Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.068384

ARTICLE

AGV Scheduling and Bidirectional Conflict-Free Routing Problem with Battery Swapping in Automated Container Terminals

He Huang and Jin Zhu*

Institute of Logistics Science and Engineering, Shanghai Maritime University, No. 1550, Haigang Avenue, Pudong New Area, Shanghai, 201306, China

*Corresponding Author: Jin Zhu. Email: jinzhu@shmtu.edu.cn

Received: 27 May 2025; Accepted: 25 July 2025; Published: 31 August 2025

ABSTRACT: Automated guided vehicles (AGVs) are key equipment in automated container terminals (ACTs), and their operational efficiency can be impacted by conflicts and battery swapping. Additionally, AGVs have bidirectional transportation capabilities, allowing them to move in the opposite direction without turning around, which helps reduce transportation time. This paper aims at the problem of AGV scheduling and bidirectional conflict-free routing with battery swapping in automated terminals. A bi-level mixed integer programming (MIP) model is proposed, taking into account task assignment, bidirectional conflict-free routing, and battery swapping. The upper model focuses on container task assignment and AGV battery swapping planning, while the lower model ensures conflict-free movement of AGVs. A double-threshold battery swapping strategy is introduced, allowing AGVs to utilize waiting time for loading for battery swapping. An improved differential evolution variable neighborhood search (IDE-VNS) algorithm is developed to solve the bi-level MIP model, aiming to minimize the completion time of all jobs. Experimental results demonstrate that compared to the differential evolution (DE) algorithm and the genetic algorithm (GA), the IDE-VNS algorithm reduces fitness values by 44.49% and 45.22%, though it does increase computation time by 56.28% and 62.03%, respectively. Bidirectional transportation reduces the fitness value by an average of 10.97% when the container scale is small. As the container scale increases, the fitness value of bidirectional transportation gradually approaches that of unidirectional transportation. The results further show that the double-threshold battery swapping strategy enhances AGV utilization and reduces the fitness value.

KEYWORDS: Automated container terminal (ACT); AGV scheduling; bidirectional conflict-free routing; battery swapping; different evolution algorithm

1 Introduction

With the maturation of automated equipment technology and the update of port comprehensive management systems, traditional container terminal is gradually transformed into automated container terminal. Automated container terminals (ACT) can be divided into three core areas: terminal operation area, Automated guided vehicles (AGV) transportation area and yard operation area. AGV transportation area serves as the connecting link between these areas, and its efficiency has a significant impact on the overall operation of the automated container terminal [1]. A typical layout of an automated container terminal is shown in Fig. 1. In this layout, the quay cranes (QC) handle the loading and unloading of containers on the quay area, while the yard cranes (YC) manage these operations on the yard area. The AGV-supports are equipped on the yard side to reduce equipment waiting time [2].

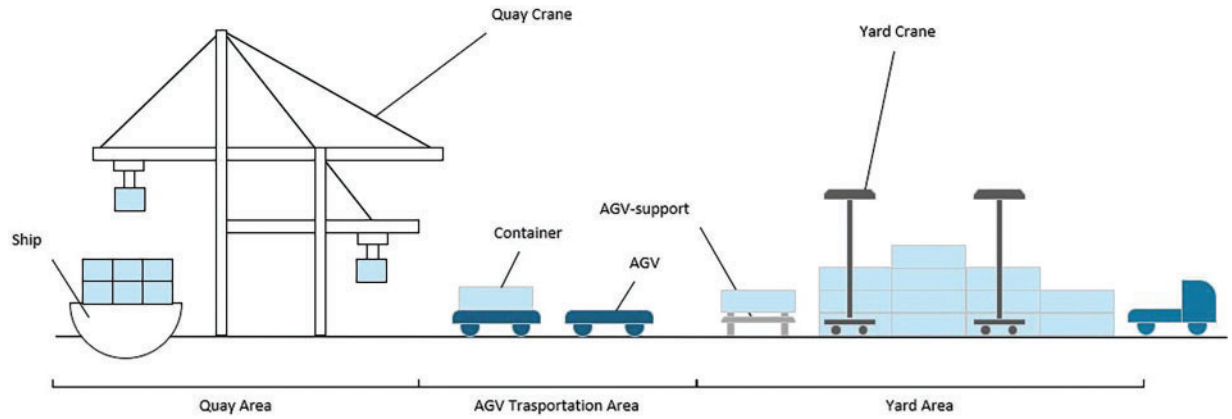

Figure 1: Layout of automated container terminals

In various scenarios, the related challenges of AGV scheduling for automated materials handling systems [3] and flexible manufacturing system [4,5] have been studied. The AGV scheduling problem is complex because it involves not only task assignment but also route planning for AGVs. Multiple AGVs influence one another in the process of transportation, causing conflicts and deadlocks, which affects the scheduling of AGVs. Therefore, some studies jointly consider the scheduling problem and the conflict-free routing problem of AGVs [6–8]. But most of these studies assume that AGVs move in a single fixed direction, and this movement pattern leads to redundant paths for AGVs during transportation operations. In order to solve this problem, bidirectional transportation of AGVs has been considered and applied in AGV path planning for ACTs in a few studies [9,10]. This bidirectional transportation mode enables AGVs to move along the shortest path during transportation and improves the transportation efficiency of AGVs. However, in the recent research on AGV scheduling and conflict-free routing, there are relatively few issues considering the energy consumption and replenishment of AGVs to reduce the complexity of problem-solving. During the actual operation of ACTs, the energy replenishment of AGVs cannot be ignored. Once AGVs fall into a stranded state, it will lead to delays or even paralysis of the entire transportation operation. Therefore, it is worthy of study to take into account the two-way conflict-free transportation operation and energy replenishment of AGVs in the scheduling of AGVs at the same time.

Nowadays, AGVs in automated container terminals are driven mainly by electric power, such as Shanghai Yangshan Port Phase IV in China, the APM Terminals' Maasvlakte II (APMT MVII) terminal in Netherlands and the middle harbor terminal at the Port of Long Beach in America all adopt electric AGVs to complete the transportation of container import and export tasks [11,12]. Therefore, the energy management of AGVs can be regarded as the battery management of AGVs. Electrically driven AGVs are mainly divided into two types: rechargeable and battery-swapping. Battery-swapping AGVs are widely used in many automated terminals due to their higher efficiency [13]. The rechargeable-type operates by setting up some charging stations in the docking area. When the AGV is idle or its battery level falls below a threshold, it heads to the charging station. During charging, the AGV cannot operate, and the charging process takes a longer time. In contrast, the battery-swapping type involves setting up a battery-swapping station in the docking area. The swapping station is responsible for storing batteries, replacing the AGV's battery, and charging the depleted batteries. When the AGV needs to recharge, it goes to the swapping station to replace the battery with a fully charged one. The battery-swapping process allows the AGV to resume its tasks more quickly than the charging method. Thus, some studies focus on how to coordinate AGV scheduling and the battery replenishment of AGVs. For example, Yang et al. [11] study the AGV scheduling problem with limited capacity of battery swap stations with the goal of minimizing the energy consumption of AGVs. Xiao

et al. [14] studied the AGV scheduling problem in the hybrid mode of battery swapping and charging and concluded that the mode can effectively reduce the costs. However, few AGV scheduling studies that consider battery management focus on the conflict-free routing issue, assuming that AGV scheduling is feasible under the condition of having sufficient spaces in the horizontal transport area.

This paper aims to solve the AGV scheduling problem in ACTs by simultaneously considering bidirectional conflict-free routing, battery swapping and the import/export container tasks. A double-threshold battery swapping strategy is proposed and an improved differential evolution variable neighborhood search (IDE-VNS) algorithm is developed to solve this problem. Specifically, the potential contributions of this paper are as follows:

(1) A bi-level mixed integer programming (MIP) model is built to solve the AGV scheduling and conflict-free routing problem with battery swapping in the ACT. This model considers AGV scheduling and bidirectional conflict-free routing, battery swapping process, and import/export container tasks, thereby providing a more comprehensive analysis of the AGV scheduling problem in automated terminals.
(2) A flexible double-threshold battery swapping constraint is adopted. By setting an additional battery swapping threshold, the algorithm provides more flexible options for AGV battery swapping when this threshold is triggered. Specifically, it allows the AGV to use the waiting time for loading to perform the swap, thereby reducing the transportation delay caused by the swapping operation.
(3) The improved differential evolution-variable neighborhood search (IDE-VNS) algorithm is developed. The algorithm performs a more detailed local search within the neighborhood intervals of the solutions during the iterative process of the differential evolution algorithm to prevent the algorithm from getting trapped in local optima.

This paper is organized as follows: Section 2 provides a review of related works; Section 3 defines the problem and formulates a bi-level MIP model; Section 4 discusses the proposed IDE-VNS algorithm for solving the problem; Section 5 presents and discusses the computational results; Section 6 concludes the work and proposes several directions for future research.

2 Related Works

2.1 AGV Scheduling with Conflict-Free Routing in ACT

In an ACT system, AGV scheduling involves assigning starting and ending points to AGVs and arranging for them to complete transportation tasks between these points. Multiple AGVs influence one another in the transportation process. Some scholars endeavor to solving problems such as conflicts [15], deadlocks [16] and congestion [17] to enable AGVs to arrive at the destination nodes as swiftly as possible. Some recent works integrated the above two problems and studied the joint optimization of AGV scheduling and conflict-free path planning. Yang et al. [6] proposed a bi-level model to solve the integrated scheduling problem for handling equipment cooperation and AGV routing. Adamo et al. [7] investigated vehicle paths and speeds on different road sections to avoid conflicts, introducing a path-planning scheme that mitigates route conflicts by adjusting AGV path selection and operational speed. Zhong et al. [8] developed a mixed-integer programming model based on dynamic path planning with the goal of minimizing AGV delay time to solve the multi-AGV scheduling problem with conflict-free paths. Shen et al. [18] proposed a scheduling mode of group operation area and used a hybrid algorithm with fuzzy logic controller to simulate the behavior of AGVs, in order to minimize the maximum completion time of AGVs. Yue and Fan [19] considered the AGV scheduling and conflict-free routing problems under the delayed waiting issue in the uncertain environment during dynamic scheduling optimization, establishing a two-stage model and an

integrated algorithm combining the Dijkstra and Q-learning algorithms to minimize AGV transportation costs. Gao et al. [20] established mathematical models for multi-AGV dispatching and use a contract net protocol to solve the assignment problem, while machine learning is used to the path problem. Lou et al. [21] explored a conflict resolution method based on the Yen algorithm to predict and resolve potential conflicts of AGVs, thereby achieving the scheduling and conflict-free path of AGVs. Zhong et al. [22] studied the joint scheduling problem of AGVs and YCs, considering conflict-free paths for AGVs and the capacity constraints on buffer bracket in blocks, with the goal of minimizing AGV energy consumption. Liu et al. [23] studied conflict-free scheduling of horizontal transportation and loading/unloading equipment in a U-shaped yard layout, creating a bilevel programming model to minimize AGV and truck completion and waiting times. Xu et al. [24] studied the scheduling problem of using Automated Intelligent Vehicles (AIVs) with precise and flexible mobility at automated terminals and proposed an Adaptive Large Neighborhood Search (ALNS) algorithm for real-time scheduling to quickly generate conflict-free scheduling solutions. Li et al. [25] proposed a grid-based path model to solve the conflict problem of AGV dynamic scheduling. Xiong et al. [26] proposed a fast and dynamic rolling scheduling model to efficiently and quickly perform global scheduling and realize the dynamic obstacle avoidance of AGVs. Terfasse et al. [27] suggested an integer linear programming model for collision-free path planning of AGVs in container terminals.

The above studies assume that AGVs perform transportation tasks in a single direction to reduce the occurrence of conflicts. The above articles all assume that AGVs operate in a single direction to reduce conflicts, but this leads to some paths not being the shortest. To enable AGVs to find shorter routes for transportation, some studies have applied the bidirectional characteristics of AGVs in automated container terminals [9,10]. Wang and Zeng [9] established a mixed integer programming model for the problem considering the transportation path of bidirectional multi-parallel lanes to better fit the operation status of the actual automated terminal, and solved the AGV scheduling and conflict-free routing problem on a small scale by using the branch and bound algorithm. Cao et al. [10] proposed a bi-level mixed integer programming model to minimize the completion time of all tasks for the automated terminal operation problem with comprehensive consideration of equipment collaborative, bidirectional conflict-free routing of AGV and import and export containers.

However, these above studies overlook AGV battery management when addressing AGV scheduling and conflict-free path planning. This paper incorporates energy consumption and battery swapping into the scheduling process, aligning it more closely with AGV operations in automated terminals.

2.2 AGV Scheduling with Battery Management in ACT

With the increasing application of electric AGVs in ACTs, some recent studies consider the battery management of AGVs when studying the AGV scheduling problem. Zou et al. [13] explored the two main battery management methods of automated terminals, battery charging and battery swapping, and pointed out that the battery swapping strategy was better than battery charging strategy in terms of operation time, but higher than battery charging strategy in terms of cost. Ma et al. [28] designed the layout of distributed charging stations and the progressive charging mode, and conducted simulation configuration experiments on the charging AGV of the automated wharf to prove the superiority of the proposed layout and mode. Xiang and Liu [29] developed a nested semi-open queuing network model considering the battery management and unbalanced task allocation problems of automated container terminals. Zhao et al. [30] developed two coupled models for the AGV scheduling process and battery swapping process, and solved the problem by decoupling the battery swapping demand and swapping plan as common variables. Yang et al. [31] model the integrated scheduling for battery swapping and variability of AGV speed. Li et al. [32] modeled the AGV scheduling problem with battery swapping and random tasks by establishing a two-stage model and designed

a double-threshold battery swapping strategy in order to improve the utilization rate and overall operational efficiency of AGVs. Yang et al. [11] studied the solution to congestion in the case of limited capacity of AGV battery swap stations through a set partitioning method based on breadth-first and depth-first search. Li et al. [33] focused on optimizing task scheduling and battery swap times for AGVs, considering varying battery degradation states. Zhou et al. [34] took an integrated approach to optimize AGV scheduling and energy system operation and maintenance, developing a multi-objective mathematical model to minimize AGV operation time and reduce energy system maintenance costs.

However, the above studies lack the consideration of the conflict-free path of AGV. In the actual transportation of AGVs, conflicts and congestion of AGVs can have a significant impact on the scheduling process of AGVs. This paper simultaneously considers the battery management of AGVs and the bidirectional conflict-free paths, thereby better arranging the scheduling of AGVs.

2.3 Analisis of Relate Works

Through a review of the above literature, the following insights and research gaps can be summarized:

(1) In the scheduling problem of Automated Guided Vehicles (AGVs), ensuring conflict-free paths is crucial for improving AGV scheduling efficiency. Therefore, relevant research needs to address conflict resolution for AGVs. Additionally, the bidirectional transportation method is an effective way to enhance AGV transportation efficiency. However, there is still lake of studies on AGV scheduling and conflict-free path planning under bidirectional transportation.
(2) In AGV scheduling, battery swapping operations can cause delays in transportation tasks, and AGV breakdowns can lead to significant losses in container handling operations. Therefore, related research should take the AGV battery swapping process into account. However, there is still a lack of studies considering battery swapping in AGV scheduling and conflict-free path planning.

This paper focous on the above two problems and studies AGV scheduling considering bidirectional conflict-free routing, battery swapping and the import/export container tasks. And proposed a battery swapping strategy and IDE-VNS algorithm to solve this problem.

3 Problem Description and Modeling

3.1 Problem Description

In this paper, the AGV scheduling and bidirectional conflict-free routing problem with battery swapping is addressed according the typical terminal layout commonly used in most automated terminals as shown in Fig. 2. This paper considers the operations in the horizontal transportation area of AGVs, including the interactive operations between QC, YC and AGVs, as well as the conflict-free movement and battery swapping operations of AGV equipment. Therefore, the transportation area can be simplified to the network diagram as shown in Fig. 3. In the Fig. 3, node 12 is set as the position of battery swap station, nodes {n2, n4, n6, n8, n10} are set as the position of QCs and nodes {n14, n16, n18, n20, n22} are set as the position of YCs, and the edges {(n1, n23), (n3, n21), (n5, n19), (n7, n17), (n9, n15), (n11, n13)} and their nodes are set as the buffer areas of AGVs, that is, there are a sufficient number of parking areas on these edges. For an import container task, the container is loaded onto a AGV when the AGV reaches QC operation area. Subsequently, the AGV conveys the container to YC operation area along a series of preset paths. Thereafter, the YC unloads the container from the AGV and transfers it to the yard. Export container operations can be regarded as a set of reverse operation processes. It is necessary to arrange for the AGV to go to the battery swap station to swap the battery to ensure that the AGV can maintain a normal working state when the battery capacity of the AGV meets the predetermined condition.

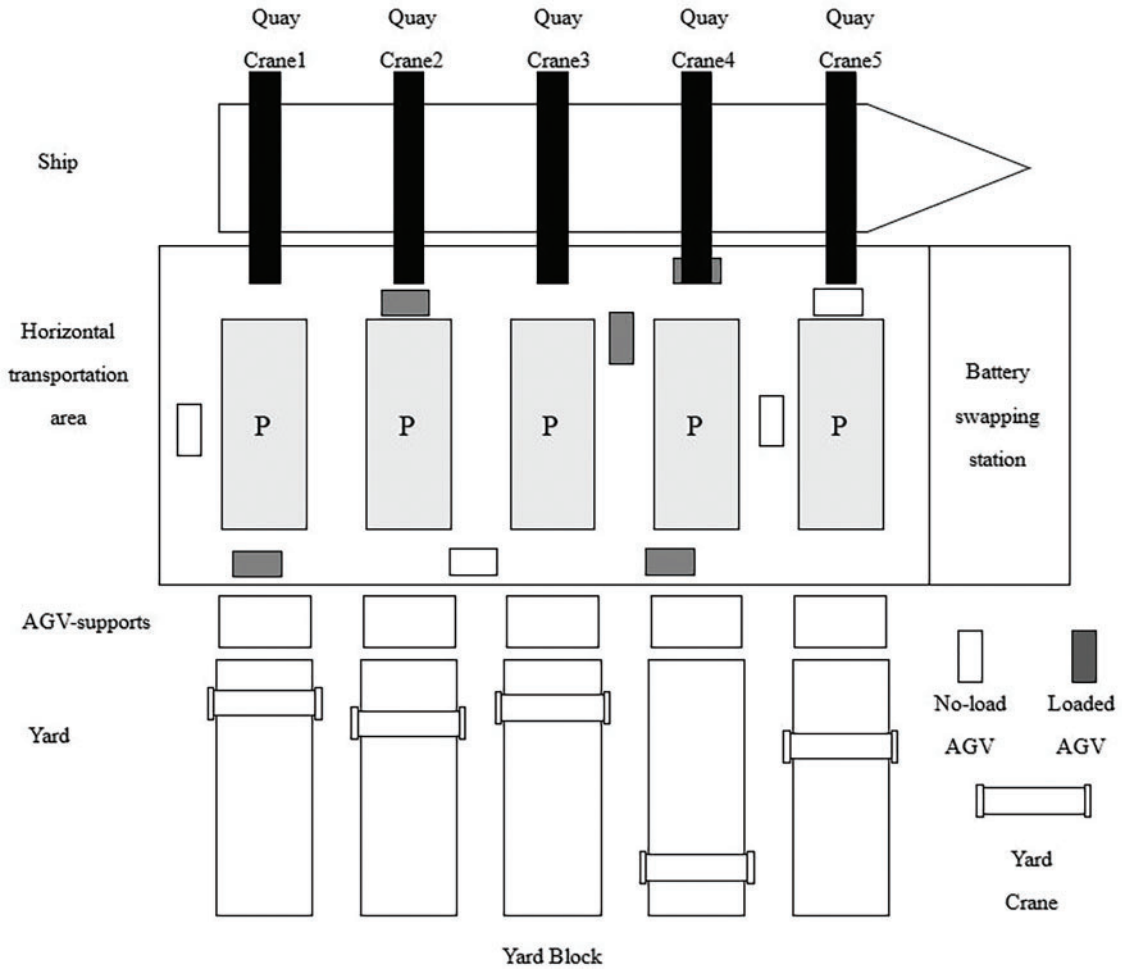

Figure 2: Container handling system in the ACT

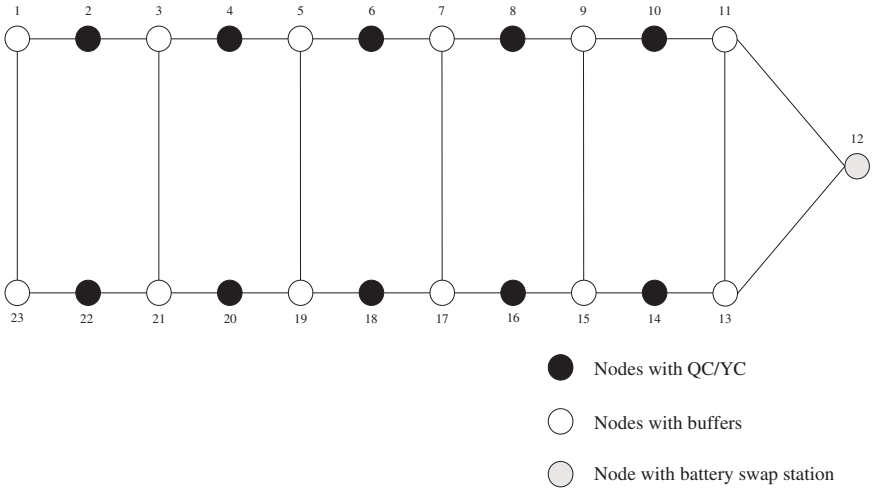

Figure 3: Network of AGV transportation area

This paper considers the bidirectional transportation characteristics of AGVs. By taking advantage of this movement feature of AGVs, it enables AGVs to receive and complete two container tasks within any working cycle. For instance, when an AGV submits a container task on a YC, it can choose any YC to receive the export container task provided by it through bidirectional transportation. However, for the unidirectional transportation mode, it can only accept the task when the AGV can subsequently pass through the YC that provides the export container task. Additionally, the bidirectional transportation of the AGV offers more path options. Compared with the unidirectional transportation mode, bidirectional transportation has a shorter moving path and improve operational efficiency. As shown in Fig. 4a, the AGV follows a unidirectional movement pattern in the horizontal transport area, while Fig. 4b illustrates the bidirectional movement mode. By comparing the transport path of an AGV that picks up and transports an inbound container, we can more clearly explain the advantages of bidirectional transport. For a transport task where the AGV needs to move from point 1 to point 2 to pick up the task and then transport it to point 3, in the unidirectional transport mode, the AGV can only move in a single clockwise direction. Therefore, it needs to pass through four path nodes to reach point 2. In contrast, with bidirectional transport, the AGV can reach the loading point using only two path nodes. During the transport process, the unidirectional mode requires the AGV to travel through an additional path to reach the target node 3, while with the bidirectional mode, the AGV can choose any crossing path between the current and target nodes, achieving a shorter distance. Thus, the bidirectional transport mode offers a clear advantage in terms of distance cost compared to the unidirectional mode.

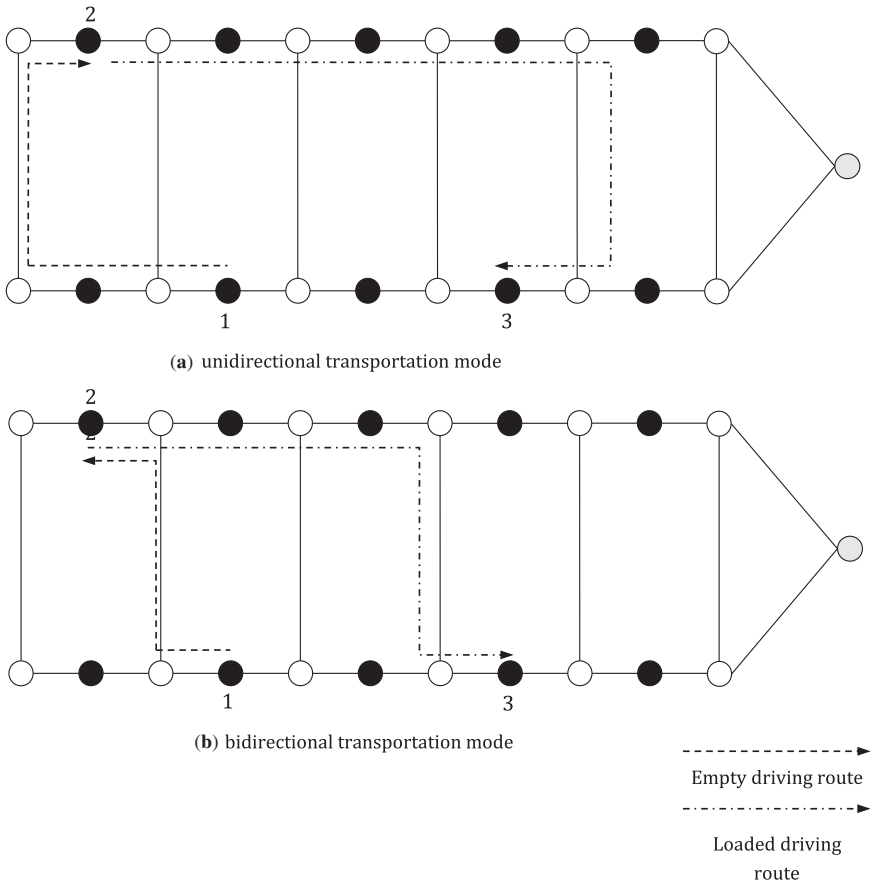

Figure 4: Unidirectional and bidirectional transportation mode of AGV transporting import containers

Furthermore, the conflict problem is also taken into account. As shown in Fig. 5, the common collisions included in this mode are pursuit collisions, the reverse collisions, and the cross collisions. Pursuit collisions refer to the collision that occurs when the rear vehicle of two AGVs moving in the same direction rear-ends the front vehicle in the same path segment. Reverse collisions refer to the collision that occurs when two AGVs moving in opposite directions move in the same path segment. Cross collisions refer to the collision that occurs when two AGVs move along different paths to the same node and compete for that node.

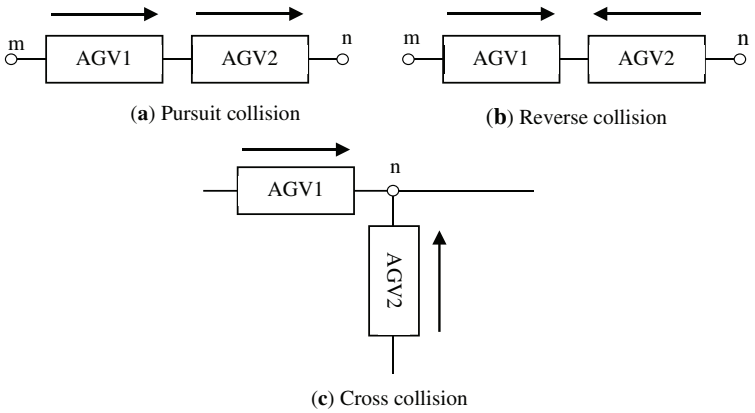

Figure 5: Types of collisions

3.2 Assumption

The assumptions are as follows:

(1) The container task of QC and YC are known, consider the loading/unloading process of containers between QC/YC and AGV.
(2) Each AGV can serve any QC and YC.
(3) QCs, YCs and AGVs can only operate one container task at the same time. The next task cannot be carried out before the last task is completed.
(4) The AGVs maintain a safe distance from one another. The lane change time of the AGV is not considered.
(5) The velocities of the unloaded and loaded AGVs are known and constant.
(6) The initial charge of the AGV is full, and the power consumption per unit time of waiting for loading and unloading, loaded and unloaded transportation is known and constant.
(7) The battery swap station is equipped with sufficient available batteries.
(8) The entire operation involves both import container operations and export container operations. QC should give priority to the import container when carrying out container operation.

3.3 Model Setting

The problem can be characterized as follows: Given a specific quantity and starting positions of AGVs, containers are assigned to each AGV. AGVs traverse along preset paths to complete the transportation of container tasks, with these paths determined by the occupancy status of each path segment. AGVs need to go for battery swapping in time according to the battery swapping strategy during transportation. Therefore, this paper designs a bi-level MIP model, consisting of task layer and routing layer.

The purpose of the task layer model is to assign container tasks to AGVs and adjust the task order to minimize the completion time of all tasks while considering the battery swapping of AGVs. Thus, the

task layer model of AGV can be regarded as the Parallel Machine Scheduling Problem. The model can be described as follows: (1) Denote A as the set of AGVs, each with identical capacity and able to transport at most one container at a time. (2) Let N represent the set of container tasks need be transported by AGVs. (3) The loading and unloading operations of each container task are performed by cranes in the defined pick-up delivery area. (4) Cranes are divided into QC and YC, with QCs located in the quay area and YCs in the yard area. Each AGV is responsible for transporting containers between QC and YC. (5) Container tasks are classified into two categories: import and export containers; (6) The AGV goes to swap battery according to the battery swapping strategy before picking up the container task.

This paper designs a more flexible battery swapping strategy to optimize the battery swapping judgment of AGVs as shown in Fig. 6. The double battery swapping threshold strategy includes a criterion similar to the single fixed threshold: when the AGV's battery level falls below this value, it must proceed immediately for battery swapping. This is defined as the low threshold. Additionally, there is a up threshold used to decide whether the AGV should continue waiting to load containers or go for battery swapping before loading. To maximize battery utilization, the low threshold is set at the minimum level that only guarantees the AGV can reach the swapping station, while the up threshold ensures the AGV can complete its current task before proceeding to swap batteries. This double-threshold battery swapping strategy can be described as follows: This strategy has two battery swapping thresholds, includes the up threshold and the low threshold. When the AGV is assigned a new task, if the battery level falls below the lower threshold, it needs to go directly for battery swapping; if the battery level is above the upper threshold, it will perform the task. When the battery level of AGV between the two thresholds, if loading the container task after the battery swapping will not cause any delay, the AGV will swap its battery first. Otherwise, the task must be performed first.

The routing layer model of AGV can be simplified to an undirected graph $G = (N, A)$, where N represents the node set and A represents the arc set between each node. The network of AGV transportation area as shown in Fig. 3. In this figure, QCs are located at points {n2, n4, n6, n8, n10} and are responsible for completing the handling operation of container tasks between the ship and AGVs. YCs are located at points {n14, n16, n18, n20, n22}, and are responsible for completing the handling operation of container tasks between the yard and AGV. The battery swap station is located at point n12, where the AGV completes the battery swapping. Each AGV is required to convey containers from picking location to the delivery location and load/unload the containers through interaction with the crane. The bidirectional characteristics enables AGVs to travel in either direction along any edge of the undirected graph G, thereby enhancing its mobility and operational flexibility.

The key to ensuring the routing of AGVs without conflicts is to set the time window constraints of AGVs in the path segments, record the occupied time of AGVs entering and leaving each path segment during their movement, and then by comparing the time conflict information of AGVs in the path segments, when there is an overlap in occupied time, arrange another AGV to wait in the buffer to avoid the overlap of time. In this paper, the edges {(n1, n23), (n3, n21), (n5, n19), (n7, n17), (n9, n15), (n11, n13)} and their nodes are set as the buffer areas of AGVs, that is, there are a sufficient number of parking areas on these edges. When conflicts occur, The AGV can wait in these edges until the time conflict is resolved before continuing the current task. The specific conflict resolution method can be described as shown in Fig. 7. AGV1 and AGV2 need to pass through the edge (n2, n3) from two directions successively. AGV1 is on the edge (n2, n3) and enters the edge (n2, n3) through node 2 at 10 s and reaches node 3 at 20 s to leave the edge (n2, n3). AGV2 enters the edge (n2, n3) through node 3 at 15s and reaches node 2 at 25 s on the edge (n2, n3). It can be seen that within the time interval of 15 s–20 s, there is a conflict between AGV1 and AGV2 at the edge (n2, n3), and there is a buffer zone at node n3 where AGV2 is located. Therefore, AGV2 waits in the buffer until the occupation of the edge by AGV1 ends before entering that edge.

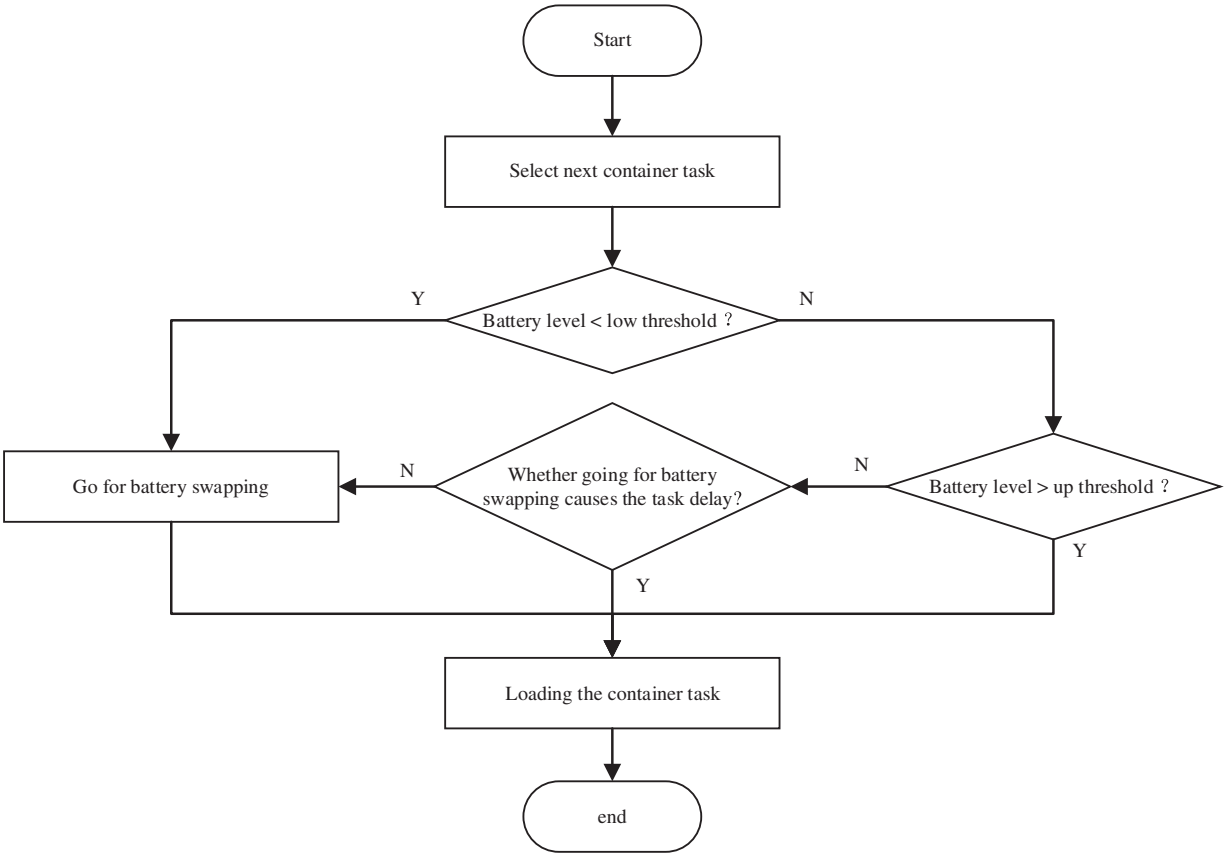

Figure 6: Diagram of battery swapping strategy

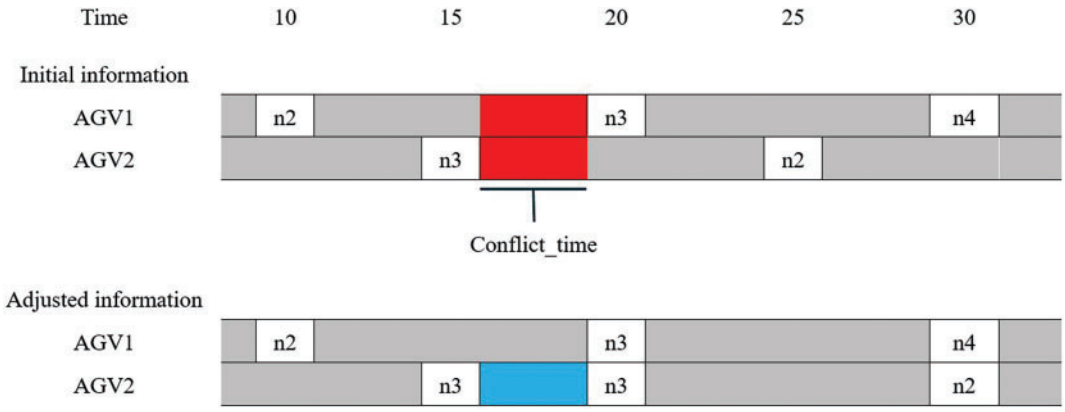

Figure 7: Take up the buffer waiting for solution of conflict

3.3.1 Model Parameters

Sets	
C_i	The set of import containers, $c_i \in C_i$.
C_o	The set of export containers, $c_o \in C_o$.
C	The set of all containers, $C = C_i \bigcup C_o$, $c \in C$.
Q	The set of quay cranes, $q \in Q$.
Y	The set of yard cranes, $y \in Y$.
K	The set of all cranes, $K = Q \bigcup Y$, $k \in K$.
A	The set of AGVs, $a \in A$.
N	The set of all nodes in the network, $n \in N$.
N_1	The set of nodes neighboring node n in the network, $n_1 \in N_1$.
N_2	The set of nodes neighboring node n_1 except the node n in the network, $n_2 \in N_2$.
Paraments	
M	A relatively large positive number.
B_M	Full electric capacity of AGV.
$\overline{B}$	Up threshold of AGV battery swapping.
$\underline{B}$	Low threshold of AGV battery swapping.
T_{ck}	The time consumption of equipment k to operate container task c.
T_r	The battery swap time of AGV.
Variables	
t_{ca}^{init}	The initial time of AGV a transporting task c.
t_{ca}^{finish}	The finish time of AGV a transporting task c.
t_{ck}^{init}	The initial time of crane k operating task c.
t_{ck}^{finish}	The finish time of crane k operating task c.
$t_{acc'}$	The time consumption of AGV a going to pick up container task c' after container task c is completed.
t_{acr}	The time consumption of AGV a going to battery swap station after container task c is completed.
t_{arc}	The time consumption of AGV a going to pick up container task c' after swapping battery.
B_{ca}	The battery level of AGV a completes container task c.
$B_{acc'}$	The power consumption of AGV a going to battery swap station after container task c is completed.
$t_{acnn_1}^{in}$	The time of AGV a performing container task c enters the path nn_1.
$t_{acnn_1}^{out}$	The time of AGV a performing container task c leaves the path nn_1.
H_{cq}	$H_{cq} = 1$, if the container task c is operated by QC q; otherwise $H_{cq} = 0$.
E_{nn_1}	$E_{nn_1} = 1$, if path nn_1 exists; otherwise $E_{nn_1} = 0$.
Y_{ca}	$Y_{ca} = 1$, if the container task c is completed by the AGV a; otherwise $Y_{ca} = 0$.
$X_{acc'}$	$X_{acc'} = 1$, if the AGV a goes to pick up the container task c' after the container task c is completed; otherwise $X_{acc'} = 0$.
Z_{ca}	$Z_{ca} = 1$, if the AGV a going to battery swap station after the container task c is completed; otherwise $Z_{ca} = 0$.
D_{acnn_1}	$D_{acnn_1} = 1$, if the AGV a executing the container task c passes the path nn_1; otherwise $D_{acnn_1} = 0$.

3.3.2 Task Layer Model

The model aims to minimize the container task operation time, defined as the time difference between the completion time of the last container task and the start time of the first container task, as shown in Eq. (1). Specifically, the start time of the first container task is $\min_{c\in C,a_1\in A} t_{ca_1}^{init}$, the finish time of the last one is $\max_{c'\in C,a_2\in A} t_{c'a_2}^{finish}$.

$$F_1 = \max_{c'\in C,a_2\in A} t_{c'a_2}^{finish} - \min_{c\in C,a_1\in A} t_{ca_1}^{init} \tag{1}$$

The constraint conditions are as follows:

$$\sum_{a\in A} Y_{ca} = 1, \forall c \in C \tag{2}$$

$$\sum_{c'\in C} X_{acc'} = 1, \forall a \in A, \forall c \in C \tag{3}$$

$$\sum_{c'\in C} X_{ac'c} = 1, \forall a \in A, \forall c \in C \tag{4}$$

$$X_{acc'} + X_{ac'c} \le 1, \forall a \in A, \forall c, c' \in C \tag{5}$$

$$B_{ca} = B_M, \forall a \in A, \forall c \in R \tag{6}$$

$$t_{ca}^{finish} - t_{ca}^{init} \ge 0, \forall a \in A, \forall c \in C \cup R \tag{7}$$

$$t_{ca}^{finish} \le t_{c'a}^{init} + M\left(1 - X_{acc'}\right) + Z_{ca}\left(t_{acr} + t_{arc'} + T_r\right), \forall a \in A, \forall c, c' \in C, \forall r \in R \tag{8}$$

$$t_{cq}^{finish} \le t_{c'q}^{init} + M(1 - H_{cq}H_{c'q}), \forall q \in Q, \forall c \in C_i, \forall c' \in C_o \tag{9}$$

$$t_{cq}^{finish} \le t_{ca}^{finish} \le t_{cy}^{init}, \forall q \in Q, \forall a \in A, \forall y \in Y, \forall c \in C_i \tag{10}$$

$$t_{cy}^{finish} \le t_{ca}^{finish} \le t_{cq}^{init}, \forall q \in Q, \forall a \in A, \forall y \in Y, \forall c \in C_o \tag{11}$$

$$\max\left(t_{ca}^{finish} + t_{acr} + t_{arc'} + T_r - t_{c'a}^{init}, B_{ca} - \overline{B}\right)(1 - Z_{ca}) \ge 0, \forall a \in A, \quad \forall c, c' \in C, \forall r \in R \tag{12}$$

$$(B_{ca} - \underline{B})(1 - Z_{ca}) \ge 0, \forall a \in A, \forall c \in C \tag{13}$$

Eq. (2) ensures that each container task is assigned to only one AGV. Eq. (3) guarantees that each AGV has a subsequent task while performing the current container task. Eq. (4) ensures that each AGV has a prerequisite task when performing the current container task. Eq. (5) ensures that the preceding container tasks and subsequent container tasks of each container task are different. Eq. (6) indicates that the battery capacity of each AGV is fully after battery swapping at the battery swap station. Eq. (7) ensures that the finish time of each task is not earlier than its start time. Eq. (8) ensures that when the same AGV performs two consecutive container tasks, the start of subsequent container tasks will not be earlier than the completion time of the currently executed container task. Eq. (9) ensures that each QC can carry out the task of exporting containers only after completing all the imported containers. Eq. (10) represents the time relationship in which the imported containers are moved from the QC to the AGV, transported by the AGV to the interaction area of the yard and handed over to the YC for handling. Eq. (11) represents the time relationship of the export container being moved from the YC to the AGV and then transported by the AGV to the berth interaction area and handed over to the QC for handling.

Eqs. (12) and (13) indicate the double-threshold battery swapping. Eq. (12) serves as the upper threshold constraint and Eq. (13) defines the lower threshold constraint. In Eq. (12), if current battery level B_{ca} is greater than the up swappping threshold $\overline{B}$, then $\max\left(t_{ca}^{finish} + t_{acr} + t_{arc'} + T_r - t_{c'a}^{init}, B_{ca} - \overline{B}\right) > 0$ always holds true, so $Z_{ca} = 0$, that is, there is no need to go to the battery swap station. If current battery level B_{ca} is less than

the up swappping threshold $\overline{B}$ and the start time of the next task $t_{c'a}^{init}$ greater than the sum of the end time of the previous task t_{ca}^{finish} and the moving time for going for battery swapping and returning $t_{acr} + t_{arc'} + T_r$, then $\max\left(t_{ca}^{finish} + t_{acr} + t_{arc'} + T_r - t_{c'a}^{init}, B_{ca} - \overline{B}\right) < 0$ is always valid, so the variable $Z_{ca} = 1$, which means it needs to be arranged to go for battery swapping; otherwise, there is no need to go for battery swapping. In Eq. (13), if current battery level B_{ca} is less than the low swappping threshold $\underline{B}$, so the variable $Z_{ca} = 1$, which means it needs to be arranged to go for battery swapping.

3.3.3 Routing Layer Model

The goal of routing layer model is to minimize the transportation time of AGVs, the objective function is shown in Eq. (14):

$$F_2 = \min\left\{\max_{n,n_1 \in N} t_{acnn_1}^{out}\right\} \tag{14}$$

The constraint conditions of the routing layer are as follows:

$$D_{acmn} \le E_{mn}, \forall a \in A, \forall c \in C, \forall n, n_1 \in N \tag{15}$$

$$\sum_{n_1 \in N} D_{acnn_1} = \sum_{n_1 \in N} D_{acn_1 n}, \forall a \in A, \forall c \in C, \forall n \in N \tag{16}$$

$$t_{acnn_1}^{in} + T_{nn_1} \le t_{acnn_1}^{out}, \forall a \in A, \forall c \in C, \forall n, n_1 \in N \tag{17}$$

$$t_{acnn_1}^{out} \le t_{acn_1 n_2}^{in} + M\left(1 - D_{acnn_1} D_{acn_1 n_2}\right), \forall a \in A, \forall c \in C, \forall n \in N, \forall n_1 \in N_1, \forall n_2 \in N_2 \tag{18}$$

$$\left|t_{a_1 c_1 nn_1}^{in} - t_{a_2 c_2 nn_1}^{in}\right| \ge T_d D_{a_1 cnn_1} D_{a_2 cnn_1}, \forall a_1, a_2 \in A, \forall c_1, c_2 \in C, \forall n \in N, \forall n_1 \in N_1 \tag{19}$$

$$\left|t_{a_1 c_1 nn_1}^{in} - t_{a_2 c_2 n_1 n}^{in}\right| \ge T_{nn_1} D_{a_1 cnn_1} D_{a_2 cnn_1}, \forall a_1, a_2 \in A, \forall c_1, c_2 \in C, \forall n \in N, \forall n_1 \in N_1 \tag{20}$$

$$\left|t_{a_1 c_1 n_1 n}^{in} - t_{a_2 c_2 n_2 n}^{in}\right| \ge 0, \forall a_1, a_2 \in A, \forall c_1, c_2 \in C, \forall n \in N, \forall n_1 \in N_1, \forall n_2 \in N_2 \tag{21}$$

Eq. (15) illustrates the correlation between path selection and the connection matrix. Eq. (16) represents the path flow constraint of AGVs. Eq. (17) represents the time relationship between the entry and exit of an AGV transport container along a certain path section. Eq. (18) represents the time relationship between the entry and exit of an AGV transport container along two adjacent path sections. Eq. (19) ensures that two AGVs transporting different tasks maintain a safe distance when they enter the same path section successively. Eq. (20) represents the temporal relationship for two AGVs to avoid reverse collisions, while Eq. (21) represents the temporal relationship for two AGVs to avoid cross collisions.

4 Improve Differential Evolution—Variable Neighborhood Search Algorithm

4.1 Coding and Decoding

According to Assumption (1), all container tasks are operated by the identified QC and YC. Take a set of 10 container tasks as an example: containers 1–5 are import containers, and containers 6–10 are export containers. The starting and ending points of containers are randomly distributed on 5 QC and 5 YC. The information of the container tasks is shown in Fig. 8.

4.1.1 Coding and Decoding of Task Layer

The task layer needs to solve the sequential arrangement of containers and configure appropriate AGVs for the execution of containers. Therefore, this paper designs a 2-N dimensional vector to represent the solution of the task layer model, where N is determined by the quantity of containers, and 2 signifies that the code includes the order of tasks and the assignment of AGVs. The solution to the order of tasks of task layer

can be described as follows. First, generate a set of random numbers with values ranging from (0, 1) based on the number of tasks. For instance, the initial vector could be 0.002-0.674-0.908-0.761-0.934-0.325-0.190-0.136-0.682-0.214. Arrange the initial vector in descending order to obtain the initial sequence of container tasks, which is 10-5-2-3-1-6-8-9-4-7. Adjust the initial order of container tasks according to Eq. (10) so that the operation for the import containers of any QC takes priority over the export containers. The order of container Task 1 and container Task 7 needs to be swapped. Therefore, the sequence code of the swapped tasks is 0.190-0.674-0.908-0.761-0.934-0.325-0.002-0.136-0.682-0.214. And according to the adjusted vector order, it can be known that the operation sequence of the container tasks is 8-5-2-3-1-6-10-9-4-7. The process of adjusting the code and generating the task sequence is shown in Fig. 9.

	Import containers					Export containers				
Containers	1	2	3	4	5	6	7	8	9	10
QC	2	4	5	3	1	3	2	4	5	1
YC	1	4	3	5	2	5	3	2	4	1

Figure 8: The information of container tasks

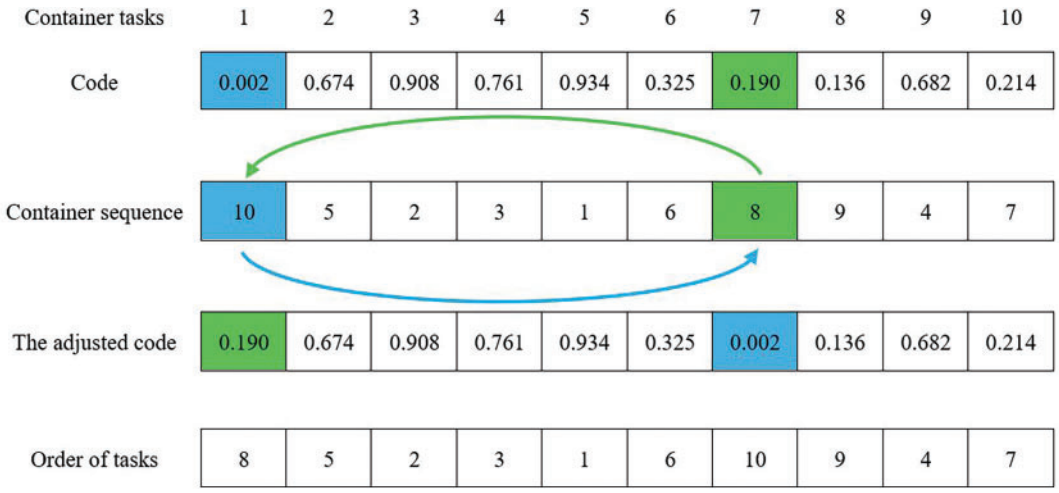

Figure 9: Encoding and decoding of task orders

For the solution of the assignment of AGVs, the value range corresponding to each AGV is partitioned based on the quantity of AGVs, and generate a random sequence of N-dimensional values ranging from (0, 1) based on the number of containers. When the value falls within the value range of the corresponding AGV, it undertakes the transportation task, that is, when $\left\{\frac{i-1}{n} < r < \frac{i}{n}\right\}$, the task is transported by the AGV i, where n represents the quantity of AGVs and r is a random number within the range (0, 1). For example, the generated assignment code is 0.527-0.883-0.879-0.752-0.289-0.518-0.841-0.199-0.329-0.480. Then, the container tasks are assigned to each AGV based on the value interval. The allocation process is shown in Fig. 10. As can be seen from Fig. 9, the container tasks are randomly added to the task list of each AGV, waiting to complete the transportation operation.

Therefore, the encoding of the task layer can be represented by the double-layer real number sequence structure as shown in Fig. 11.

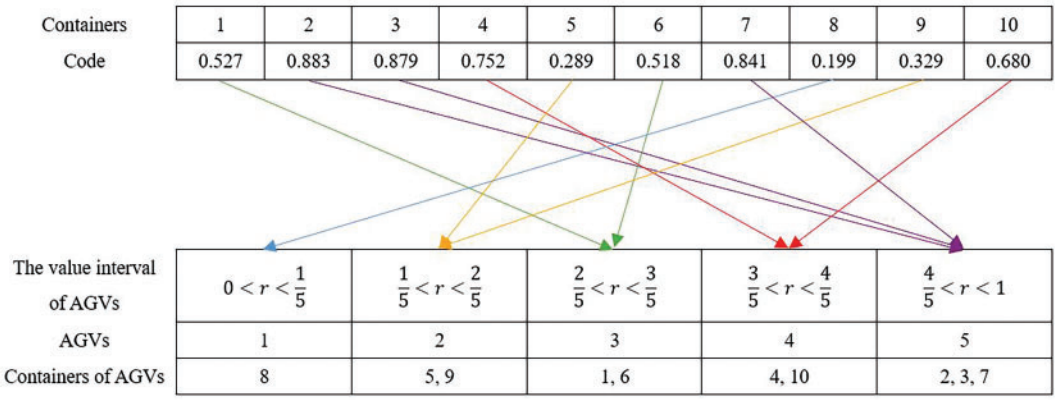

Figure 10: Encoding and decoding of task assignments

Containers	1	2	3	4	5	6	7	8	9	10
Code of task layer										
Order layer	0.190	0.674	0.908	0.761	0.934	0.325	0.002	0.136	0.682	0.214
Assigment layer	0.527	0.883	0.879	0.752	0.289	0.518	0.841	0.199	0.329	0.680

Figure 11: The code of task layer

4.1.2 Coding and Decoding of Routing Layer

In this paper, a 2* N-dimensional real number vector is set to represent the solution of the routing layer, with N being determined by the quantity of containers. The route for an AGV to complete a container task can be determined by the starting point, the destination, and two points on the entry and exit vertical paths. Therefore, for each container task, there should be a pair of values. The first value gives the vertical path that the AGV needs to pass through during transportation, and the second value gives the vertical path that the AGV needs to enter when leaving after completing the current task. If there are no subsequent tasks, it indicates that the AGV will enter the AGV parking area from this interval after completing the task. In order to convert the real number code of AGV into the moving path of AGV, therefore, (0, 1) is divided into six value intervals to represent six selectable crossing paths. When the real number of the path selection code falls within the corresponding value interval, it indicates that the two endpoints of this path are selected as the critical path points for the AGV's movement to form the path of the AGV.

The routing layer needs to plan the movement of each AGV separately. Therefore, the path of each AGV needs to be set according to the container task sequence arranged by the task layer and the allocation of AGVs. According to the task layer, the container tasks of AGV5 are 3-2-7 in sequence. The vectors of the three tasks are obtained as 0.90-0.39-0.51-0.77-0.23-0.63, and then they are transformed into initial vertical paths, which are selected as 6-3-4-5-2-4 to represent the crossing paths passed by the AGV during its movement. When the AGV has two consecutive container tasks of different types, it does not need to go through the vertical path when traveling from the end of the current task to the beginning of the next task. Therefore, the path vector of the AGV needs to be adjusted to meet the actual movement requirements. The adjustment process is shown in Fig. 12. Since the submission node of container Task 2 and the receiving node of Task 7 are both on the yard side, the AGV only needs to move on the yard side to continue its operation tasks at the starting point of Task 7 after completing Task 2. Therefore, the second crossing path corresponding to Task 2 is selected as 0, indicating that it does not need to go through the crossing path. The final obtained adjustment

path is 6-3-4-0-2-4. Therefore, the complete job task node movement of AGV5 can be represented as shown in Fig. 13: (1) after picking up the first import container on node 10 where the 5-th QC is located, the AGV drive cross the 6-th crossing path (n11, n13) and moves to the destination node 18 of the import container to submit it to the 3-rd YC, (2) then drives cross the 3-th vertical path (n5, n19) and travels to the start node 8 to load the second import container task at the 4-th QC, and then drives cross the 4-th vertical path (n7, n17) and reaches to the destination node 16 to submit task to the 4-th YC,(3) AGV drives to node 18 to pick up the export container task by the 3-thYC, and then drives cross the 2-th vertical path (n3, n21) and reaches to the destination node 4 to submit task to the 2-th QC, finally, drives to the buffer zone and waits for the next round of scheduling task.

Container tasks of the AGV	3		2		7	
Code	0.90	0.39	0.51	0.77	0.23	0.63

The vertical path according to the code

The initial path	6	3	4	5	2	4
The types of container tasks	Import container		Import container		Export container	

Adjust vertical path according to the type of container task

Final path	6	3	4	0	2	4

The value interval of vertical path

1	2	3	4	5	6
$0 < r < \frac{1}{6}$	$\frac{1}{6} < r < \frac{2}{6}$	$\frac{2}{6} < r < \frac{3}{6}$	$\frac{3}{6} < r < \frac{4}{6}$	$\frac{4}{6} < r < \frac{5}{6}$	$\frac{5}{6} < r < 1$

Figure 12: Encoding and decoding of the vertical path selection

4.2 Design of Algorithm

Variable neighborhood search (VNS) algorithm can effectively prevent the algorithm from being trapped in local optima and enhance the algorithm's local search capability. In this paper, an improved differential evolution variable neighborhood search (IDE-VNS) algorithm is proposed to solve the problem as shown in Fig. 14.

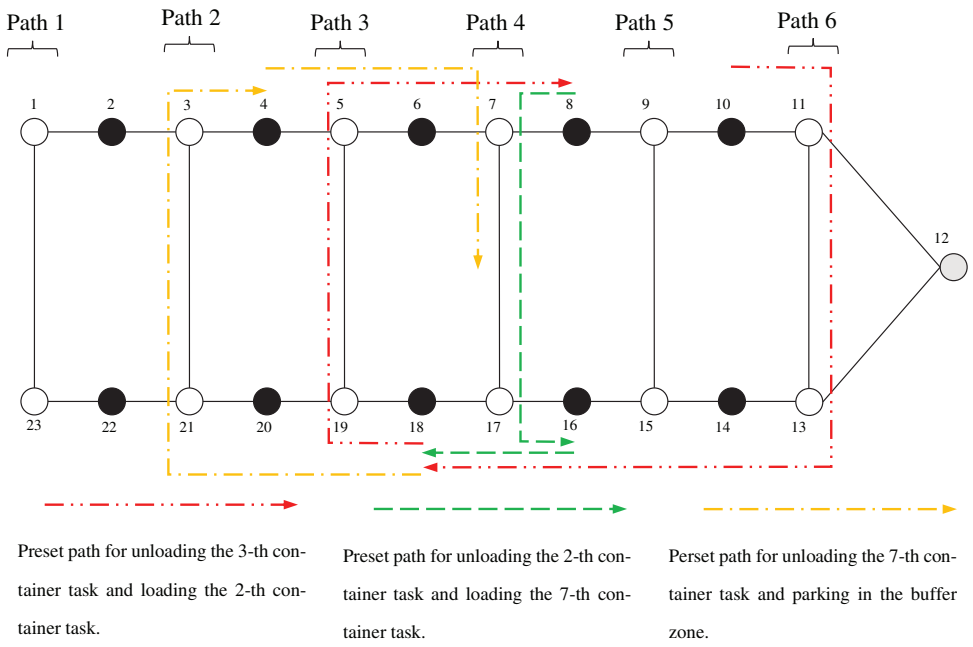

Figure 13: Preset path of AGV obtained from the decoding

4.2.1 Global Search

The differential evolution algorithm is used for global search. Firstly, a mutation operation is performed, three different vectors $X_{r_1}^G$, $X_{r_2}^G$, $X_{r_3}^G$ are randomly selected from the target population to generate trial vector according to Eq. (22).

$$V_i^G = X_{r_1}^G + F \cdot \left(X_{r_2}^G - X_{r_3}^G\right) \tag{22}$$

where X_r^G represents the r-th vector of in the G generation, r_1, r_2, r_3 are three different integers in the range [1, P], P is the population size, F is the crossover operator, and the value range is between [0, 2].

All dimensions of each vector should be between (0, 1). If a solution outside the feasible domain emerges during the mutation process, that is, $v_{ij}^G < 0$ or $v_{ij}^G > 1$, boundary processing is required as shown in Eq. (23).

$$v_{ij}^G = rand \tag{23}$$

where v_{ij}^G is the j-th dimension of the i-th trail vector of the G generation, and $rand$ is a random number between [0, 1].

Secondly, the trial vector V_i^G and the vector X_i^G are selected in each dimension to obtain the offspring vector U_i^G according to Eq. (24).

$$u_{ij}^G = \begin{cases} v_{ij}^G, & if\ rand < CR\ or\ randi(1, D) = J \\ x_{ij}^G, & otherwise \end{cases} \tag{24}$$

where u_{ij}^G is the j-th dimension of the i-th offspring vector of the G generation, x_{ij}^G is the j-th dimension of the i-th vector of the G generation, CR is a mutation operator, and its value is between [0, 1], $rand$ is a random number of [0, 1], $randi(1, D)$ is a random integer within [1, D], D is the dimension of the problem, and J represents a random dimension in the vector.

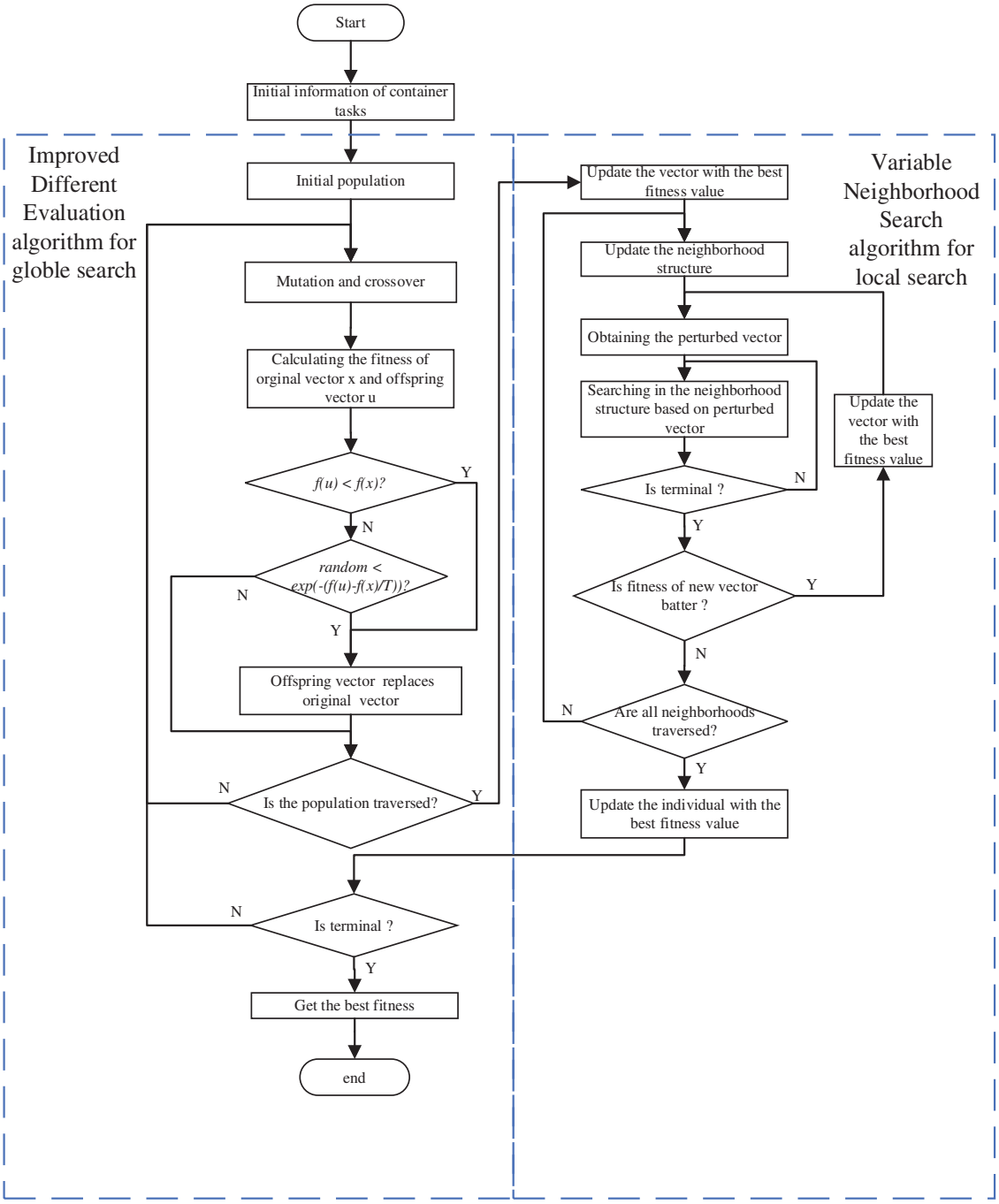

Figure 14: The IDE-VNS flowchart

Finally, the selection operation is performed. The selection operation of the traditional DE algorithm usually adopts the one-to-one greedy idea, that is, only the fitness values of each target vector and its corresponding experimental vector are compared, and the vector with better fitness values is selected to enter the next generation. This selection method may eliminate the poor vectors that are very close to the optimal solution, resulting in a decrease in the convergence speed and search quality of the algorithm. Therefore, the solution acceptance criterion from the simulated annealing algorithm is introduced, and the vectors with poorer performance are accepted according to the probability of being affected by temperature changes.

To enhance search efficiency of the algorithm, the adaptive variable temperature method is adopted. The variation of the annealing temperature T is shown in Eq. (25) as follows:

$$T = T \times \left(1 - \frac{G}{G_{max} + 1}\right) \tag{25}$$

For the minimization problem, vectors with smaller fitness values are selected for the next generation; otherwise, poorer vectors are accepted according to the probability $\exp\left(-\frac{f(U_i^G) - f(X_i^G)}{T}\right)$. The selection operation is shown in Eq. (26):

$$X_i^{G+1} = \begin{cases} U_i^G, & f\left(U_i^G\right) < f\left(X_i^G\right) \\ U_i^G, & f\left(U_i^G\right) < f\left(X_i^G\right) \text{ and } rand < \exp\left(-\dfrac{f\left(U_i^G\right) - f\left(X_i^G\right)}{T}\right) \\ X_i^G, & otherwise \end{cases} \tag{26}$$

where $f()$ is the fitness of the vector.

4.2.2 Local Search

The variable neighborhood search algorithm expands its search range by changing the neighborhood structure of the solution, enabling the algorithm to more easily escape the local optimum. In this paper, variable neighborhood search is performed on the optimal vectors in the current population to improve the local search ability of the differential evolution algorithm.

This paper constructs the following three neighborhood structures to generate new local solutions:

(1) Neighborhood structure N1: In the first-layer vector of the task layer, two dimensions are randomly selected. The corresponding values of the selected dimensions represent the sequence of the two container tasks, and the values corresponding to these two dimensions are swapped. Then, readjust the vector according to the adjustment method in Section 4.1 to ensure that the imported containers can be given priority for processing.
(2) Neighborhood structure N2: In the second-layer vector of the task layer, a dimension is randomly selected. The corresponding value of the selected dimension represents the AGV transporting the container task, and another AGV is then selected for the container task to complete it.
(3) Neighborhood structure N3: In the routing layer vector, a dimension is randomly selected. The value of the selected dimension represents the crossing path of the AGV when it is moving, and the crossing path is reselected to replace the current path.

Based on the above three neighborhood structures, the operation steps are as follows:

Step 1: Take the optimal solution in the contemporary population as the initial vector X, initialize $q = 1$, $q_{max} = 3$, $c_{max} = 40$.

Step 2: If the loop termination condition $q > q_{max}$ is met, take the vector X as the optimal vector X_{best} found in the current search and update it to the optimal solution in the original population. Otherwise, proceed to Step 3.

Step 3: Random disturbance. Obtain the perturbed vector X' by searching in the neighborhood structure Nq through the initial vector X.

Step 4: Local search.

(1) initialize c = 1, Take the perturbated vector X' as the initial solution for the local search.
(2) Output the local optimal vector X' and proceed to Step 5 if $c > c_{max}$; otherwise, proceed to (3).

(3) Obtain a new vector X'' by searching in the neighborhood structure Nq through X', if $f(X'') < f(X')$, replace X' with X''.

Set $c = c + 1$ and return to step (2) to continue the current loop.

Step 5: If $f(X') < f(X)$, let $X = X'$ and proceed to step 4; otherwise, set $q = q + 1$ and proceed to step 2.

5 Experiment and Analysis

In this section, a set of experiments were conducted carried out to evaluate the effectiveness of the algorithm, bidirectional transportation and battery swapping strategy. The experiments were conducted using Python 3.11.5 on a computer with Intel(R) Core (TM) i5-12600KF 3.70 GHz, 16.0 GB of RAM and Windows 11 operation system.

5.1 Parameters Setting of the Model

Experiments are conducted and parameters are set based on the automated terminal layout in Fig. 2. This layout consists of a 23-node network diagram with 5 quay cranes, 5 yard cranes and 1 battery swap station. The distance between each node is set to 64 m, the velocity of AGV is 4 m/s, the unit power consumption of the AGV when waiting for loaded and unloaded is 0.012 s^{-1}, the unit power consumption when moving without load is 0.015 s^{-1}, the unit power consumption when moving with load is 0.016 s^{-1}, the battery swap time is 80 s, and the time for QC/YC to load/unload containers is 120 s.

5.2 Parameters Setting of IDE-VNS Algorithm

The parameters setting of the IDE-VNS algorithm will affect the accuracy and convergence speed of the solution. Therefore, in order to better utilize this algorithm to solve the problem, a parameter configuration experiment was carried out. The crossover operator F and the mutation operator CR are the key parameters that affect the global search performance. Therefore, set F ∈ {0.4, 0.8, 1.2, 1.6, 2.0} and CR ∈ {0.2, 0.4, 0.6, 0.8, 1.0} to solve the problems under four different configurations as shown in Table 1. Best Fitness Value (BFV) is the best fitness value of the result and GAP is the difference of BFV between the BFV under the parameter Settings and the minimum BFV. Avg_gap is the average value of the difference for experiments of different sizes under the parameter Settings.

Table 1: Results of the parameter setting experiment

No.	Parameter		20 Containers–5 AGVs		20 Containers–10 AGVs		50 Containers–5 AGVs		50 Containers–10 AGVs		Avg_Gap
	F	CR	BFV	GAP (%)	BFV	GAP (%)	BFV	GAP (%)	BFV	GAP (%)	
1	0.4	0.2	1666	1.77%	1053	10.26%	4291	1.59%	2689	3.82%	4.36%
2	0.4	0.4	1704	4.09%	1021	6.91%	4485	6.18%	2610	0.77%	4.49%
3	0.4	0.6	1733	5.86%	994	4.08%	4530	7.24%	2654	2.47%	4.91%
4	0.4	0.8	1690	3.24%	1032	8.06%	4401	4.19%	2774	7.10%	5.65%
5	0.4	1	1693	3.42%	1077	12.77%	4405	4.29%	2590	0.00%	5.12%
6	0.8	0.2	1720	5.07%	1008	5.55%	4465	5.71%	2786	7.57%	5.98%
7	0.8	0.4	1688	3.12%	1000	4.71%	4426	4.78%	2696	4.09%	4.18%
8	0.8	0.6	1648	0.67%	981	2.72%	4344	2.84%	2624	1.31%	1.89%
9	0.8	0.8	1792	9.47%	1019	6.70%	4229	0.12%	2706	4.48%	5.19%
10	0.8	1	1674	2.26%	963	0.84%	4282	1.37%	2752	6.25%	2.68%
11	1.2	0.2	1709	4.40%	955	0.00%	4490	6.30%	2744	5.95%	4.16%
12	1.2	0.4	1669	1.95%	1002	4.92%	4326	2.41%	2669	3.05%	3.08%
13	1.2	0.6	1707	4.28%	1018	6.60%	4496	6.44%	2683	3.59%	5.23%

(Continued)

Table 1 (continued)

No.	Parameter		20 Containers–5 AGVs		20 Containers–10 AGVs		50 Containers–5 AGVs		50 Containers–10 AGVs		Avg_Gap
	F	CR	BFV	GAP (%)	BFV	GAP (%)	BFV	GAP (%)	BFV	GAP (%)	
14	1.2	0.8	1675	2.32%	1034	8.27%	4408	4.36%	2728	5.33%	5.07%
15	1.2	1	1637	0.00%	1056	10.58%	4462	5.63%	2742	5.87%	5.52%
16	1.6	0.2	1674	2.26%	997	4.40%	4354	3.08%	2746	6.02%	3.94%
17	1.6	0.4	1730	5.68%	1024	7.23%	4365	3.34%	2702	4.32%	5.14%
18	1.6	0.6	1690	3.24%	1024	7.23%	4418	4.59%	2706	4.48%	4.89%
19	1.6	0.8	1752	7.03%	1078	12.88%	4310	2.04%	2734	5.56%	6.88%
20	1.6	1	1688	3.12%	1032	8.06%	4398	4.12%	2643	2.05%	4.34%
21	2	0.2	1709	4.40%	1006	5.34%	4347	2.91%	2691	3.90%	4.14%
22	2	0.4	1672	2.14%	1042	9.11%	4255	0.73%	2701	4.29%	4.07%
23	2	0.6	1720	5.07%	1032	8.06%	4453	5.42%	2686	3.71%	5.57%
24	2	0.8	1746	6.66%	1010	5.76%	4242	0.43%	2709	4.59%	4.36%
25	2	1	1774	8.37%	1021	6.91%	4224	0.00%	2709	4.59%	4.97%

The experimental results are shown in Table 1, where BFV is the optimal fitness value obtained, and GAP represents the relative error between the problem scale and the minimum fitness value obtained. As can be seen from Fig. 15, the parameter group has the smallest average error value when (F, CR) = (0.8, 0.6). Therefore, the parament group (F, CR) = (0.8, 0.6) used in the following experiments.

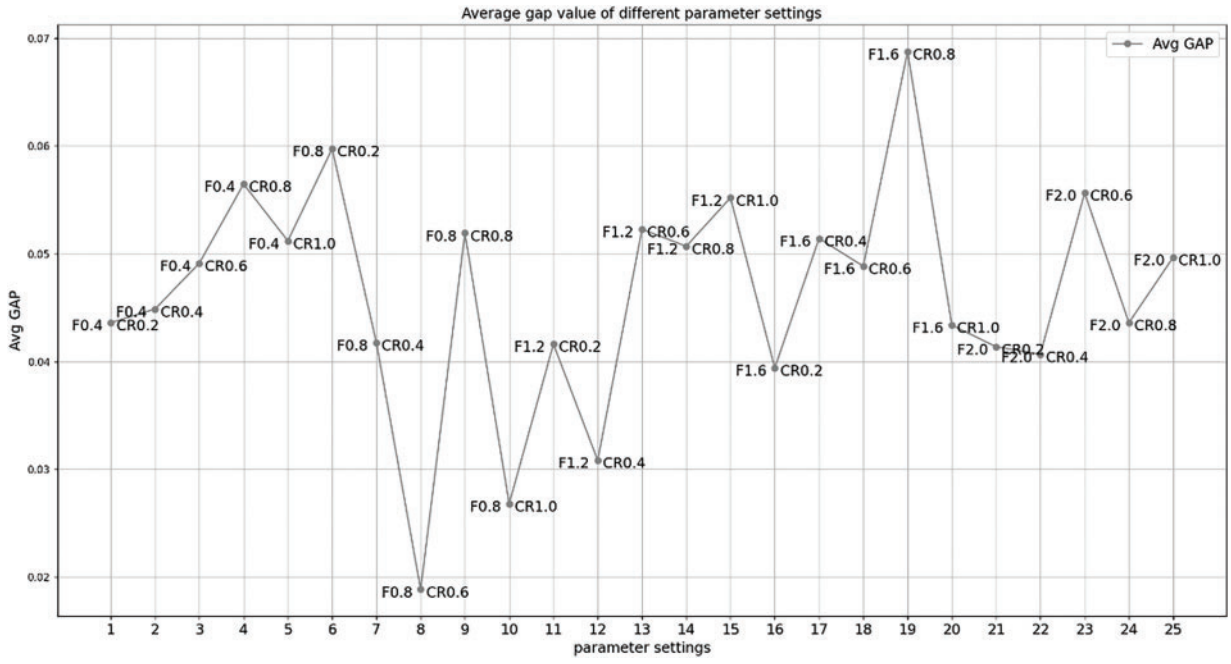

Figure 15: Average gap value of different parameter settings

In addition, to verify the stability of the solution of the algorithm, 20 experiments were conducted on the scale of 50 containers–10 AGVs, and the results are presented in Table 2. BFV is the best fitness values of the result and GAP is the difference of BFV between the BFV under the parameter Settings and the minimum BFV.

Table 2: Results of the stability analysis experiment

No.	BFV	GAP	No.	BFV	GAP
1	2424	0.21%	11	2611	7.94%
2	2441	0.91%	12	2605	7.69%
3	2517	4.05%	13	2579	6.61%
4	2634	8.89%	14	2591	7.11%
5	2419	0.00%	15	2622	8.39%
6	2541	5.04%	16	2596	7.11%
7	2618	8.23%	17	2432	0.54%
8	2579	6.61%	18	2621	8.35%
9	2645	9.34%	19	2504	3.51%
10	2516	4.01%	20	2457	1.57%

It can be known from the 20 groups of experiments in Table 2 that the GAP between the worst solution and the best solution is 9.34%, and the average gap remains at about 5%. Therefore, the error range of the improved algorithm in solving the problem can be accepted, which shows the stability of the algorithm.

5.3 Comparative Experiments and Analysis of IDE-VNS

This section compares the performance between IDE-VNS, differential evolution (DE) algorithm and genetic algorithm (GA) in terms of solution accuracy, convergence speed and solution time. The above three methods adopt the same encoding and decoding for solution. The GA parameters encompass a crossover rate pc = 0.8, and a mutation rate pm = 0.01. For DE, the paraments comprise a crossover operator F = 0.7 and a mutation operator CR = 0.5. The results are presented in Tables 3 and 4. The best fitness value (BFV) and the CPU time obtained by the three algorithms are compared at a small scale and a larger scale. GAP_V is the difference of BFV between algorithms, while GAP_C is the difference of CPU time between algorithms.

Table 3: The results of the algorithm comparison experiment in small-scale problems

No.	Container	AGV	IDE-VNS		DE		GA		GAP_V		GAP_C	
			BFV	CPU Time	BFV	CPU Time	BFV	CPU Time	GAP_V1	GAP_V2	GAP_C1	GAP_C2
1	20	5	1616	122.769	1984	57.266	2106	52.696	23.77%	31.38%	−53.35%	−57.08%
2	20	10	995	132.391	1181	66.654	1295	52.537	18.69%	30.15%	−49.65%	−60.32%
3	25	5	2088	148.399	2613	73.97	2605	67.258	25.14%	24.76%	−50.15%	−54.68%
4	25	10	1483	169.761	1642	85.037	1848	67.118	10.72%	24.61%	−49.91%	−60.46%
5	30	5	2430	156.626	3142	80.787	3236	69.584	29.30%	33.17%	−48.42%	−55.57%
6	30	10	1468	173.807	2016	86.502	2073	73.156	37.33%	41.21%	−50.23%	−57.91%
7	35	5	2963	236.359	3820	119.489	4017	107.619	28.92%	35.57%	−49.45%	−54.47%
8	35	10	1844	256.043	2528	128.979	2582	111.429	37.09%	40.02%	−49.63%	−56.48%
9	40	5	3502	282.461	4622	153.543	4718	132.415	31.98%	34.72%	−45.64%	−53.12%
10	40	10	2051	305.358	2957	155.376	2911	134.163	44.17%	41.93%	−49.12%	−56.06%
11	45	5	3916	347.231	5146	185.669	5294	161.995	31.41%	35.19%	−46.53%	−53.35%
12	45	10	2378	393.238	3334	188.264	3410	160.171	40.20%	43.40%	−44.64%	−52.79%
13	50	5	4184	406.362	5725	224.975	6025	191.824	36.83%	44.00%	−44.64%	−52.79%
14	50	10	2749	448.139	3689	216.808	3842	183.657	34.19%	39.76%	−51.62%	−59.02%
AVG_GAP									30.70%	35.71%	−49.32%	−56.47%

Notes: BFV is best fitness value, CPU time is solution time. GAP_V1 = (DE_BFV − IDE-VNS_BFV)/IDE-VNS_BFV, GAP_V2 = (GA_BFV − IDE-VNS_BFV)/IDE-VNS_BFV, GAP_C1 = (DE_CPU − IDE-VNS_CPU)/IDE-VNS_CPU, GAP_C2 = (GA_CPU − IDE-VNS_CPU)/IDE-VNS_CPU.

Table 4: The results of the algorithm comparison experiment in larger-scale problems

No.	Container	AGV	IDE-VNS		DE		GA		GAP_V		GAP_C	
			BFV	CPU Time	BFV	CPU Time	BFV	CPU Time	GAP_V1	GAP_V2	GAP_C1	GAP_C2
15	100	20	3637	1474.85	5641	685.045	5612	584.499	55.10%	54.30%	−53.55%	−60.37%
16	100	30	3339	1626.084	4682	713.75	4755	605.612	40.22%	42.41%	−56.11%	−62.76%
17	150	20	6523	3289.185	9157	1456.871	8905	1230.34	40.38%	36.52%	−55.71%	−62.59%
18	150	30	5291	2695.461	7547	1189.808	7295	1014.022	42.64%	37.88%	−55.86%	−62.38%
19	200	20	8690	4866.406	12,720	2197.473	13,087	1860.706	46.38%	50.60%	−54.84%	−61.76%
20	200	30	7304	4609.112	10,448	2062.403	10,953	1758.642	43.04%	49.96%	−55.25%	−61.84%
21	250	20	11,461	8095.697	16,312	3427.914	16,247	2960.497	42.33%	41.76%	−57.66%	−63.43%
22	250	30	9273	7632.106	13,423	3290.959	13,592	3186.239	44.75%	46.58%	−56.88%	−58.25%
23	300	20	13,691	12191.821	19,861	4807.727	20,131	4106.616	45.07%	47.04%	−60.57%	−66.32%
24	300	30	11,396	11633.423	16,526	5075.002	16,545	4580.617	45.02%	45.18%	−56.38%	−60.63%
AVG_GAP									44.49%	45.22%	−56.28%	−62.03%

Table 3 presents the results for small-scale problems, including the optimal fitness values and computation times for the three algorithms. It can be observed that the average gap (GAP-V1) between IDE-VNS and the DE algorithm is 30.70%, and the average gap (GAP-V2) between IDE-VNS and the GA algorithm is 35.71%. The IDE-VNS algorithm outperforms the DE and GA algorithms in terms of solution quality at this scale. As the number of container tasks increases, the solution quality improves. However, in terms of computation time, the IDE-VNS algorithm is slower than the other two algorithms, with an average gap (GAP_C1) of −49.32% compared to the DE algorithm, and an average gap (GAP_C2) of −56.47% compared to the GA algorithm. For the same scale of container tasks, increasing the number of AGVs can improve operational efficiency to some extent, resulting in smaller optimal fitness values.

Table 4 displays the results for large-scale problems. At this scale, the solution quality of the IDE-VNS algorithm has improved further compared to the DE and GA algorithms, and the relationship between solution quality and container number follows the trend shown in Table 3. The average gap with the DE algorithm (GAP_V1) is 44.49%, and with the GA algorithm (GAP_V2) it is 45.22%. However, the gap in computation time has also increased, with an average gap (GAP_C1) of −49.32% compared to the DE algorithm, and an average gap (GAP_C2) of −56.47% compared to the GA algorithm.

Fig. 16 illustrates the trend of the optimal values across the 24 experimental data points. From the graph, it is clearer that when the container task scale is small, the differences in solution quality between IDE-VNS, DE, and GA algorithms are minimal. However, as the container task scale increases, the solution quality gap between IDE-VNS and the other two algorithms becomes more pronounced. Furthermore, increasing the number of AGVs can enhance the operational efficiency of the automated terminal to some extent. The experimental results indicate that despite IDE-VNS sacrificing some computation speed, it significantly improves solution quality and can be applied to scheduling problems of various scales. This is due to the IDE-VNS algorithm's use of variable neighborhood search on the optimal fitness value of each iteration, which is the main reason for the increased computation time. However, this local search strategy effectively prevents the algorithm from getting stuck in local optima, allowing for a more precise search for the global optimum.

In this study, two experiments are conducted to examine the convergence capability of each algorithm. The result of the experiments as shown in Fig. 17. Under the experimental scale of 10 AGVs handling 50 containers, IDE-VNS obtains the optimal value around the 50th generation, while the convergence speeds of GA and DE are both lower than the algorithm in this paper. Under larger-scale experiments, the solution values of IDE-VNS tended to stabilize around the 150th generation, and new solution values were further

found at the 250th generation, indicating that this algorithm has a certain ability to escape the local optimum. In addition, it can be seen that in terms of the ability to obtain better fitness values, IDE-VNS > DE > GA.

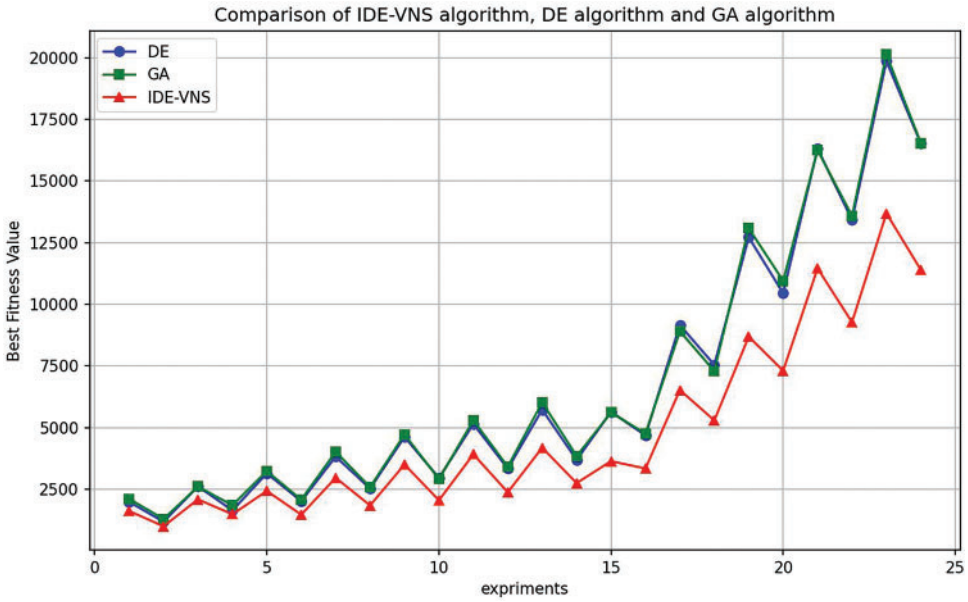

Figure 16: Performance trends of IDE-VNS, DE and GA with 24 examples

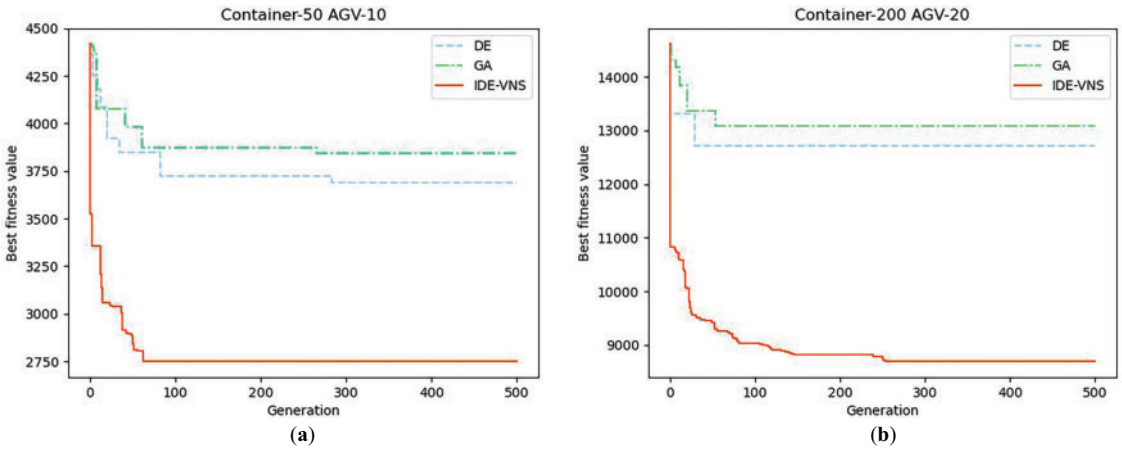

Figure 17: Typical convergence of IDE-VNS, DE and GA algorithm. (**a**) Convergence of algorithms for 10 AGVs transporting 50 containers; (**b**) Convergence of algorithms for 20 AGVs transporting 200 containers

5.4 Comparative Experiments and Analysis of Unidirectional and Bidirectional Transportation

The bidirectional transportation can make the movement of AGVs more flexible and handle container tasks more flexibly. In order to verify the above characteristics, a comparative experiment between the unidirectional and bidirectional transportation was conducted. The experimental results were obtained in Tables 5 and 6. BFV is the best fitness value and CPU time is solution time, GAP_V is the difference of BFV of the bidirectional and unidirectional transportation, GAP_C is the difference of CPU time of the bidirectional and unidirectional transportation.

Table 5: The results of different transportation comparison experiment in small-scale problems

No.	Container	AGV	Bidirection		Unidirection		GAP_V	GAP_C
			BFV	CPU Time	BFV	CPU Time		
1	20	5	1632	98.175	1896	109.343	16.18%	11.38%
2	20	10	1145	114.001	1073	116.629	−6.29%	2.31%
3	25	5	2062	123.256	2352	134.354	14.06%	9.00%
4	25	10	1292	142.045	1495	154.201	15.71%	8.56%
5	30	5	2531	163.187	2904	172.648	14.74%	5.80%
6	30	10	1431	164.989	1657	190.901	15.79%	15.71%
7	35	5	2803	187.703	3439	205.01	22.69%	9.22%
8	35	10	1950	215.944	1979	220.831	1.49%	2.26%
9	40	5	3675	243.435	3927	244.626	6.86%	0.49%
10	40	10	2147	251.566	2313	260.256	7.73%	3.45%
11	45	5	4000	286.86	4426	286.907	10.65%	0.02%
12	45	10	2460	316.896	2586	307.332	5.12%	−3.02%
13	50	5	4210	324.363	4842	334.626	15.01%	3.16%
14	50	10	2495	355.133	2839	342.5	13.79%	−3.56%
AVG_GAP							10.97%	4.63%

Notes: BFV is best fitness value, CPU time is solution time. GAP_V = (bidirection_BFV − unidirection_BFV)/bidirection_BFV, GAP_C = (bidirection_CPU_time − unidirection_CPU_time)/bidirection_CPU_time.

Table 6: The results of different transportation comparison experiment in larger-scale problems

No.	Container	AGV	Bidirection		Unidirection		GAP_V	GAP_C
			BFV	CPU Time	BFV	CPU Time		
15	100	20	3926	1431.516	3976	1226.426	1.27%	−14.33%
16	100	30	3322	1534.52	3378	1296.504	1.69%	−15.51%
17	150	20	6178	2904.311	6104	2265.118	−1.20%	−22.01%
18	150	30	5184	3185.334	5162	2507.339	−0.42%	−8.40%
19	200	20	8338	5140.373	8354	3469.233	0.19%	−32.51%
20	200	30	7128	4950.013	7173	3564.789	0.63%	−27.98%
21	250	20	10,808	8024.918	10,842	4721.084	0.31%	−41.17%
22	250	30	8912	7281.694	8872	4517.616	−0.45%	−37.96%
23	300	20	13,360	11,162.187	13,280	5881.15	−0.60%	−47.31%
24	300	30	10,996	11,039.667	11,004	5460.423	0.07%	−50.54%
AVG_GAP							0.03%	−29.77%

The results derived from Tables 5 and 6 are as follows: (1) When the number of AGVs is fixed, the BFV of unidirectional and bidirectional transportation gradually increase with the number of container tasks increases. (2) When the number of containers is fixed, the increase in the quantity of AGVs can reduce BFV of unidirectional and bidirectional transportation. (3) The BFV of bidirectional transportation is superior to that of unidirectional transportation in small-scale problem, and the average gap GAP_V is 10.97%, but the average gap GAP_V is 0.03% in larger-scale problem. (4) The average gap GAP_C is 4.63% in small-scale problem and −29.77% in larger-scale problem. Fig. 18 illustrates the trend of optimal values based on

the above experimental data. From the figure, it is clear that when the container task scale is small, the bidirectional transportation mode performs better than the unidirectional mode, and the gap between the two modes narrows as the number of AGVs increases. Specifically, in our experiments, when the number of containers is 100 or fewer, the bidirectional transportation method demonstrates a clear advantage in BFV compared to the unidirectional approach. However, as the number of container tasks increases and the problem scale expands, bidirectional transportation gradually shows certain disadvantages, this is because, as the problem size increases, conflicts become more complex, making it difficult to find better scheduling solutions than the unidirectional mode within an efficient timeframe. Additionally, due to the increased number of conflicts, more solving time is required to resolve them, resulting in longer AGV solving times for bidirectional transportation compared to unidirectional transportation in large-scale problems. Therefore, bidirectional transportation is best suited for small to medium-sized automated container terminals, where the number of container tasks is relatively low and operational conflicts are fewer, allowing this approach to effectively enhance operational efficiency.

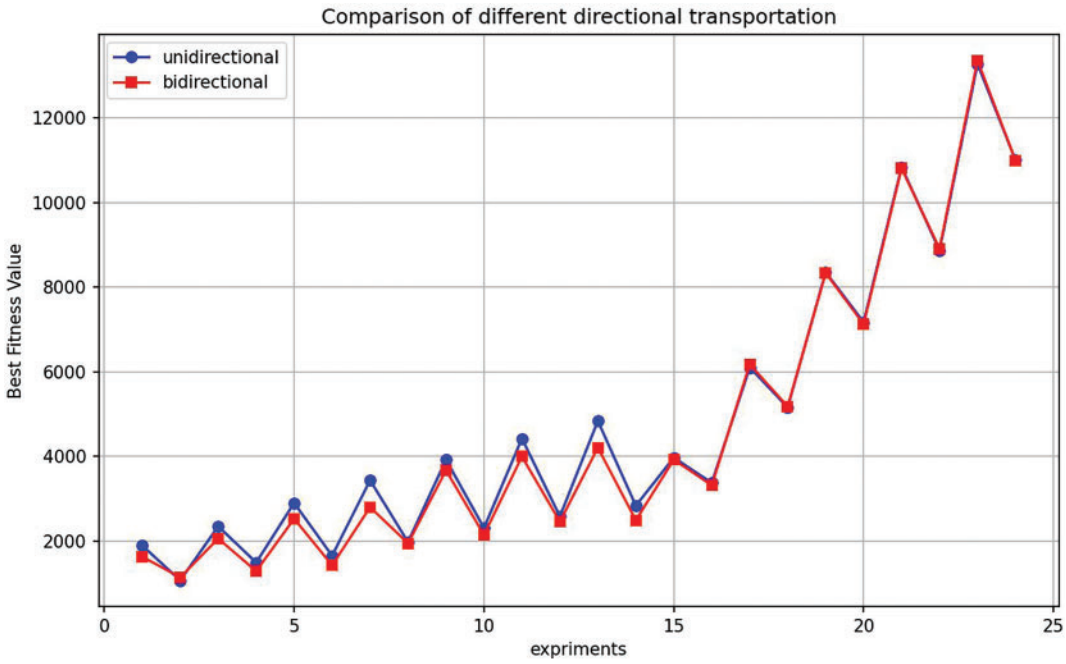

Figure 18: Performance trends of unidirectional and bidirectional transprotation

Furthermore, in order to clearly illustrate the operation process of the AGV implemented in this paper and the bidirectional conflict-free transportation, the AGV transportation routes as shown in Fig. 19 is obtained in 20 containers–5 AGVs. The Complete route as shown in Fig. 19a shows the paths of all AGVs. Fig. 19b shows in more detail the movement path of a single AGV processing task and the time relationship of interaction with the equipment. As shown in this figure, AGV3 enters node 22 from node 21 to start the container delivery operation. After YC completes the unloading operation of the above container, the new container is delivered to AGV3 and the transportation operation begins. Starting to move from node 22 back to node 21. It further demonstrates the characteristic of AGV's free movement in the horizontal transportation area. Fig. 19c,d details the conflict-free paths of AGVs. In Fig. 19c, AGV1 needs to move from node 17 to node 18 around 330 s. However, at this time, AGV4 is entering node 18 from node 19. Since node 18 has no buffer, in order to avoid conflicts, AGV1 must wait at node 17 with a buffer until AGV4 reaches node 18 first before starting to move. Then the subsequent movement of AGV4 is from node 18 to node 17,

and there is still a conflict. Therefore, AGV1 needs to continue waiting for AGV4 to move to node 17 and release the occupied paths and nodes before it can continue to move. The safe distance between AGVs is shown in Fig. 19d. When AGV4 and AGV5 reach node 7 with a buffer and both need to enter node 8, AGV5 waits in the buffer until AGV4 leaves and ensures a safe distance before restarting its movement.

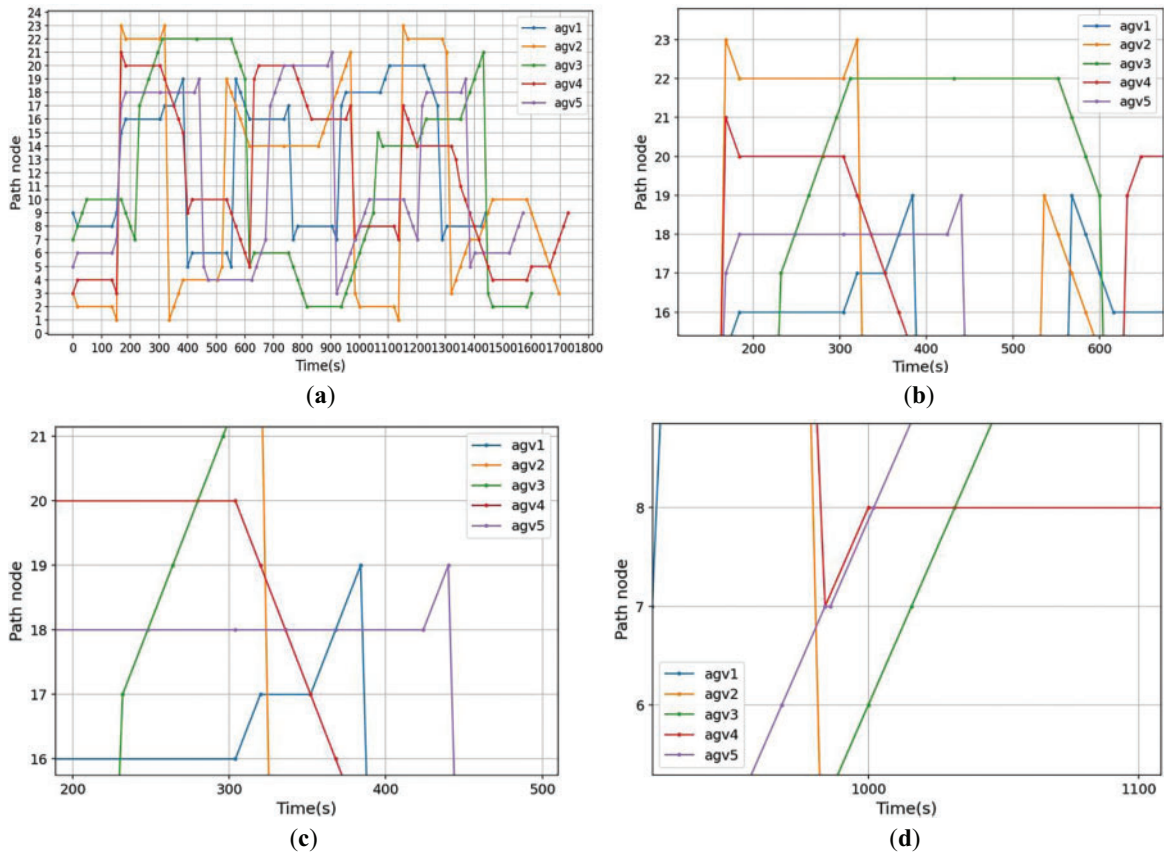

Figure 19: Route results for 5 AGVs transporting 20 containers in bidirectional transportation. (**a**) Complete result; (**b**) Bidirectional transportation of AGV; (**c**) Avoidance of reverse collision; (**d**) Avoidance of pursuit collision

5.5 Comparison and Analysis of Different Battery Swapping Strategies

To address battery management for automated guided vehicles, this study establishes two critical thresholds: a 10% minimum safety threshold to ensure AGVs can reach the battery swapping station and a 30% operational threshold that allows AGVs to complete their current tasks before recharging. Specifically, the 10% threshold guarantees sufficient power for the return trip, while the 30% threshold balances task continuity with energy conservation.

In this section, a comparative experiment was conducted on the impact of different battery swapping strategies. This paper compares the utilization rates and maximum completion times of AGVs with 30%_10% battery swap thresholds (double-thresholds strategy), 30% battery swap thresholds, and 10% battery swap thresholds under different scales of 10 and 15 AGVs transporting 200, 250, and 300 containers. The experimental results are presented in Table 7. BFV is the best fitness values. Use_rate is the use rate of AGV, which is the sum of the operation times of each AGV divided by the sum of the differences between the start and end times of each AGV. Swapping_times is the total number of battery swaps for all AGVs during the operation process.

Table 7: The result of different battery swapping strategies

No.	Container	AGV	Battery swapping strategies								
			30%_10% (This Paper)			30%			10%		
			BFV	Use_ Rate	Swapping_ Times	BFV	Use_ Rate	Swapping_ Times	BFV	Use_ Rate	Swapping_ Times
1	200	10	12,194	59.2%	10	12,556	57.1%	11	12,562	57.7%	10
2	200	15	9554	50.8%	8	10,376	48.5%	11	10,028	49.5%	5
3	250	10	15,728	58.3%	15	16,376	55.7%	18	16,314	56.2%	11
4	250	15	12,874	49.2%	15	13,394	47.3%	15	13,178	47.2%	12
5	300	10	18,914	57.5%	19	21,216	51.7%	23	19,920	53.8%	17
6	300	15	15,386	48.8%	16	16,488	46.4%	21	16,158	47.3%	15

The findings obtained from the Table 7 are as follows: (1) With a fixed number of AGVs, the BFV and the number of battery swaps increase with the increase of task scale; (2) With a fixed number of containers, as the scale of the AGV increases, the BFV increase and the number of battery swaps decrease, but it leads to a decrease in the utilization rate of AGVs; (3) BFVs with double-threshold battery swapping outperform those with fixed-threshold battery swapping in different scale problems. The experimental results in Table 7 can be summarized in Fig. 20, which compares the BFV with the number of battery swaps. From the figure, it can be observed that the double-threshold battery swapping strategy results in fewer swaps compared to the 30% fixed swapping threshold. This is because the AGV's swapping decision threshold is lower, allowing for more efficient use of the AGV's battery while ensuring it does not run out of power. In contrast to the 10% fixed swapping threshold, although more swaps are required, better optimization results are achieved. This is because, during double-threshold swapping, some swap operations take advantage of waiting intervals to perform battery changes. Although this increases the number of swaps, it prevents delays in AGV operations caused by battery swapping, ultimately improving the AGV's transportation efficiency.

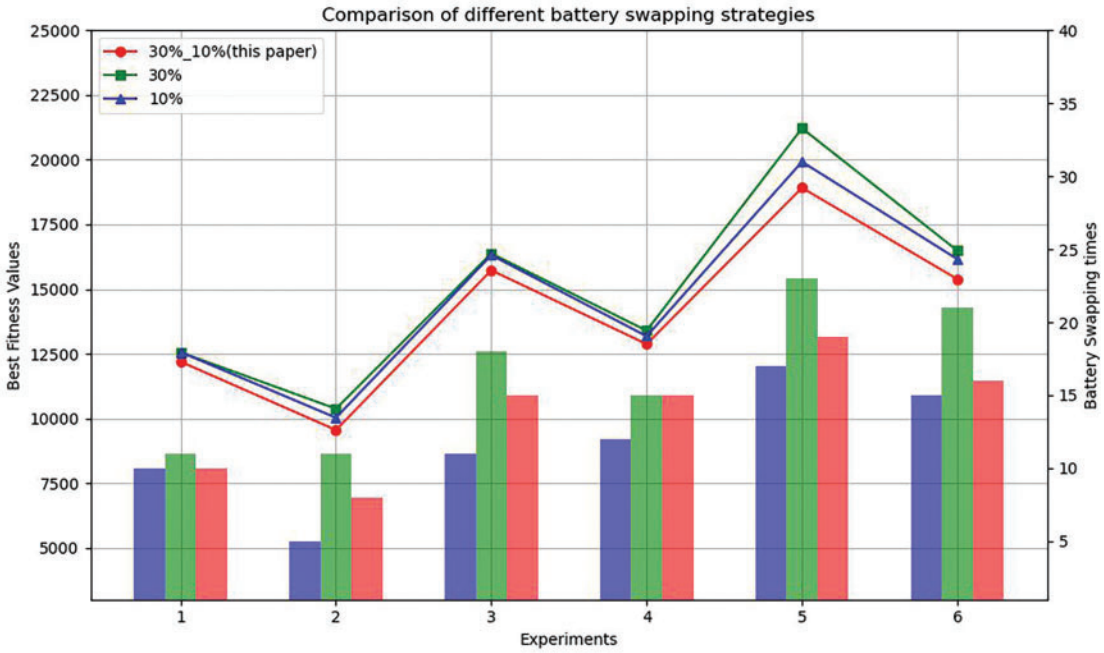

Figure 20: Comparison of different battery swapping strategies

To further verify that the advantage of double-threshold battery swapping is to utilize the waiting time of AGVs, the time consumption of various states during the operation process of AGVs under 300 container tasks is listed in Table 8. The operation time of AGV refers to the time cost of the AGV transporting container task. The waiting time of AGV refers to the time cost of AGV waiting for QC/YC to complete the loading/unloading of containers. The empty time of AGV refers to the time cost of AGV moving to receive the next task. The battery swap cost of AGV refers to the time cost of moving from the current position to the battery swapping station and swapping the battery.

Table 8: The result of cost for 300 container tasks with different battery swapping strategies

Battery swapping strategies	Operation time of AGVs	Waiting time of AGVs	Empty time of AGVs	Swapping times of AGVs	Battery swap cost of AGVs	Use rate of AGVs	BFV
30%_10%	10,7936	39,770	37,092	19	3018	56.3%	18,914
10%	10,7002	51,074	37,642	17	3188	53.8%	19,920
30%	10,9350	56,568	38,926	23	4060	53.9%	21,216

It can be obtained from Table 8 that the operation time of AGV under the double-threshold battery swapping strategy is very close to that of the other two battery swapping strategies, significantly reducing the waiting cost during the operation process. This is because the double-threshold battery swapping strategy can take advantage of the gap when the AGV is waiting for the QC/YC to complete the operation of the previous container task to go for battery swapping. Thereby, the idle time of the AGVs is reduced and their utilization rate is enhanced.

6 Conclusion

This paper discusses a combinatorial optimization problem in the ACT, where AGV scheduling, bidirectional conflict-free routing, battery swapping, and import/export container tasks are considered. A bi-level mixed integer programming (MIP) model is proposed, and the upper layer achieves the assignment of container tasks and the battery swapping planning of AGVs, while the lower layer ensures the movement of AGVs without conflict. A double-threshold battery swapping strategy is proposed and an improved differential evolution variable neighborhood search (IDE-VNS) algorithm is developed to solve the bi-level MIP model to minimize the completion time of all jobs. The result of the transportation mode experiment indicates that the bidirectional transportation of AGVs represents an efficient means to enhance the operational efficiency of AGVs. This is because of its more adaptable route selection and shorter driving route. Besides, the complicated routes of the bidirectional transportation have a substantial impact on AGV operations when operating at large container terminals, making it challenging to find an optimal schedule within the time limit. Therefore, the bidirectional transportation is more effectively applied to the small-scale ACT. The result of the battery swapping strategy experiment indicates that the double-threshold battery swapping strategy can reasonably arrange the battery swapping timing of AGVs, improve AGV utilization, and enhance operational efficiency. The result of the algorithm comparison experiment indicates that the IDE-VNS algorithm can better search for the optimal solution and avoid getting trapped in local optima.

Further research can be more consistent with the actual operation of ACT from the following aspects: (1) Consider the variable speed of AGVs. During the actual transportation process, the speed of AGVs is not constant. Therefore, issues such as AGV deceleration to avoid conflicts and variable speed during start and stop in the dispatching process can be further considered. (2) Further consider the collaborative operation

between AGV and other equipment in the automated terminal, such as the collaboration between AGV and QC/YC. (3) Consider the impact of uncertainty problems in ACT, such as additional task arrivals, task cancellations, and AGV failures.

Acknowledgement: The authors gratefully acknowledge the National Natural Science Foundation of China and the Shanghai Science and Technology Commission for supporting this study.

Funding Statement: This work was supported by National Natural Science Foundation of China (No. 62073212), Shanghai Science and Technology Commission (No. 23ZR1426600).

Author Contributions: He Huang: Methodology, software, validation, investigation, resources, writing—original draft, visualization; Jin Zhu: conceptualization, formal analysis, writing—review and editing, supervision, project administration, funding acquisition. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data generated in this paper are available from the corresponding author on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Sun PZ, You J, Qiu S, Wu EQ, Xiong P, Song A, et al. AGV-based vehicle transportation in automated container terminals: a survey. IEEE T Intell Transp. 2022;24(1):341–56. doi:10.1109/tits.2022.3215776.
2. Tian Y, Wang J, Chen J, Fan H. Collaborative scheduling of QCs, L-AGVs and ARMGs under mixed loading and unloading mode in automated terminals. J Shanghai Marit Univ. 2018;39(3):14–21. doi:10.13340/j.jsmu.2018.03.003.
3. Liu Q, Wang N, Li J, Ma T, Li F, Gao Z. Research on flexible job shop scheduling optimization based on segmented AGV. Comput Model Eng Sci. 2023;134(3):2073. doi:10.32604/cmes.2022.021433.
4. Rahman HF, Janardhanan MN, Nielsen P. An integrated approach for line balancing and AGV scheduling towards smart assembly systems. Assembly Autom. 2020;40(2):219–34. doi:10.1108/aa-03-2019-0057.
5. Zhang Z, Chen J, Guo Q. Application of automated guided vehicles in smart automated warehouse systems: a survey. Comput Model Eng Sci. 2023;134(3):1529. doi:10.32604/cmes.2022.021451.
6. Yang Y, Zhong M, Dessouky Y, Postolache O. An integrated scheduling method for AGV routing in automated container terminals. Comput Ind Eng. 2018;126:482–93. doi:10.1016/j.cie.2018.10.007.
7. Adamo T, Bektaş T, Ghiani G, Guerriero E, Manni E. Path and speed optimization for conflict-free pickup and delivery under time windows. Transport Sci. 2018;52(4):739–55. doi:10.1287/trsc.2017.0816.
8. Zhong M, Yang Y, Dessouky Y, Postolache O. Multi-AGV scheduling for conflict-free path planning in automated container terminals. Comput Ind Eng. 2020;142:106371. doi:10.1016/j.cie.2020.106371.
9. Wang Z, Zeng Q. A branch-and-bound approach for AGV dispatching and routing problems in automated container terminals. Comput Ind Eng. 2022;166:107968. doi:10.1016/j.cie.2022.107968.
10. Cao Y, Yang A, Liu Y, Zeng Q, Chen Q. AGV dispatching and bidirectional conflict-free routing problem in automated container terminal. Comput Ind Eng. 2023;184:109611. doi:10.1016/j.cie.2023.109611.
11. Yang X, Hu H, Jin J. Battery-powered automated guided vehicles scheduling problem in automated container terminals for minimizing energy consumption. Ocean Coast Manage. 2023;246:106873. doi:10.1016/j.ocecoaman.2023.106873.
12. Farrell BR, McKie R. Designing a battery exchange building for automated guided vehicles. In: Ports 2016. Reston, VA, USA: ASCE Library. p. 71–80. doi:10.1061/9780784479919.008.

13. Zou B, Xu X, De Koster R. Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system. Eur J Oper Res. 2018;267(2):733–53. doi:10.1016/j.ejor.2017.12.008.
14. Xiao S, Huang J, Hu H, Gu Y. Automatic guided vehicle scheduling in automated container terminals based on a hybrid mode of battery swapping and charging. J Mar Sci Eng. 2024;12(2):305. doi:10.3390/jmse12020305.
15. Zhou X, Zhu J. An improved bounded conflict-based search for multi-AGV pathfinding in automated container terminals. Comput Model Eng Sci. 2024;139(3):2705–27. doi:10.32604/cmes.2024.046363.
16. Kim KH, Jeon SM, Ryu KR. Deadlock prevention for automated guided vehicles in automated container terminals. OR Spectr. 2006;28:659–79. doi:10.1007/978-3-540-49550-5_12.
17. Xiang X, Liu C, Lee LH, Chew EP. Performance estimation and design optimization of a congested automated container terminal. IEEE T Autom Sci Eng. 2022;19(3):2437–49. doi:10.1109/tase.2021.3085329.
18. Shen Z, Liang C, Gen M. Scheduling of AGV with group operation area in automated terminal by hybrid genetic algorithm. In: Proceedings of the Fifteenth International Conference on Management Science and Engineering Management; 2021 Aug 1–3; Harbin, China. Berlin/Heidelberg, Germany: Springer; 2021. p. 427–42. doi:10.1007/978-3-030-79203-9_33.
19. Yue L, Fan H. Dynamic scheduling and path planning of automated guided vehicles in automatic container terminal. IEEE/CAA J Autom Sin. 2022;9(11):2005–19. doi:10.1109/jas.2022.105950.
20. Gao Y, Chen C-H, Chang D. A machine learning-based approach for multi-AGV dispatching at automated container terminals. J Mar Sci Eng. 2023;11(7):1407. doi:10.3390/jmse11071407.
21. Lou P, Zhong Y, Hu J, Fan C, Chen X. Digital-twin-driven AGV scheduling and routing in automated container terminals. Math-Basel. 2023;11(12):2678. doi:10.3390/math11122678.
22. Zhong Z, Guo Y, Zhang J, Yang S. Energy-aware integrated scheduling for container terminals with conflict-free AGVs. J Syst Sci Syst Eng. 2023;32(4):413–43. doi:10.1007/s11518-023-5563-y.
23. Liu W, Zhu X, Wang L, Wang S. Multiple equipment scheduling and AGV trajectory generation in U-shaped sea-rail intermodal automated container terminal. Measurement. 2023;206:112262. doi:10.1016/j.measurement.2022.112262.
24. Xu Z, Li J, Li H, An L, Hu J. A conflict-free dispatching method for Aivs in automated container terminals. In: Proceedings of the 2023 China Automation Congress (CAC); 2023 Nov 17–19; Chongqing, China. Piscataway, NJ, USA: IEEE; 2023. p. 7707–12. doi:10.1109/cac59555.2023.10451308.
25. Li S, Fan L, Jia S. A hierarchical solution framework for dynamic and conflict-free AGV scheduling in an automated container terminal. Transport Res C-Emer. 2024;165:104724. doi:10.1016/j.trc.2024.104724.
26. Xiong C, Wang C, Zhou S, Song X. Dynamic rolling scheduling model for multi-AGVs in automated container terminals based on spatio-temporal position information. Ocean Coast Manage. 2024;258:107349. doi:10.1016/j.ocecoaman.2024.107349.
27. Terfasse K, Cherif G, Huguet M-J. Integer linear programming for automated guided vehicles path planning in container terminals. In: Proceedings of the 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT); 2024 Apr 10–13; Vallette, Malta. Piscataway, NJ, USA: IEEE; 2024. p. 940–5. doi:10.1109/codit62066.2024.10708293.
28. Ma N, Zhou C, Stephen A. Simulation model and performance evaluation of battery-powered AGV systems in automated container terminals. Simul Model Pract Th. 2021;106:102146. doi:10.1016/j.simpat.2020.102146.
29. Xiang X, Liu C. Modeling and analysis for an automated container terminal considering battery management. Comput Ind Eng. 2021;156:107258. doi:10.1016/j.cie.2021.107258.
30. Zhao T, Liang C, Hu X, Wang Y. Solution of AGV scheduling and battery exchange two layer model for automated container terminal. J Dalian Univ Technol. 2021;61:623–33. doi:10.7511/dllgxb202106010.
31. Yang X, Hu H, Cheng C, Wang Y. Automated guided vehicle (AGV) scheduling in automated container terminals (ACTs) focusing on battery swapping and speed control. J Mar Sci Eng. 2023;11(10):1852. doi:10.3390/jmse11101852.

32. Li L, Li Y, Liu R, Zhou Y, Pan E. A two-stage stochastic programming for AGV scheduling with random tasks and battery swapping in automated container terminals. Transport Res E-Log. 2023;174:103110. doi:10.1016/j.tre.2023.103110.
33. Li Y, Li L, Liu R, Pan E. An integrated scheduling method for AGVs in an automated container terminal considering battery swapping and battery degradation. Flex Serv Manuf J. 2024;35:1–48. doi:10.1007/s10696-024-09581-7.
34. Zhou W, Zhang Y, Tang K, He L, Zhang C, Tian Y. Co-optimization of the operation and energy for AGVs considering battery-swapping in automated container terminals. Comput Ind Eng. 2024;195:110445. doi:10.1016/j.cie.2024.110445.