.6 Wh	c_t	0.1
r_d	0.048 Wh/m	c_a	1000
r_c	0.48 Wh/s	α	0.2

5.1. Experimental setup and methods

The instances used in this study are derived from Foxconn Technology Group, a leading electronics manufacturing company. The experiment utilizes 110 instances of varying sizes, specifically 7, 10, 20, 30, 40, and 50 customers. For sizes 10, 20, 30, 40, and 50, each size includes 20 instances, while size 7 includes 10 instances. Instances are labeled with "T" followed by their size and index. For example, T30I6 denotes the 6th instance with 30 customers.

Each task in the instance includes detailed information such as the task type, positional coordinates, stock level in the buffer at the time of the call, call time, expiration time, and service duration. The task type is categorized as customer (c), warehouse (d), or charging station (f). For example, the task C_1 with detailed information {c, 8, 16, 30, 14, 674, 15} can be interpreted as follows: c identifies it as a customer task, 8 and 16 represent the x and y coordinates, 30 indicates the stock level in the buffer at the time of the call, 14 is the call time, 674 denotes the expiration time, and 15 is the service duration. Additionally, the parameter settings are summarized in Table 5.

The multi-AGV routing integrated with fast charging is a novel problem involving the routing of AGVs in matrix manufacturing workshops. To evaluate performance, the Gurobi solver and three algorithms tailored to this problem were selected for comparison. The Gurobi solver was used to validate the effectiveness of the proposed model. As a heuristic method, First-Come-First-Served (FCFS) is widely used in matrix manufacturing workshops, making it a suitable choice for comparison. Additionally, the variable neighborhood search (VNS) algorithm and the SA algorithm were included in the comparison.

The performance of all methods was evaluated using the objective function F. Each algorithm was executed 10 times for all 110 instances. For the Gurobi solver, obtaining a solution within a reasonable computation time is challenging unless the instance size is very small. Therefore, a maximum computation time of 600 s was set.

In real-world production, AGV control systems must generate routing solutions within a limited time frame to prevent production disruptions. Unlike previous studies that often use the number of iterations as a stopping criterion, this study imposes a CPU runtime limit of ΔT = 10 s. This runtime limit ensures efficient solution generation, maintaining the continuity of production processes.

5.2. Experimental environment

All algorithms were implemented using Python 3.11, with the development environment running on a Windows 10 (64-bit) operating system. The computational experiments were conducted on a workstation equipped with an Intel Core i7-11700 processor, operating at a base clock speed of 2.5 GHz, alongside 16 GB of RAM. This configuration was selected to ensure efficient processing capabilities for the algorithms, particularly in handling the computational workload associated with the problem-solving process. The selected hardware and software environment offers a reliable platform for evaluating the performance and scalability of the proposed methods.

Table 6
Parameter levels for ALNS.

Parameter level	1	2	3	4
T_0	100	200	300	400
h	0.8	0.9	0.999	0.99999
θ	0.1	0.4	0.6	0.9
$[\mu_1, \mu_2, \mu_3, \mu_4]$	[0.45,0.3,0.15,0]	[33,13,9,0]	[45,15,3,0]	[100,10,1,0]

Table 7
Orthogonal test scheme L_{16} and results of orthogonal experiment.

Exp	T_0	h	θ	$[\mu_1, \mu_2, \mu_3, \mu_4]$	S/N
1	100	0.8	0.1	[0.45,0.3,0.15,0]	−73.168
2	100	0.9	0.4	[33,13,9,0]	−73.138
3	100	0.999	0.6	[45,15,3,0]	−73.140
4	100	0.99999	0.9	[100,10,1,0]	−73.141
5	200	0.8	0.4	[45,15,3,0]	−73.138
6	200	0.9	0.1	[100,10,1,0]	−73.156
7	200	0.999	0.9	[0.45,0.3,0.15,0]	−73.150
8	200	0.99999	0.6	[33,13,9,0]	−73.153
9	300	0.8	0.6	[100,10,1,0]	−73.133
10	300	0.9	0.9	[45,15,3,0]	−73.128
11	300	0.999	0.1	[33,13,9,0]	−73.142
12	300	0.99999	0.4	[0.45,0.3,0.15,0]	−73.149
13	400	0.8	0.9	[33,13,9,0]	−73.130
14	400	0.9	0.6	[0.45,0.3,0.15,0]	−73.130
15	400	0.999	0.4	[100,10,1,0]	−73.143
16	400	0.99999	0.1	[45,15,3,0]	−73.146

5.3. Parameter selection

Optimizing parameter settings is essential for enhancing the performance of metaheuristic algorithms. In the proposed ALNS algorithm, four key parameters require calibration: the initial temperature (T_0), temperature cooling rate (h), decay rate (θ), and reward combinations ($[\mu_1, \mu_2, \mu_3, \mu_4]$). Taguchi's method (Leon, Shoemaker, & Kacker, 1987; Zhou & Lee, 2020) was employed to determine appropriate values for these parameters. Each parameter was evaluated across four predefined levels to identify suitable values for subsequent experiments. The selected parameter ranges are detailed in Table 6.

This study evaluates the influence of parameter settings using the objective function as a performance indicator, where smaller values indicate greater parameter efficiency. Consequently, we adopt the "smaller-the-better" principle to compute the signal-to-noise (S/N) ratio, defined as:

$$S/N = -10\log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}(F_i)^2\right) \tag{31}$$

where F_i represents the objective value from the ith experiment, and n denotes the total number of experiments.

To design the experiments, the L_{16} orthogonal array from the Taguchi framework was employed. Instance T30I10 was used for testing, with each parameter combination in the orthogonal array evaluated 20 times. The parameter settings, their corresponding S/N ratios, and the L_{16} orthogonal array are presented in Table 7.

The S/N results presented in Table 7 were analyzed to calculate the average response values for each parameter level, revealing their respective trends. These trends are illustrated in Fig. 4, which depicts the influence of each parameter on the S/N ratio. The steepness of the slopes in the figure reflects the relative significance of the parameters, with steeper slopes indicating a greater impact. Parameters associated with higher S/N ratios exhibit superior performance. Based on this analysis, the best parameter settings were determined as $T_0 = 400$, $h = 0.9$, $\theta = 0.9$, $[\mu_1, \mu_2, \mu_3, \mu_4] = [45, 15, 3, 0]$. To ensure a fair and reproducible comparison, we detail the parameter settings for the benchmark algorithms as follows. The computation time for SA, VNS, and ALNS was uniformly limited to 10 s. For the SA algorithm, we employed the same initial temperature (400) and cooling rate (0.9) as used in our ALNS framework. The VNS algorithm was implemented using 5 neighborhood operators, including 2-opt*, 2-opt, or-opt, cross exchange, and merge route, following standard practices in the literature. The FCFS, as a greedy heuristic, does not require adjustable parameters.

5.4. Comparison and analysis with other algorithms

5.4.1. Small-scale instance

In this section, small-scale instances with 7 and 10 customers are tested to evaluate the effectiveness of the proposed model. Tables 8 and 9 provide a comparison of the objective values and computation time (CT) for these instances. The metaheuristic algorithm is executed 10 times for each instance, with the average objective value and standard deviation calculated. Additionally, due to the extremely short computation time of FCFS for all instances, its computation time is denoted as "-". The relative performance difference between the ALNS algorithm and the benchmark method is quantified by Δ_{Obj}, calculated using the following formula:

$$\Delta_{\text{Obj}} = (\text{Obj}_{\text{ALNS}} - \text{Obj}_Z)/\text{Obj}_Z \tag{32}$$

where Obj_Z and Obj_{ALNS} represent the average objective values achieved by the benchmark method Z and ALNS, respectively. A negative Δ_{Obj} value indicates that the ALNS algorithm outperforms the benchmark by achieving a lower objective value.

As shown in Table 8, for small-scale instances with 7 customers, the average Δ_{Obj} values for Gurobi, FCFS, SA, and VNS are 0.0%, −5.4%, −0.2%, and −3.4%, respectively. In terms of average values and standard deviations, GUROBI and ALNS yield the best results, with SA closely approximating the optimal values across all 10 instances. For instances with 10 customers, Table 9 reports the average Δ_{Obj} values of −0.0%, −44.0%, −13.0%, and −6.1%, indicating that ALNS achieves the best performance in nearly all cases. Although ALNS does not yield the best solution for all 20 instances, it consistently generates solutions that are very close to the results of GUROBI solving for 600 s. Furthermore, the algorithm demonstrates remarkable stability across all instances. Overall, ALNS proves highly effective for solving this problem in the small-scale matrix workshop, consistently outperforming other algorithms in the comparison.

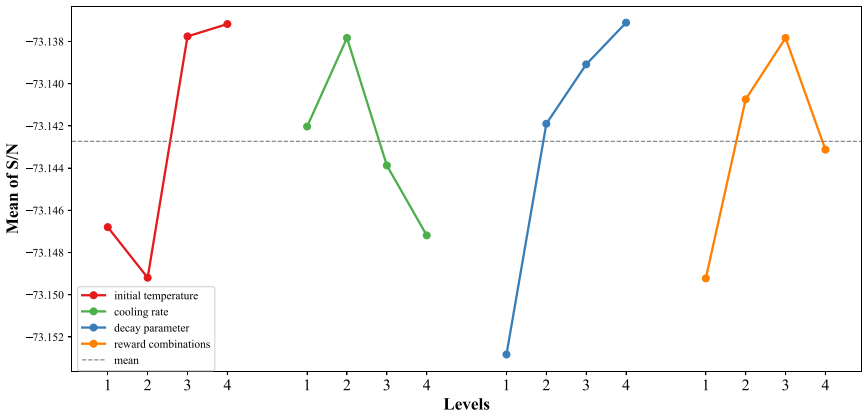

Fig. 4. The mean S/N plot for different levels of the parameters for T30I10.

Table 8
Experimental results for the instances with 7 customers.

Instance	Gurobi		FCFS		SA	VNS	ALNS	vs. Gurobi	vs. FCFS	vs. SA	vs. VNS
	Obj	CT	Obj	CT	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T7I1	**1459.9**	150	1643.2	–	1470.7 ± 9.3	1522.9 ± 12.5	**1459.9 ± 0.0**	0.0%	−11.2%	−0.7%	−4.1%
T7I2	**1457.7**	80	1588.8	–	1463.6 ± 7.6	1497.5 ± 16.1	**1457.7 ± 0.0**	0.0%	−8.1%	−0.3%	−2.5%
T7I3	**1429.9**	75	1492.9	–	**1429.9 ± 0.0**	1508.9 ± 20.8	**1429.9 ± 0.0**	0.0%	−4.2%	0.0%	−5.2%
T7I4	**1416.0**	60	1458.9	–	1421.0 ± 5.2	1472.4 ± 14.5	**1416.0 ± 0.0**	0.0%	−2.9%	−0.3%	−3.8%
T7I5	**1437.7**	170	1525.8	–	**1437.7 ± 0.0**	1495.1 ± 6.5	**1437.7 ± 0.0**	0.0%	−5.8%	0.0%	−3.8%
T7I6	**1448.2**	365	1517.5	–	**1448.2 ± 0.0**	1507.5 ± 13.2	**1448.2 ± 0.0**	0.0%	−4.6%	0.0%	−3.9%
T7I7	**1473.1**	115	1484.6	–	1475.0 ± 2.4	1491.3 ± 13.0	**1473.1 ± 0.0**	0.0%	−0.8%	−0.1%	−1.2%
T7I8	**1482.7**	251	1534.1	–	**1482.7 ± 0.0**	1521.2 ± 18.4	**1482.7 ± 0.0**	0.0%	−3.3%	0.0%	−2.5%
T7I9	**1451.9**	106	1576.6	–	**1451.9 ± 0.0**	1499.5 ± 11.3	**1451.9 ± 0.0**	0.0%	−7.9%	0.0%	−3.2%
T7I10	**1424.3**	106	1498.4	–	1429.3 ± 8.4	1474.3 ± 21.9	**1424.3 ± 0.0**	0.0%	−4.9%	−0.4%	−3.4%
Average	**1448.1**	147.8	1532.1	–	1451.1 ± 20.4	1501.3 ± 21.3	**1448.1 ± 20.3**	0.0%	−5.4%	0.2%	−3.4%

Note: The maximum computation time for SA, VNS, and ALNS is set to 10 s, and the computation time for FCFS is denoted by "–".

Table 9
Experimental results for the instances with 10 customers.

Instance	Gurobi		FCFS		SA	VNS	ALNS	vs. Gurobi	vs. FCFS	vs. SA	vs. VNS
	Obj	CT	Obj	CT	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T10I1	**1584.8**	600	2973.3	–	2751.1 ± 43.1	1653.7 ± 16.2	1586.2 ± **4.5**	0.1%	−46.7%	−42.3%	−4.1%
T10I2	**1580.2**	600	2817.4	–	1630.5 ± 90.1	1687.6 ± 41.2	**1580.2 ± 0.0**	0.0%	−43.4%	−2.2%	−5.6%
T10I3	**1548.2**	600	2942.9	–	1550.1 ± 6.2	1646.4 ± 15.3	**1548.2 ± 0.0**	0.0%	−47.4%	−0.1%	−6.0%
T10I4	**1522.5**	600	2774.7	–	2637.8 ± 25.8	1600.1 ± 17.7	1524.1 ± **2.0**	0.1%	−45.1%	−42.2%	−4.8%
T10I5	**1592.4**	600	2943.7	–	2780.9 ± 73.4	1783.3 ± 359.1	1597.5 ± **5.9**	0.3%	−45.7%	−42.6%	−10.4%
T10I6	**1583.5**	600	1604.8	–	1586.9 ± 5.5	1608.7 ± 7.4	**1583.5 ± 0.0**	0.0%	−1.3%	−0.2%	−1.6%
T10I7	**1581.9**	600	2964.2	–	1595.8 ± 14.1	1682.8 ± 34.2	**1581.9 ± 0.0**	0.0%	−46.6%	−0.9%	−6.0%
T10I8	**1579.7**	600	2849.8	–	2650.2 ± 19.8	1778.1 ± 352.6	1582.3 ± **4.1**	0.2%	−44.5%	−40.3%	−11.0%
T10I9	**1553.5**	600	2857.1	–	**1553.4 ± 0.0**	1664.6 ± 17.9	**1553.4 ± 0.0**	0.0%	−45.6%	0.0%	−6.7%
T10I10	**1554.1**	600	2909.0	–	1570.4 ± 13.4	1662.2 ± 33.2	**1554.1 ± 0.0**	0.0%	−46.6%	−1.0%	−6.5%
T10I11	**1543.1**	600	2721.9	–	1558.8 ± 15.1	1639.3 ± 40.0	**1543.1 ± 0.0**	0.0%	−43.3%	−1.0%	−5.9%
T10I12	**1551.1**	600	2790.2	–	**1551.1 ± 0.0**	1611.8 ± 6.5	**1551.1 ± 0.0**	0.0%	−44.4%	0.0%	−3.8%
T10I13	**1540.7**	600	2795.3	–	2631.3 ± 21.0	1640.6 ± 15.9	**1540.7 ± 0.0**	0.0%	−44.9%	−41.4%	−6.1%
T10I14	**1560.1**	600	2923.5	–	1569.0 ± 14.5	1646.8 ± 19.2	1560.6 ± **1.6**	0.0%	−46.6%	−0.5%	−5.2%
T10I15	**1538.6**	600	2784.6	–	1546.7 ± 8.8	1631.2 ± 18.6	**1538.6 ± 0.0**	0.0%	−44.7%	−0.5%	−5.7%
T10I16	**1549.9**	600	2907.7	–	1557.0 ± 7.7	1639.4 ± 16.0	**1549.9 ± 0.0**	0.0%	−46.7%	−0.5%	−5.5%
T10I17	**1535.9**	600	3908.7	–	1546.3 ± 11.2	1649.1 ± 26.2	**1535.8 ± 0.0**	0.0%	−60.7%	−0.7%	−6.9%
T10I18	**1556.3**	600	2901.5	–	1578.0 ± 20.8	1739.9 ± 58.3	**1556.3 ± 0.0**	0.0%	−46.4%	−1.4%	−10.6%
T10I19	**1537.2**	600	2880.9	–	1541.1 ± 5.2	1641.9 ± 17.4	**1537.2 ± 0.0**	0.0%	−46.6%	−0.3%	−6.4%
T10I20	**1549.8**	600	2747.9	–	2667.6 ± 24.4	1597.6 ± 30.6	**1549.8 ± 0.0**	0.0%	−43.6%	−41.9%	−3.0%
Average	**1549.8**	600	2850.0	–	1899.6 ± 510.0	1667.4 ± 145.4	1558.7 ± **21.4**	0.0%	−44.0%	−13.0%	−6.1%

Note: The maximum computation time for SA, VNS, and ALNS is set to 10 s, and the computation time for FCFS is denoted by "–".

Table 10
Experimental results for the instances with 20 customers.

Instance	FCFS	SA	VNS	ALNS	vs. FCFS	vs. SA	vs. VNS
	Obj	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T20I1	4639.4	3110.1 ± 30.1	3189.1 ± 27.4	**3055.5 ± 0.7**	−34.1%	−1.8%	−4.2%
T20I2	6157.4	3115.9 ± 22.0	3260.1 ± 31.8	**3076.3 ± 6.7**	−50.0%	−1.3%	−5.6%
T20I3	4510.2	3093.6 ± 49.1	3196.2 ± 29.4	**3047.4 ± 6.3**	−32.4%	−1.5%	−4.7%
T20I4	4645.2	4243.3 ± 54.1	3243.3 ± 30.6	**3072.4 ± 7.1**	−33.9%	−27.6%	−5.3%
T20I5	5881.6	3129.9 ± 19.3	3222.3 ± 35.1	**3088.3 ± 2.0**	−47.5%	−1.3%	−4.2%
T20I6	4588.4	3192.3 ± 57.6	3271.3 ± 48.5	**3095.0 ± 4.8**	−32.5%	−3.0%	−5.4%
T20I7	4787.4	3161.0 ± 27.8	3254.4 ± 28.2	**3088.1 ± 1.4**	−35.5%	−2.3%	−5.1%
T20I8	4920.1	3130.0 ± 23.6	3200.0 ± 25.5	**3073.1 ± 8.4**	−37.5%	−1.8%	−4.0%
T20I9	4599.2	3093.6 ± 15.5	3161.9 ± 18.0	**3055.7 ± 9.3**	−33.6%	−1.2%	−3.4%
T20I10	4639.3	3073.7 ± 20.6	3181.7 ± 49.2	**3038.2 ± 3.2**	−34.5%	−1.2%	−4.5%
T20I11	4747.8	3149.8 ± 40.8	3265.7 ± 46.9	**3072.9 ± 5.7**	−35.3%	−2.4%	−5.9%
T20I12	4690.4	3097.3 ± 16.9	3190.7 ± 15.8	**3078.5 ± 3.9**	−34.4%	−0.6%	−3.5%
T20I13	4631.2	3102.8 ± 43.1	3203.0 ± 29.0	**3045.0 ± 4.5**	−34.3%	−1.9%	−4.9%
T20I14	4822.8	3137.2 ± 44.2	3258.0 ± 16.1	**3058.4 ± 10.6**	−36.6%	−2.5%	−6.1%
T20I15	4672.1	3108.6 ± 45.1	3218.1 ± 29.0	**3043.8 ± 0.0**	−34.9%	−2.1%	−5.4%
T20I16	4639.3	4296.6 ± 34.9	3209.5 ± 36.4	**3103.5 ± 4.0**	−33.1%	−27.8%	−3.3%
T20I17	4718.8	4317.4 ± 154.6	3216.4 ± 47.2	**3075.0 ± 2.1**	−34.8%	−28.8%	−4.4%
T20I18	4622.3	3082.6 ± 20.2	3173.4 ± **1.6**	**3051.8** ± 7.5	−34.0%	−1.0%	−3.8%
T20I19	4878.0	3145.0 ± 26.3	3200.7 ± 26.9	**3072.1 ± 0.0**	−37.0%	−2.3%	−4.0%
T20I20	4461.4	3084.1 ± 34.6	3184.9 ± 24.4	**3037.7 ± 2.0**	−31.9%	−1.5%	−4.6%
Average	4812.6	3293.2 ± 421.7	3215.0 ± 44.5	**3066.4 ± 19.6**	−35.9%	−5.7%	−4.6%

5.4.2. Large-scale instance

Tables 10–13 present a comparative analysis of the computational performance of ALNS, FCFS, VNS, and SA on large-scale instances with 20, 30, 40, and 50 customers, respectively, to evaluate the proposed ALNS algorithm. Gurobi is excluded from the comparison due to preliminary tests demonstrating its inability to solve large-scale instances within reasonable computation times. Additionally, since the computation time for FCFS is extremely short and the metaheuristic methods are executed with a fixed runtime, CT is no longer used as a performance metric. The metaheuristic algorithm was run 10 times for each instance, and the corresponding average objective values and standard deviations were computed. These tables also present the average objective values for all methods across different instance sizes at the bottom. To visually illustrate the differences in objective values among these metaheuristics, Fig. 5 depicts the mean, maximum, and minimum objective values for each instance of each table.

A comparison of Tables 10–13 highlights the superiority of the proposed ALNS method over the other three approaches. In terms of objective values, ALNS consistently achieves the best results across all instances, underscoring its effectiveness in solving the problem. Stability is also a critical factor in all scenarios, and ALNS demonstrates the smallest standard deviations in every case, showcasing its remarkable stability and reliability in providing high-quality solutions. Additionally, the FCFS method, currently used in factories and based on call time for AGV routing, yields suboptimal results. Its computational inefficiency further confirms that FCFS is unsuitable for addressing this problem. Overall, ALNS consistently achieves strong performance across all instances. Thus, ALNS remains the best-performing method.

Fig. 5 presents a comparison of the maximum, minimum, and average objective values for all instances across various customer sizes. As shown in Fig. 5, ALNS consistently outperforms other methods, achieving both lower average and minimum objective values, underscoring its strong optimization capability. Moreover, ALNS demonstrates remarkable robustness, as reflected by the notably narrower gap between its maximum and minimum objective values compared to SA and VNS. The results confirm that ALNS not only excels in solution quality but also ensures reliability across varying instances, affirming its suitability for practical, large-scale optimization tasks. Consequently, ALNS remains the best-performing method.

5.4.3. Convergence curve comparison

Fig. 6 illustrates the convergence trends of ALNS, SA, and VNS on four instances: T20I6, T30I6, T40I6, and T50I6. The convergence curves are generated using the transportation costs at different time points within a calculation phase to clearly illustrate the convergence behavior of the algorithms across various instance scales. FCFS is excluded from this comparison because, as a heuristic method, it lacks a convergence process and is therefore unsuitable for analysis in this context.

The convergence curves highlight the advantages of the ALNS algorithm in terms of both convergence speed and solution quality. At the initial stages, ALNS rapidly reduces the objective value, reflecting high search efficiency. Across all tested instances, ALNS achieves superior solution quality within a short period. In contrast, SA and VNS exhibit slower convergence and produce lower-quality solutions, particularly for larger instances, where ALNS maintains stable performance and superior results. Although SA and VNS may occasionally show comparable early convergence rates to ALNS, their final solutions fall short of reaching the same level, indicating that ALNS exhibits strong adaptability and superior performance across different instance scales.

In summary, ALNS excels in both convergence speed and final solution quality, underscoring its effectiveness and adaptability in solving this problem.

5.5. Sensitivity analysis

This section evaluates the performance of the ALNS algorithm and two benchmark algorithms, SA and VNS, through sensitivity analysis conducted on large-scale instances commonly encountered in practice, namely instances T20I6, T30I6, T40I6, and T50I6. The three algorithms were tested across these four instances to provide managerial insights. Sensitivity analysis was conducted by varying each experimental parameter independently while keeping all other conditions constant, ensuring a rigorous evaluation of their impacts on scheduling performance. Each algorithm was executed ten times for each instance, and the average total cost was calculated for analysis. We analyze the effect of battery thresholds in 5.5.1, computation time in 5.5.2, and present managerial insights in 5.5.3.

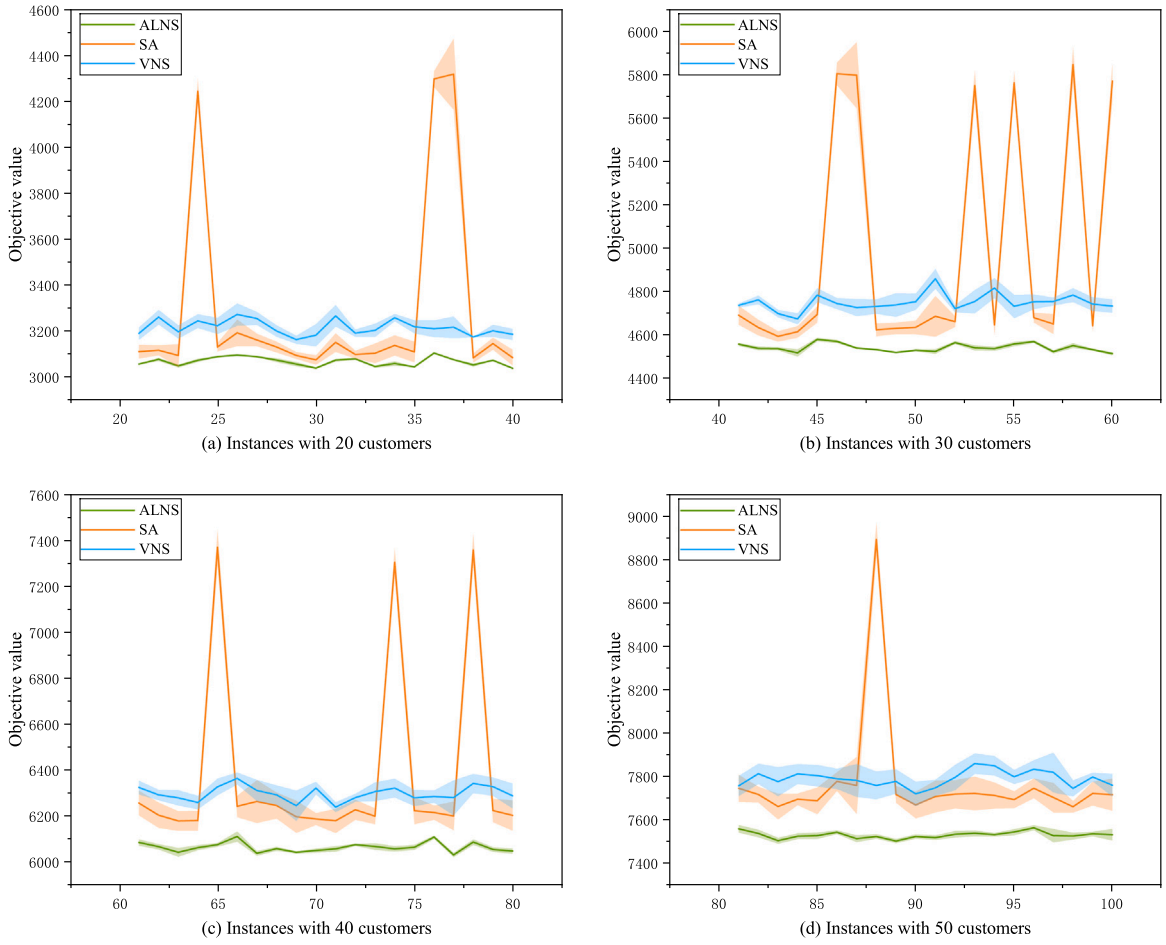

Fig. 5. Comparison of the maximum and minimum values and average results corresponding to the experimental results.

Table 11
Experimental results for the instances with 30 customers.

Instance	FCFS	SA	VNS	ALNS	vs. FCFS	vs. SA	vs. VNS
	Obj	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T30I1	7680.9	4689.7 ± 43.2	4734.8 ± 9.9	**4556.3 ± 6.9**	−40.7%	−2.8%	−3.8%
T30I2	7733.0	4633.3 ± 36.5	4759.9 ± 20.3	**4537.3 ± 9.1**	−41.3%	−2.1%	−4.7%
T30I3	6375.1	4592.1 ± 24.1	4697.5 ± 17.4	**4536.2 ± 7.2**	−28.8%	−1.2%	−3.4%
T30I4	7412.9	4613.8 ± 26.9	4673.6 ± 25.4	**4516.6 ± 16.7**	−39.1%	−2.1%	−3.4%
T30I5	7642.0	4693.8 ± 37.0	4782.4 ± 32.4	**4578.5 ± 8.2**	−40.1%	−2.5%	−4.3%
T30I6	6304.2	5802.4 ± 52.3	4743.3 ± 26.1	**4569.3 ± 7.4**	−27.5%	−21.3%	−3.7%
T30I7	6528.3	5796.4 ± 152.6	4725.3 ± 39.5	**4538.6 ± 4.3**	−30.5%	−21.7%	−4.0%
T30I8	7631.0	4623.3 ± 29.6	4729.8 ± 33.3	**4531.4 ± 3.3**	−40.6%	−2.0%	−4.2%
T30I9	7503.4	4629.9 ± 26.6	4737.0 ± 54.5	**4518.7 ± 3.8**	−39.8%	−2.4%	−4.6%
T30I10	6423.3	4634.2 ± 31.2	4752.0 ± 37.5	**4528.8 ± 4.7**	−29.5%	−2.3%	−4.7%
T30I11	7791.9	4685.5 ± 93.6	4857.7 ± 46.4	**4523.2 ± 11.5**	−42.0%	−3.5%	−6.9%
T30I12	6398.9	4660.6 ± 24.6	4719.4 ± 16.4	**4564.1 ± 6.9**	−28.7%	−2.1%	−3.3%
T30I13	7435.1	5747.7 ± 69.8	4752.5 ± 54.2	**4540.0 ± 12.2**	−38.9%	−21.0%	−4.5%
T30I14	7788.8	4645.0 ± 45.3	4814.9 ± 46.4	**4535.7 ± 9.9**	−41.8%	−2.4%	−5.8%
T30I15	8861.5	5760.4 ± 51.4	4730.1 ± 54.1	**4557.2 ± 10.1**	−48.6%	−20.9%	−3.7%
T30I16	6224.1	4678.9 ± 22.2	4751.9 ± 34.3	**4569.0 ± 6.9**	−26.6%	−2.4%	−3.8%
T30I17	7489.3	4650.1 ± 46.6	4752.8 ± 18.5	**4522.1 ± 7.9**	−39.6%	−2.8%	−4.9%
T30I18	7721.9	5845.6 ± 79.4	4782.6 ± 32.2	**4550.2 ± 11.5**	−41.1%	−22.2%	−4.9%
T30I19	7677.2	4641.8 ± 26.2	4740.9 ± 31.8	**4531.5 ± 4.6**	−41.0%	−2.4%	−4.4%
T30I20	6299.4	5768.7 ± 86.4	4731.8 ± 31.0	**4512.8 ± 7.6**	−28.4%	−21.8%	−4.6%
Average	7246.1	4989.7 ± 527.1	4748.5 ± 51.4	**4540.9 ± 20.6**	−36.7%	−8.1%	−4.4%

Table 12
Experimental results for the instances with 40 customers.

Instance	FCFS	SA	VNS	ALNS	vs. FCFS	vs. SA	vs. VNS
	Obj	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T40I1	11 884.2	6259.9 ± 53.7	6327.5 ± 30.2	**6088.0 ± 13.9**	−48.8%	−2.7%	−3.8%
T40I2	9540.8	6206.7 ± 55.2	6294.9 ± 22.8	**6069.3 ± 11.5**	−36.4%	−2.2%	−3.6%
T40I3	9434.9	6181.2 ± 42.8	6282.5 ± 33.8	**6045.2 ± 19.3**	−35.9%	−2.2%	−3.8%
T40I4	9145.3	6182.9 ± 43.5	6262.2 ± 30.0	**6065.4 ± 11.0**	−33.7%	−1.9%	−3.1%
T40I5	9428.0	7372.7 ± 75.1	6330.4 ± 37.0	**6078.2 ± 8.2**	−35.5%	−17.6%	−4.0%
T40I6	7959.6	6245.0 ± 47.1	6366.9 ± 26.5	**6113.9 ± 22.4**	−23.2%	−2.1%	−4.0%
T40I7	9453.9	6266.5 ± 93.6	6314.7 ± 46.3	**6040.7 ± 10.7**	−36.1%	−3.6%	−4.3%
T40I8	9425.1	6249.1 ± 56.8	6294.9 ± 41.9	**6060.8 ± 8.9**	−35.7%	−3.0%	−3.7%
T40I9	9417.7	6199.9 ± 69.6	6248.1 ± 66.7	**6045.3 ± 5.2**	−35.8%	−2.5%	−3.2%
T40I10	7983.2	6190.6 ± 26.1	6325.0 ± 28.1	**6053.1 ± 9.0**	−24.2%	−2.2%	−4.3%
T40I11	9510.2	6182.0 ± 52.6	6241.8 ± 21.7	**6059.8 ± 12.4**	−36.3%	−2.0%	−2.9%
T40I12	9385.2	6230.6 ± 44.0	6282.8 ± 17.7	**6078.2 ± 4.7**	−35.2%	−2.4%	−3.3%
T40I13	9430.3	6202.4 ± 34.4	6309.4 ± 41.3	**6070.3 ± 14.8**	−35.6%	−2.1%	−3.8%
T40I14	9755.2	7307.5 ± 61.0	6325.4 ± 41.0	**6060.4 ± 12.1**	−37.9%	−17.1%	−4.2%
T40I15	10 539.0	6226.5 ± 59.0	6281.9 ± 35.7	**6067.2 ± 11.4**	−42.4%	−2.6%	−3.4%
T40I16	9336.3	6218.2 ± 31.0	6288.4 ± 27.0	**6111.2 ± 7.7**	−34.5%	−1.7%	−2.8%
T40I17	9321.4	6203.8 ± 62.3	6282.6 ± 76.7	**6034.2 ± 9.5**	−35.3%	−2.7%	−4.0%
T40I18	9274.8	7360.2 ± 67.6	6345.2 ± 41.7	**6089.5 ± 12.3**	−34.3%	−17.3%	−4.0%
T40I19	9290.1	6227.3 ± 51.2	6331.2 ± 40.5	**6057.5 ± 10.5**	−34.8%	−2.7%	−4.3%
T40I20	9249.3	6205.7 ± 65.8	6290.7 ± 54.5	**6050.6 ± 11.2**	−34.6%	−2.5%	−3.8%
Average	9438.2	6385.9 ± 409.1	6301.3 ± 49.9	**6066.9 ± 23.9**	−35.3%	−4.7%	−3.7%

Table 13
Experimental results for the instances with 50 customers.

Instance	FCFS	SA	VNS	ALNS	vs. FCFS	vs. SA	vs. VNS
	Obj	Obj	Obj	Obj	Δ_{Obj}	Δ_{Obj}	Δ_{Obj}
T50I1	13 561.4	7748.8 ± 62.7	7759.7 ± 46.5	**7561.9 ± 17.8**	−44.2%	−2.4%	−2.5%
T50I2	12 223.3	7720.2 ± 36.5	7816.0 ± 47.1	**7540.6 ± 17.7**	−38.3%	−2.3%	−3.5%
T50I3	12 473.4	7664.6 ± 59.3	7779.7 ± 67.3	**7507.3 ± 14.0**	−39.8%	−2.1%	−3.5%
T50I4	10 823.0	7697.9 ± 26.8	7814.6 ± 46.1	**7527.9 ± 13.4**	−30.4%	−2.2%	−3.7%
T50I5	12 400.9	7691.6 ± 61.7	7807.8 ± 50.1	**7530.7 ± 15.0**	−39.3%	−2.1%	−3.5%
T50I6	11 119.8	7780.2 ± 43.1	7792.6 ± 46.8	**7546.6 ± 9.0**	−32.1%	−3.0%	−3.2%
T50I7	13 679.0	7761.6 ± 131.5	7784.9 ± 75.7	**7516.5 ± 15.8**	−45.1%	−3.2%	−3.4%
T50I8	12 366.5	8895.1 ± 75.1	7762.3 ± 64.3	**7527.2 ± 10.8**	−39.1%	−15.4%	−3.0%
T50I9	12 204.1	7720.7 ± 38.6	7780.6 ± 56.3	**7505.8 ± 9.9**	−38.5%	−2.8%	−3.5%
T50I10	10 936.9	7672.2 ± 62.0	7722.9 ± 56.9	**7526.9 ± 10.5**	−31.2%	−1.9%	−2.5%
T50I11	12 398.5	7711.2 ± 73.1	7751.1 ± 37.5	**7521.7 ± 10.4**	−39.3%	−2.5%	−3.0%
T50I12	12 340.3	7721.8 ± 65.7	7798.7 ± 59.1	**7537.2 ± 15.2**	−38.9%	−2.4%	−3.4%
T50I13	11 168.1	7725.1 ± 77.9	7863.1 ± 47.5	**7541.3 ± 13.3**	−32.5%	−2.4%	−4.1%
T50I14	12 364.4	7715.5 ± 60.2	7853.0 ± 45.4	**7535.3 ± 8.4**	−39.1%	−2.3%	−4.0%
T50I15	10 929.7	7696.0 ± 38.8	7801.5 ± 32.6	**7546.9 ± 14.1**	−31.0%	−1.9%	−3.3%
T50I16	11 114.2	7748.8 ± 45.3	7836.6 ± 41.8	**7567.0 ± 11.6**	−31.9%	−2.3%	−3.4%
T50I17	12 218.3	7706.3 ± 68.2	7821.8 ± 92.5	**7530.9 ± 30.7**	−38.4%	−2.3%	−3.7%
T50I18	12 542.4	7663.9 ± 26.5	7748.2 ± 37.5	**7528.4 ± 15.7**	−40.0%	−1.8%	−2.8%
T50I19	12 402.1	7724.6 ± 55.0	7800.6 ± 20.3	**7538.8 ± 10.7**	−39.2%	−2.4%	−3.4%
T50I20	11 027.9	7719.4 ± 74.0	7762.7 ± 53.0	**7534.8 ± 26.9**	−31.7%	−2.4%	−2.9%
Average	12 014.7	7774.3 ± 266.3	7792.9 ± 61.9	**7533.7 ± 21.1**	−37.0%	−3.0%	−3.3%

5.5.1. *Impact of battery thresholds*

The battery threshold is defined as the minimum residual battery level, expressed as a fraction of the total battery capacity, that an AGV must maintain upon arriving at the warehouse, ensuring adequate energy reserves for subsequent operations. To assess the impact of this parameter, the battery threshold was systematically varied from 0.1 to 0.5 in increments of 0.1 while all other parameters remained constant.

Fig. 7 presents several noteworthy trends. A battery threshold of 0.3 consistently provides optimal or near-optimal performance, frequently yielding the lowest average total cost or costs close to the best values across all instances. In contrast, a threshold of 0.1 consistently results in inferior performance, primarily due to insufficient charging, leading to increased penalty costs for early arrivals. Similarly, the highest threshold setting of 0.5 also leads to poorer performance, as it triggers premature or redundant charging, leading to excessive charging events and unnecessary travel. Regarding algorithm performance, ALNS consistently demonstrates superior results with relatively stable behavior. Specifically, for ALNS, higher thresholds (0.3, 0.4, and 0.5) consistently outperform lower thresholds (0.1 and 0.2), with a notable cost reduction typically observed between thresholds of 0.2 and 0.3. For VNS, the most significant reductions in average cost generally occur earlier, between thresholds of 0.1 and 0.2. Meanwhile, SA exhibits a threshold sensitivity similar to that of ALNS in most instances, with cost reductions from thresholds of 0.2 to 0.3, except for instance T30I6, where changes in the battery threshold have minimal impact on cost performance for all algorithms.

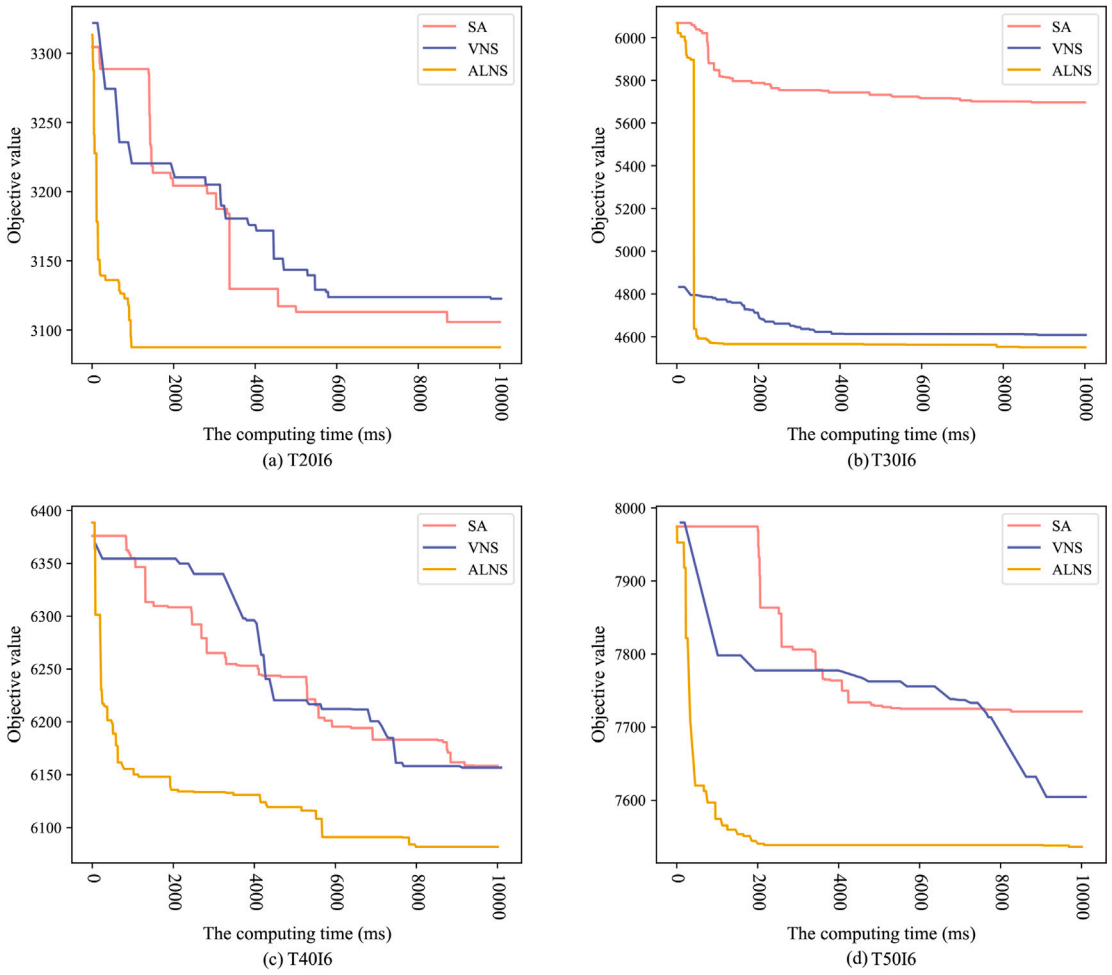

Fig. 6. Convergence curves of algorithms.

5.5.2. Impact of computation time

To investigate the impact of computation time on solution quality, experiments were conducted by varying the maximum allowed computation time from 5 to 25 s in increments of 5 s. The performance of three algorithms, ALNS, SA, and VNS, was quantified by calculating the relative performance gap, which measures the deviation between the solution obtained at a given computation time limit and the best-known solution achieved at the computation time limit of 25 s.

Fig. 8 shows that the relative performance gap decreases with increased computation time limit, confirming that extended computation time limit enables a more thorough exploration of the solution space and improves solution quality. Notably, most improvements occur in the initial time increments, particularly between 5 and 10 s, where a steep reduction in the gap is observed, suggesting that significant enhancements in solution quality can be achieved rapidly. Moreover, ALNS consistently outperforms both SA and VNS by achieving substantially smaller gaps and demonstrating robust stability across all test instances.

5.5.3. Managerial insights for industrial managers

The sensitivity analysis results provide several practical managerial insights:

(1) A moderate battery threshold (approximately 0.3) balances charging frequency and cost-efficiency, avoiding penalties associated with either excessive or insufficient charging.

(2) The ALNS algorithm consistently provides superior solution quality and robustness across various parameter settings, making it particularly suitable for real-world applications where consistent performance is crucial.

(3) Allocating moderate computation resources, specifically between 10 and 15 s, is sufficient to yield high-quality solutions. This insight is valuable for decision-makers seeking an optimal balance between computational efficiency and solution quality in practical applications.

Collectively, these insights highlight the importance of selecting appropriate battery thresholds, employing robust algorithms such as ALNS, and strategically allocating computational resources. Such considerations optimize operational strategies, thereby enhancing the overall efficiency and reliability of AGV scheduling systems in industrial contexts.

6. Conclusions and future work

In conclusion, this study addressed an AGV routing problem that incorporates fast charging to extend the endurance range of AGVs. To tackle this issue, an MILP model was formulated to minimize the total transportation costs. However, due to the complexity of the investigates problem, commercial solvers such as Gurobi handle the MILP model efficiently only on small instances. Therefore, an improved

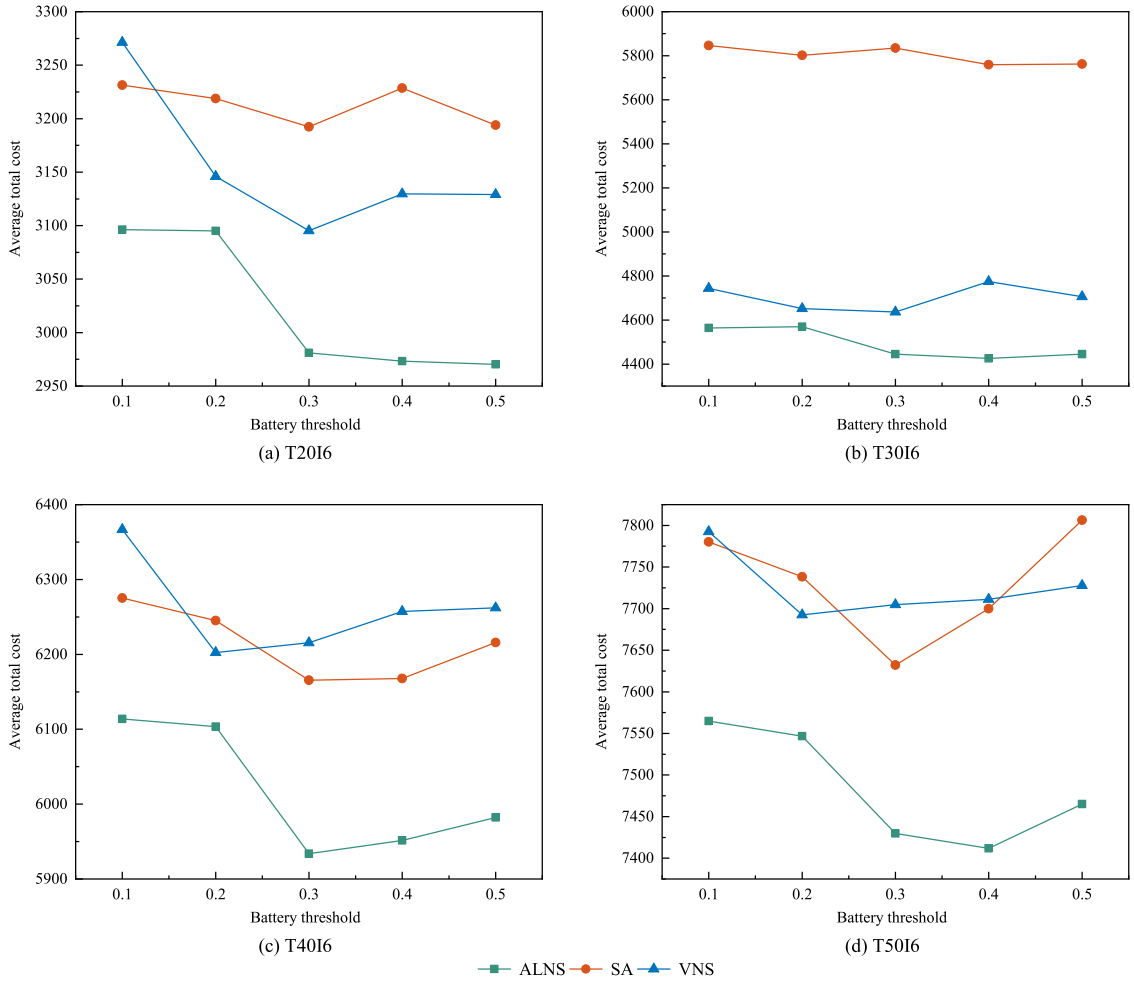

Fig. 7. Sensitivity analysis for battery threshold.

ALNS algorithm, which includes a charging station insertion heuristic, was developed. Furthermore, comprehensive numerical experiments demonstrated the effectiveness of this approach, highlighting its competitiveness against existing state-of-the-art heuristics and metaheuristic algorithms. All algorithms were tested on 110 instances derived from a matrix manufacturing workshop in the real-world electronics equipment manufacturing industry. The results indicate that the solutions generated by the proposed ALNS algorithm closely approximate the exact solutions for small-scale instances and significantly outperform the other three methods for larger instances. These findings emphasize the significance of the proposed approach in enhancing the utilization of AGVs in matrix manufacturing workshops.

Nonetheless, several limitations remain. First, our current model assumes a constant AGV speed and dual-lane pathways, which may not accurately reflect the variability in real-world production environments. Second, the problem setting in this paper omits consideration of unforeseen events and the associated adaptive strategies. To implement the proposed method in a dynamic production format, future research should focus on integrating real-time data and adaptive scheduling mechanisms. For instance, incorporating online learning algorithms or real-time optimization techniques could enable continuous updating of routing decisions based on current production conditions, thereby further enhancing operational responsiveness and efficiency.

CRediT authorship contribution statement

Jianbin Xin: Writing – review & editing, Supervision, Conceptualization, Writing – original draft, Methodology. **Shilong Guo:** Writing – original draft, Resources, Visualization, Methodology. **Yanhong Liu:** Writing – review & editing, Validation, Conceptualization, Supervision. **Yanjie Zhou:** Supervision, Writing – review & editing, Validation, Methodology, Writing – original draft. **Andrea D'Ariano:** Investigation, Supervision, Validation, Writing – review & editing.

Acknowledgment

This research is supported by the National Natural Science Foundation of China under Grant (62173311, 72201252), Zhongyuan Scientific and Technological Innovation Leading Talent Program under Grant (254000510010) and Key Research and Development Program of Henan Province under Grant (241111110500).

Data availability

Data will be made available on request.

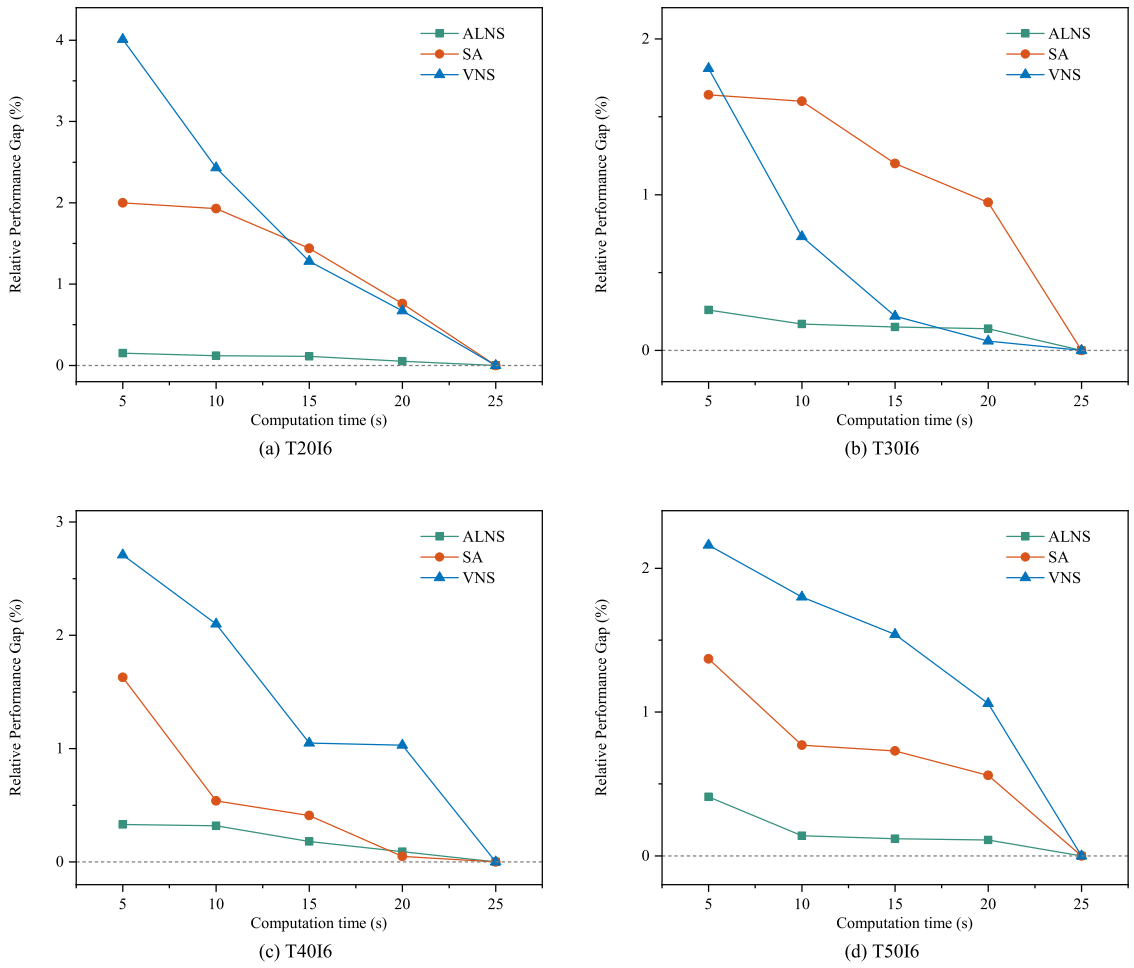

Fig. 8. Sensitivity analysis for computation time.

References

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

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.

Cai, Y., Wu, Y., & Fang, C. (2024). Double-assistant evolutionary multitasking algorithm for enhanced electric vehicle routing with backup batteries and battery swapping stations. *Expert Systems with Applications*, *237*, Article 121600.

Chen, X., He, S., Zhang, Y., Tong, L. C., Shang, P., & Zhou, X. (2020). Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework. *Transportation Research Part C: Emerging Technologies*, *114*, 241–271.

Chen, C., Tiong, L. K., & Chen, I.-M. (2019). Using a genetic algorithm to schedule the space-constrained AGV-based prefabricated bathroom units manufacturing system. *International Journal of Production Research*, *57*(10), 3003–3019.

Chi, H., Sang, H.-Y., Zhang, B., Duan, P., & Zou, W.-Q. (2024). BDE-Jaya: A binary discrete enhanced Jaya algorithm for multiple automated guided vehicle scheduling problem in matrix manufacturing workshop. *Swarm and Evolutionary Computation*, *89*, Article 101651.

Conrad, R. G., & Figliozzi, M. A. (2011). The recharging vehicle routing problem. In *Proceedings of the 2011 industrial engineering research conference, Vol. 8*. IISE Norcross, GA.

Cortés-Murcia, D. L., Prodhon, C., & Murat Afsar, H. (2019). The electric vehicle routing problem with time windows, partial recharges and satellite customers. *Transportation Research Part E: Logistics and Transportation Review*, *130*, 184–206.

Dönmez, S., Çağrı Koç, & Altıparmak, F. (2022). The mixed fleet vehicle routing problem with partial recharging by multiple chargers: Mathematical model and adaptive large neighborhood search. *Transportation Research Part E: Logistics and Transportation Review*, *167*, Article 102917.

Farooq, B., Bao, J., Raza, H., Sun, Y., & Ma, Q. (2021). Flow-shop path planning for multi-automated guided vehicles in intelligent textile spinning cyber-physical production systems dynamic environment. *Journal of Manufacturing Systems*, *59*, 98–116.

Gao, Y., Chang, D., Chen, C.-H., & Sha, M. (2024). A digital twin-based decision support approach for AGV scheduling. *Engineering Applications of Artificial Intelligence*, *130*, Article 107687.

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.

Keskin, M., & Çatay, B. (2018). A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, *100*, 172–188.

Leon, R. V., Shoemaker, A. C., & Kacker, R. N. (1987). Performance measures independent of adjustment: an explanation and extension of Taguchi's signal-to-noise ratios. *Technometrics*, *29*(3), 253–265.

Li, W., Fan, H., Cai, L., Guo, W., Wu, Z., & Yang, P. (2024). Digital twin-driven proactive-reactive scheduling framework for port multi-equipment under a complex uncertain environment. *Simulation Modelling Practice and Theory*, *136*, Article 103011.

Li, G., Li, X., Gao, L., & Zeng, B. (2018). Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *10*, 4533–4546.

Li, K., Liu, T., Kumar, P. R., & Han, X. (2024). A reinforcement learning-based hyper-heuristic for AGV task assignment and route planning in parts-to-picker warehouses. *Transportation Research Part E: Logistics and Transportation Review*, *185*, Article 103518.

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*, 7762–7770.

Li, Z. K., Sang, H. Y., Zhang, X. J., Zou, W. Q., Zhang, B., & Meng, L. L. (2022). An effective discrete invasive weed optimization algorithm for multi-AGVs dispatching problem with specific cases in matrix manufacturing workshop. *Computers & Industrial Engineering*, *174*, Article 108755.

Li, G., Zeng, B., Liao, W., Li, X., & Gao, L. (2018). A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system. *Advances in Mechanical Engineering*, *10*, Article 168781401876556.

Luo, L., Zhao, N., Zhu, Y., & Sun, Y. (2023). A* guiding DQN algorithm for automated guided vehicle pathfinding problem of robotic mobile fulfillment systems. *Computers & Industrial Engineering*, *178*, Article 109112.

Ma, M., Yu, F., Xie, T., & Yang, Y. (2024). A hybrid speed optimization strategy based coordinated scheduling between AGVs and yard cranes in U-shaped container terminal. *Computers & Industrial Engineering*, *198*, Article 110712.

Mehami, J., Nawi, M., & Zhong, R. Y. (2018). Smart automated guided vehicles for manufacturing in the context of Industry 4.0 - ScienceDirect. *Procedia Manufacturing*, *26*, 1077–1086.

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.

Qian, B., Feng, F.-L., Yu, N.-K., Hu, R., & Chen, Y.-W. (2024). An alternating direction multiplier method with variable neighborhood search for electric vehicle routing problem with time windows and battery swapping stations. *Applied Soft Computing*, *166*, Article 112141.

Riazi, S., & Lennartson, B. (2021). Using CP/SMT solvers for scheduling and routing of AGVs. *IEEE Transactions on Automation Science and Engineering*, *18*(1), 218–229.

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.

Schneider, M., Stenger, A., & Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, *48*(4), 500–520.

Wang, Z., Hou, S., & Guo, W. (2024). Inventory management of battery swapping and charging stations considering uncertainty. *International Journal of Electrical Power & Energy Systems*, *155*, Article 109528.

Wang, X., Wu, W., Xing, Z., Chen, X., Zhang, T., & Niu, H. (2022). A neural network based multi-state scheduling algorithm for multi-AGV system in FMS. *Journal of Manufacturing Systems*, *64*, 344–355.

Xie, L., Li, H., & Luttmann, L. (2023). Formulating and solving integrated order batching and routing in multi-depot AGV-assisted mixed-shelves warehouses. *European Journal of Operational Research*, *307*(2), 713–730.

Xin, B., Lu, S., Wang, Q., & Deng, F. (2025). A review of flexible job shop scheduling problems considering transportation vehicles. *Frontiers of Information Technology & Electronic Engineering*, *26*(3), 332–353.

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*, *24*(7), 6943–6954.

Xin, J., Yuan, Q., D'Ariano, A., Guo, G., Liu, Y., & Zhou, Y. (2024). Dynamic unbalanced task allocation of warehouse AGVs using integrated adaptive large neighborhood search and kuhn–munkres algorithm. *Computers & Industrial Engineering*, *195*, Article 110410.

Yang, Y., Zhong, M., Dessouky, Y., & Postolache, O. (2018). An integrated scheduling method for AGV routing in automated container terminals. *Computers & Industrial Engineering*, *126*, 482–493.

Zhang, X., Sun, X., An, Y., Zhang, X., Li, C., Zhang, K., et al. (2022). Design of a fast-charge lithium-ion capacitor pack for automated guided vehicle. *Journal of Energy Storage*, *48*, Article 104045.

Zhou, Y., & Lee, G. M. (2020). A bi-objective medical relief shelter location problem considering coverage ratios. *International Journal of Industrial Engineering*, *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.

Zou, W.-Q., Pan, Q.-K., & Wang, L. (2021). An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery. *Knowledge-Based Systems*, *218*, Article 106881.

Zou, W.-Q., Pan, Q.-K., Wang, L., Miao, Z.-H., & Peng, C. (2022). Efficient multiobjective optimization for an AGV energy-efficient scheduling problem with release time. *Knowledge-Based Systems*, *242*, Article 108334.