mathematics

MDPI

Article

Digital-Twin-Driven AGV Scheduling and Routing in Automated Container Terminals

Ping Lou, Yutong Zhong, Jiwei Hu, Chuannian Fan and Xiao Chen *

School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China
* Correspondence: chenxiao_2021@whut.edu.cn

Abstract: Automated guided vehicle (AGV) scheduling and routing are critical factors affecting the operation efficiency and transportation cost of the automated container terminal (ACT). Searching for the optimal AGV scheduling and routing plan are effective and efficient ways to improve its efficiency and reduce its cost. However, uncertainties in the physical environment of ACT can make it challenging to determine the optimal scheduling and routing plan. This paper presents the digital-twin-driven AGV scheduling and routing framework, aiming to deal with uncertainties in ACT. By introducing the digital twin, uncertain factors can be detected and handled through the interaction and fusion of physical and virtual spaces. The improved artificial fish swarm algorithm Dijkstra (IAFSA-Dijkstra) is proposed for the optimal AGV scheduling and routing solution, which will be verified in the virtual space and further fed back to the real world to guide actual AGV transport. Then, a twin-data-driven conflict prediction method is proposed to predict potential conflicts by constantly comparing the differences between physical and virtual ACT. Further, a conflict resolution method based on the Yen algorithm is explored to resolve predicted conflicts and drive the evolution of the scheme. Case study examples show that the proposed method can effectively improve efficiency and reduce the cost of AGV scheduling and routing in ACT.

Keywords: digital-twin-driven; AGV scheduling and routing; conflict prediction; conflict resolution; IAFSA-Dijkstra

MSC: 68Txx

Citation: Lou, P.; Zhong, Y.; Hu, J.; Fan, C.; Chen, X. Digital-Twin-Driven AGV Scheduling and Routing in Automated Container Terminals. *Mathematics* **2023**, *11*, 2678. https://doi.org/10.3390/math11122678

Academic Editors: Fan Zhang, Songhe Feng, Yongsheng Zhou and Junlin Hu

Received: 28 April 2023
Revised: 26 May 2023
Accepted: 29 May 2023
Published: 13 June 2023

1. Introduction

In recent years, economic globalization has led to an increase in international trade, and maritime transport has become an essential means of transportation. Ports serve as a crucial hub for both sea and land transportation and play an integral role in facilitating international trade. However, with the growing demand for maritime transportation, ports are under immense pressure to increase efficiency and reduce costs. To address these challenges, the development of automated container terminals (ACTs) has emerged as a new trend. ACTs leverage advanced technologies, such as robotics, artificial intelligence, and the Internet of Things, to automate container handling processes. This technology can significantly increase port productivity, reduce labor costs, and improve operational efficiency. Automatic guided vehicles (AGVs) are significant types of equipment in an ACT, which can walk along a pre-set guidance path to complete a series of horizontal transport operations [1]. AGV scheduling and routing are crucial problems in the ACT. The purpose of AGV scheduling is to allocate a batch of container transportation tasks to a set of AGVs and specify the sequence and time of task execution to achieve specific goals under given constraints. AGV routing is to determine an optimal path from the start position to the end position for the AGV, complete AGV's assigned transport tasks in order, and avoid collision with other AGVs. Reasonable scheduling and routing for AGVs can not only improve the efficiency of ACT operation but also reduce the traffic accident rate in the ACT

and improve the reliability of transportation [2]. Therefore, AGV scheduling and routing are considered comprehensively in this work.

Research on AGV scheduling and routing in an ACT starts with static problems, exploring optimal or near-optimal solutions under deterministic ACT environments. There are many studies about AGV static problems. Luo and Wu [3] develop an integrated modeling approach for AGV and YC static scheduling problems in ACTs that minimizes the ship's berth time; they design a genetic algorithm to solve the problem. Xu et al. [4] propose a reinforcement learning-based hyper-heuristic genetic algorithm to solve the integrated static scheduling problem of AGVs and other facilities in the U-shaped ACT; their method can avoid AGV conflicts. Lu et al. [5] propose an ant colony system-improved grey wolf optimization to solve the fourth-party logistics routing problem. Yan et al. [6] present a hybrid metaheuristic algorithm of discrete particle swarm optimization and Harris Hawks optimization to solve the location problem and vehicle routing problem. Lu et al. [7] design a bi-level whale optimization algorithm to solve a bi-level multi-objective schedule risk management model. However, a real-world ACT environment is full of uncertain events, such as AGV failure, ship arrival time delay, and weather change; thus, the static problem is impractical. In static problems, the mathematical model establishment usually relies on predefined constraints (e.g., constant AGV speed). However, the occurrence of uncertain events can destroy these constraints, which causes the scheme to become unfeasible and even induce AGV conflicts.

In order to respond to uncertain events, dynamic problems are developed. Furthermore, an increasing number of researchers are beginning to study dynamic problems. For example, in order to respond to the uncertainty caused by new job arrival, Cai et al. [8] propose a rescheduling new arrival jobs (RNJ) policy and rescheduling combination of new and unexecuted jobs (RCJ) policy for container-transportation task allocation in the ACTs. Jian et al. [9] develop a multi-objective scheduling model that uses a symmetric triangular fuzzy number to describe the AGVs' operation time distribution; they build an improved genetic algorithm to solve the problem effectively. Yue and Fan [10] introduce a dynamic scheduling process to respond to the QC waiting caused by AGVs' delay in an uncertain environment.

Uncertain events can trigger a series of chain reactions that can disrupt the entire ACT operation, causing deterioration of ACT efficiency [11]. Therefore, a timely response to dynamic events during the plan execution becomes a vital issue that needs to be addressed urgently. The emergence of the digital twin (DT) provides a new idea to meet the above challenges. DT is an integrated system that can simulate, monitor, calculate, regulate, and control the system status and process [12], with the characteristics of real-time reflection, interaction and convergence, and evolution and iteration.

This work proposes a digital-twin-driven AGV scheduling and routing framework that copes with uncertainties in automated container terminals. This paper's main contributions are as follows:

(1) A digital-twin-driven AGV scheduling and routing framework is proposed. Based on this framework, an initial scheduling and routing plan is generated first. The AGV transport process is continuously monitored, and dynamic events are fed back into the physical space, enabling timely responses to changes in the environment.
(2) The improved artificial fish swarm algorithm Dijkstra (IAFSA-Dijkstra) is proposed to solve the bi-level mixed integer programming model and obtain an optimal solution. The task combination encoding is presented to reduce the encoding length of the IAFSA-Dijkstra, and the adaptive parameter adjustment operator is used to improve the global optimization capability of the IAFSA-Dijkstra. The optimal plan is verified and fed into the physical space to guide AGV transport.
(3) Twin-data-driven conflict prediction and resolution: A twin-data-driven AGV conflict prediction method is explored to predict conflicts by comparing physical and virtual data. A conflict resolution method based on the Yen algorithm is presented to resolve these conflicts and drive timely revision and evolution of the initial plan.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 describes the AGV scheduling and routing problem and establishes a bi-level mixed integer programming model. The DT-based AGV scheduling and routing framework is discussed in Section 4. Section 5 gives examples to verify the effectiveness of the proposed method. Finally, Section 6 concludes the paper and points out future works.

2. Related Works

2.1. AGV Scheduling

There have been many studies on the AGV scheduling problem in recent years. A brief review of previous works is provided on AGV scheduling.

Existing studies on AGV static scheduling in the ACT have provided very valuable information. Rashidi and Tsang [13] propose a novel algorithm, NSA+, to solve the AGV scheduling problem considering minimizing const flow in container terminals. Ma et al. [14] used an improved mathematical model to describe the multi-load AGV scheduling problem in the ACT; they built a shuffled frog leaping algorithm with a mutant process (SFLAMUT) to solve the problem. Given the high potential synergy between automated quayside cranes (AQCs) and AGVs, Zhao et al. [15] built a collaborative model for AQCs and AGVs; they used a two-stage taboo search algorithm to solve the scheduling problem.

AGV static scheduling usually is infeasible and impractical due to uncertainties; thus, AGV dynamic scheduling has become a growing concern. Angeloudis and Bell [16] propose a novel algorithm structured around a cost/benefit concept to solve the AGV dis-patching problem under an indeterministic ACT environment, mainly including uncertain task durations for future events. Xin et al. [17] propose a rescheduling method to rearrange AGVs, quay cranes (QCs), and automated stacking cranes (ASCs) in ACT, including two types of methods: the time-efficient schedule and the energy-efficient schedule. Sahin et al. [18] develop a multi-agent-based system to simultaneously schedule flexible machine groups and AGVs under an uncertain manufacturing environment. Xu et al. [19] present a response method with AGVs based on the mode of "request-scheduling-response" to solve the logistics dynamic scheduling problem in an intelligent manufacturing workshop (IMW), which aims to minimize the finish time with the minimum AGVs and limited time. In order to deal with the AGV scheduling problem in complex material handling in smart factories, Zhang et al. [20] develop a dynamic scheduling method for self-organized AGVs (SAGV), aiming to minimize the delay and reduce the cost of logistics systems.

2.2. AGV Routing

The earliest research on AGV routing starts with single-AGV path planning. For the single robot path planning problem, Wang et al. [21] improve the traditional rapidly-exploring random tree (RRT) algorithm and propose an autonomous routing algorithm of node control (NC-RRT) with higher exploration efficiency and faster convergence speed. Lee and Jeong [22] use the Q-Learning and Dyna-Q algorithms to solve the mobile robot path planning problem in a warehouse environment.

In the actual environment, the situation of using one AGV to complete the task is extremely rare; multiple AGVs complete tasks in cooperation with each other. There have been many previous studies on multi-AGV path planning in a static environment. Oboth et al. [23] present a route-generation technique that can realize conflict-free path planning for multiple AGVs with varying speeds. Yuan et al. [24] propose a bi-level path planning algorithm; global and local paths are obtained using the A* and RRT algorithms, respectively.

The above research studies focus on routing problems in a static environment. In order to respond to uncertainties, dynamic routing is developed. Xu et al. [25] present an improved dynamic window method to deal with the routing problem in a three-dimensional dynamic environment, outperforming traditional DWA algorithms in terms of efficiency, smoothness, and security. To deal with complex and variable situations during the multi-robot operation process, Bae et al. [26] combine deep Q-learning with the CNN (convolution neural network) algorithm, which is more flexible and efficient than conventional methods.

In order to respond to dynamic obstacles and incomplete maps, Bai et al. [27] combine an improved Q-learning path optimization algorithm and an improved genetic algorithm; the proposed method has favorable performance in narrow working environments and highly congested situations. Onoufriou et al. [28] propose a new hybrid parallelism deep learning framework, which can provide a solution and unification of deep learning techniques using a common interface.

Some researchers have begun to study the AGV scheduling and routing problems simultaneously. Zhong et al. [29] tried to minimize ships' loading and unloading time via integrated scheduling of AGVs and other facilities with the hybrid GA-PSO algorithm with adaptive autotuning approaches by a fuzzy control solution. Fazlollahtabar et al. [30] studied the simultaneous scheduling and routing problem for AGVs in flexible manufacturing systems, obtaining the optimal plan by a modified network simplex algorithm (NSA). Miyamoto and Inoue [31] propose local/random search methods to deal with the dispatching and routing problem for capacitated AGV systems (DCFRPC). Desaulniers et al. [32] present an exact method for AGV dispatching and conflict-free routing in the flexible manufacturing system, implicitly considering congestion and blocking problems. Xing et al. [33] propose a novel tabu search algorithm to solve the conflicts that happen when multiple AGVs work at the same time. Liang et al. [34] present a three-stage integrated scheduling algorithm for AGV routing planning to deal with the locking problem of AGV.

2.3. Digital Twin

The above works have made an outstanding contribution to the AGV scheduling and routing problem but fail to address the challenges confronting AGV scheduling and routing mentioned in Section 1. Recently, DT has been widely applied in various fields, including predictive maintenance [35], factory design [36], assembly [37], prognostics [38], health care [39], traffic management [40], and so on.

Some researchers attempt to combine DT with other methods to address scheduling problems. For example, by introducing DT, Zhang et al. [41] address three critical problems in dynamic job-shop scheduling: machine availability prediction, disturbance detection, and performance evaluation. Wang and Wu [42] develop a planning and scheduling system by combining DT with planning and scheduling to effectively manage and control the disturbances in workshop scheduling. Negri et al. [43] present a DT-based scheduling framework, which includes equipment health predictions in the scheduling activity; their framework embeds a field-synchronized Equipment Health Indicator module into the DT simulation to synchronize the simulation model to the physical system. In AGV scheduling, Han et al. [44] developed a DT-based dynamic AGV scheduling (DTDAS) method, which can reasonably arrange the charging process to effectively improve AGV transportation efficiency in a workshop.

Several researchers have started to apply DT to routing problems. For example, Wang et al. [45] built a digital twin platform for gantry robots and present a multi-objective three-dimensional routing method for gantry robots based on the NavMesh algorithm based on the platform, improving production efficiency and safety. Zohdi [46] designed a digital twin framework for firefighting to help aerial first responders rapidly perform flight routing in risky fire environments. Guo et al. [47] constructed a digital twin system for product assembly and present a modified Q-learning algorithm to solve the routing problem based on the system. To address the shortcoming of supervised learning of requiring large-scale training data, Vasanthan and Nguyen [48] introduced a digital twin of the vessel to generate enough training data to solve the vessel's routing problem, effectively detecting potential collisions. Gao et al. [49] developed a digital-twin-enabled automated storage yard scheduling framework for uncertain port dispatching.

DT brings a chance to deal with the AGV scheduling and routing problem. By introducing DT technology, the real-time data generated by the ACT operation process can be fused with data from the virtual space to drive the dynamic evolution of the initial plan

first, and then the novel plan is simulated, verified, and, further, timely transmitted to the physical space to control the AGV operation.

3. Problem Description and Formulation

3.1. AGV Scheduling and Routing Problem

The devices of the ACT in AGV scheduling and routing involve AGVs, quay cranes (QCs), and yard cranes (YCs) (see Figure 1). AGVs are used for moving containers between QC and storage yards. QCs are to deliver containers on AGVs from a ship or containers from AGVs to a ship. The storage yard is an area for the temporary storage of containers for transferring to destinations. YCs are used to deliver containers to their storage locations in storage yards or from their storage locations in storage yards.

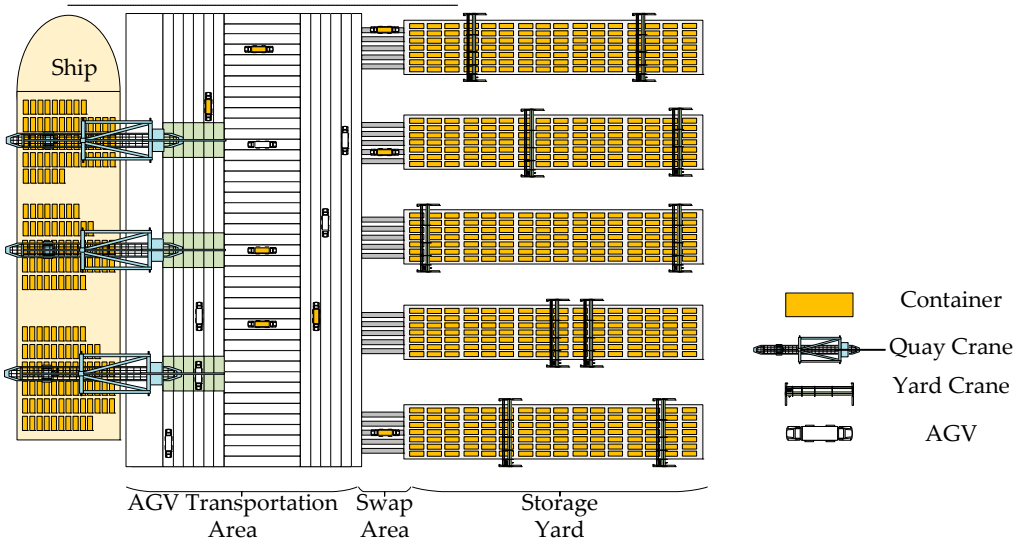
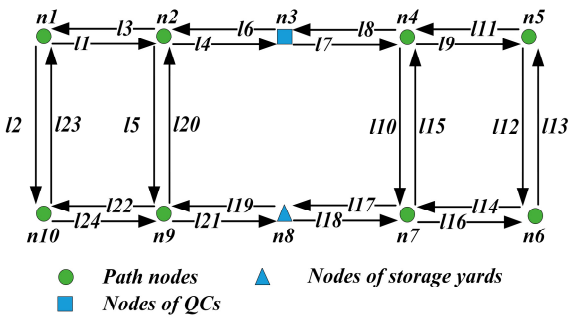


Figure 1. The layout of an automated container terminal.

The AGV transferring area has a regular route. In order to conveniently describe the AGV scheduling and routing problem of an ACT, the transferring area is formally described as a weighted directed graph, in which there are 10 nodes (see Figure 2a). The node $n3$ indicates the QC, the node $n8$ is the storage yard, the set $\{n1, n2, n4, n5, n6, n7, n9, n10\}$ is the path node set, and the set $\{l1, l2, \ldots, l24\}$ is the path set, whose lengths are $d1$, $d2$, $\ldots$, $d24$, respectively. The direction of each arrow expresses the direction of AGV that they can travel on the path. Figure 2b shows the corresponding adjacency matrix, in which the symbol ∞ means there are no ways between two nodes.



(**a**)

nodes	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10
n1	∞	d1	∞	∞	∞	∞	∞	∞	∞	d2
n2	d3	∞	d4	∞	∞	∞	∞	∞	d5	∞
n3	∞	d6	∞	d7	∞	∞	∞	∞	∞	∞
n4	∞	∞	d8	∞	d9	∞	d10	∞	∞	∞
n5	∞	∞	∞	d11	∞	d12	∞	∞	∞	∞
n6	∞	∞	∞	∞	d13	∞	d14	∞	∞	∞
n7	∞	∞	∞	d15	∞	d16	∞	d17	∞	∞
n8	∞	∞	∞	∞	∞	∞	d18	∞	d19	∞
n9	∞	d20	∞	∞	∞	∞	∞	d21	∞	d22
n10	d23	∞	∞	∞	∞	∞	∞	∞	d24	∞

(**b**)

Figure 2. The weighted directed graph and adjacent matrix. (**a**) A weighted directed graph; (**b**) an adjacent matrix.

The AGV scheduling and routing problem is satisfied under the following assumptions:

- The location of all containers' delivery points and pickup points are fixed and known. This work does not distinguish whether a task is a loading or an unloading task.
- All AGVs are in good condition without failure.
- Each AGV can only transport one container at a time, and each container can only be assigned to one AGV for transporting.
- The operation time for both QCs and YCs is fixed.
- All AGVs are homogeneous and travel at the same velocity, and AGV speed remains unchanged during turning.
- All containers have the same priority.
- The initial position of each AGV is known.

Notations used in the model of the studied problem are listed in Appendix A.

3.2. The bi-Level Mixed Integer Programming Model

In order to describe the AGV scheduling and routing problem, the bi-level mixed integer programming model is established. The upper-level scheduling model focuses on AGV scheduling and the lower-level routing model deals with AGV routing. The upper-level scheduling model determines the overall horizontal transportation operation optimization results; the lower-level route model determines the actual completion time of each operation task for each AGV.

3.2.1. The Upper-Level Scheduling Model

The objective of AGV scheduling is to minimize the completion time of all tasks. The AGV scheduling optimization model can be formulated as follows:

$$minC_{max} = \min_{k \in V} \max (C_k) \tag{1}$$

Subject to:

$$C_k = t_f^k, \ \forall k \in V \tag{2}$$

$$\sum_{k \in V} \alpha_p^k = 1, \ \forall p \in C \tag{3}$$

$$\sum_{p \in C} \beta_{0p}^k = 1, \ \forall k \in V \tag{4}$$

$$\sum_{p \in C} \beta_{pf}^k = 1, \ \forall k \in V \tag{5}$$

$$\sum_{p' \in C \cup \{0,f\}} \beta_{pp'}^k = \sum_{p' \in C \cup \{0,f\}} \beta_{p'p}^k, \ \forall p \in C, \ \forall k \in V \tag{6}$$

$$T_{O_k L_p}^k \le t_p^k + M\left(1 - \beta_{0p}^k\right), \ \forall p \in C, \ \forall k \in V \tag{7}$$

$$t_p^k + \tau_L^p + T_{L_p U_p}^k + \tau_U^p + T_{U_p L_{p'}}^k \le t_{p'}^k + M\left(1 - \beta_{pp'}^k\right), \ \forall p \in C, \ p' \in C \cup \{f\}, \ \forall k \in V \tag{8}$$

$$\alpha_p^k, \beta_{pp'}^k \in (0,1), \forall p, p' \in C, \ \forall k \in V \tag{9}$$

$$t_p^k \ge 0, \tau_L^p > 0, \tau_U^p > 0, T_{L_p U_p} > 0, T_{U_p L_{p'}} \ge 0, T_{O_k L_p} \ge 0, \forall p, p' \in C, \ \forall k \in V \tag{10}$$

Equation (1) is the optimization goal to minimize the completion time of all tasks.

Equation (2) indicates that the task completion time for each AGV is the completion time of its last task, which is the start time of its virtual last task.

Equation (3) indicates that each container is transported by one and only one AGV.

Equation (4) and equation (5) ensure that the initial task of each AGV is virtual task *0* and the ending task is virtual task *f*.

Equation (6) indicates the order in which each AGV performs its tasks. For each AGV, there is only one container transport task before and after each container transport task.

Equations (7) and (8) reflect the time correlation between the two adjacent tasks performed by the same AGV. Equation (7) indicates that the start moment of the first task for the AGV is the time from its start position to the pickup point of its first task. Equation (8) indicates that the start moment of one task for the AGV is the time from its start position to the pickup point of its first task. The start moment of the current task is equal to the sum of the start moment of the previous task, the loading time of the previous task, the transport time of the previous task, the unloading time of the previous task, and the time from the delivery point of the previous task to the pickup point of the current task.

Equation (9) is binary constraints for some decision variables.

Equation (10) is the non-negativity constraints for some variables.

3.2.2. The Lower-Level Routing Model

The objective of AGV routing is to minimize the delivery time from the start node to the end one. The AGV routing optimization model can be formulated as follows:

$$min \sum_{(i,j)\in G} \left(\mu_{ij}^{k} \times t_{ij}^{k}\right),\ \forall i,j \in N,\ \forall k \in V \tag{11}$$

Subject to:

$$t_{ij}^{k} = t_{out,ij}^{k} - t_{in,ij}^{k},\ \forall (i,j) \in G, \forall k \in V \tag{12}$$

$$t_{out,ij}^{k} \geq t_{in,ij}^{k} + d_{ij}/v_0, \forall (i,j) \in G, \forall k \in V \tag{13}$$

$$\sum_{(j,i)\in G} \mu_{ji}^{k} - \sum_{(i,j')\in G} \mu_{ij'}^{k} = \begin{cases} 1,\ i = s \\ -1,\ i = e \\ 0,\ o.w. \end{cases}, \forall k \in V \tag{14}$$

$$t_{in,si}^{k} \geq t_{in,sj}^{k'} + D_s/v_0, \forall s \in N_P, \forall k,k' \in V, \forall (s,i),(s,j) \in G \tag{15}$$

$$\mu_{ij}^{k} \in (0,1), \forall (i,j) \in G, \forall k \in V \tag{16}$$

$$t_{ij}^{k} > 0, t_{out,ij}^{k} \geq 0, t_{in,ij}^{k} \geq 0, \forall (i,j) \in G, \forall k \in V \tag{17}$$

Equation (11) is the optimization objective to minimize the delivery time from the start node to the destination one.

Equation (12) indicates that the travel time of the AGV on one path is the time difference between the moment of leaving that path and the moment of entering that path.

Equation (13) reflects the time correlation of one AGV entering and leaving one path.

Equation (14) places restrictions on the out-in degree of nodes in the current transport process. For the start node *s* of the current transport process, the out-degree is greater than the in-degree by 1. For the end node *e* of the current transport process, the in-degree is greater than the out-degree by 1. For other nodes, the out-degree and the in-degree are equal.

Equation (15) is related to AGV conflict. Equation (15) indicates that a safe distance should be maintained when AGVs visit the same path node.

Equation (16) is binary constraints for the decision variable.

Equation (17) is the non-negativity constraints for some variables.

4. Digital-Twin-Driven Scheduling and Routing

4.1. Overall Framework

The framework of DT-based AGV scheduling and routing for an ACT consists of two parts, involving the physical space and virtual space (see Figure 3).

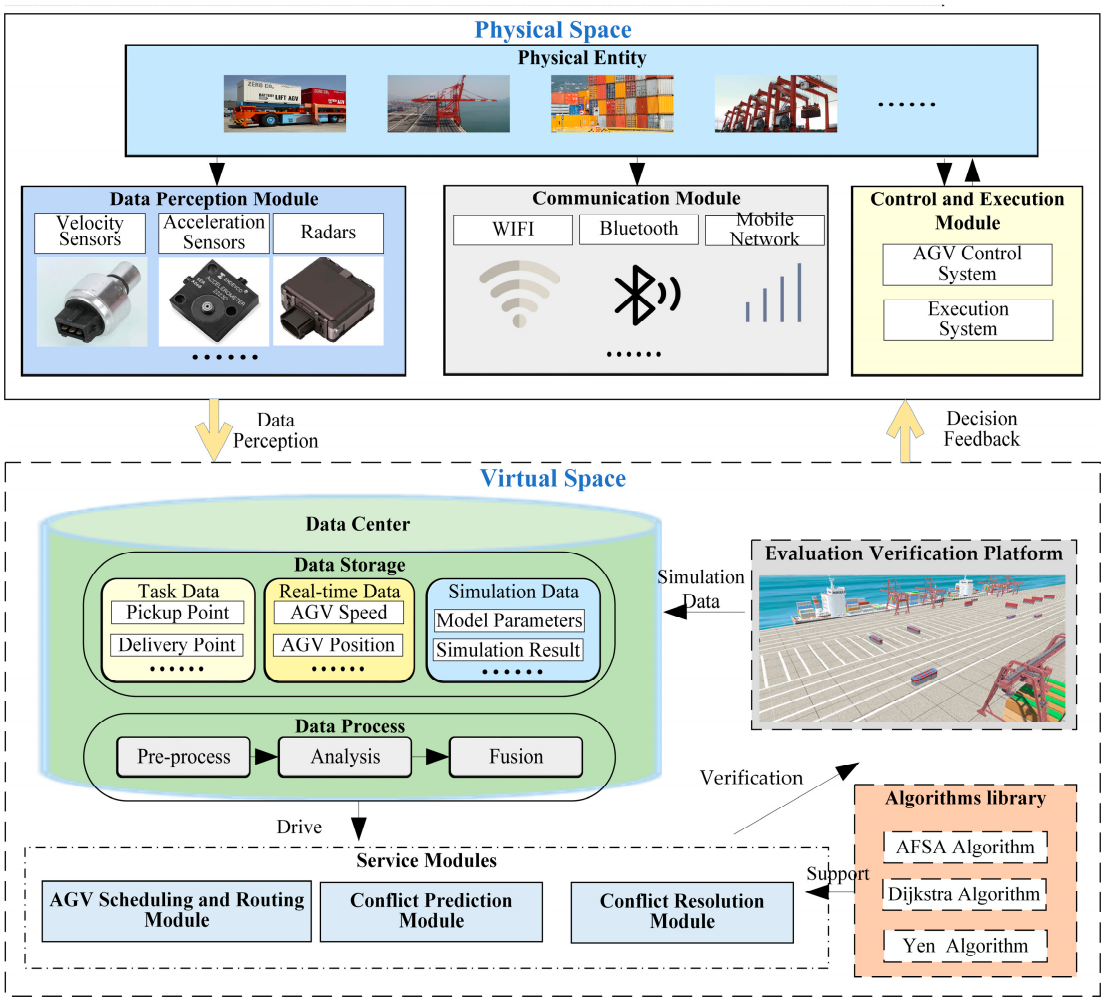


Figure 3. The framework of the digital-twin-driven AGV scheduling and routing.

In the physical space, the physical entities mainly involve AGVs, QCs, and YCs. The data sensing the conditions of various devices and its surrounding are collected and also transmitted to the virtual space. In addition, physical devices will execute in real time according to the decision feedback from the virtual space.

The virtual space consists of four parts, involving the service modules, algorithms library, evaluation verification platform, and data center. The functions of the data center are mainly to store data from the physical and virtual space and to process this data to provide data support for service modules. With the assistance of the algorithm library, the function of the service modules primarily consists of (1) generating an initial AGV scheduling and routing plan to guide the AGV transport process; (2) monitoring the ACT operation process and predicting AGV conflicts by comparing simulation data with real-time data; and (3) resolving conflicts caused by uncertain events, revising the initial plan, and feeding it back into the physical space. The purpose of the evaluation verification platform is to verify the feasibility and efficiency of the plan by simulation.

The workflow of the proposed framework is shown in Figure 4. The specific processes are as follows.

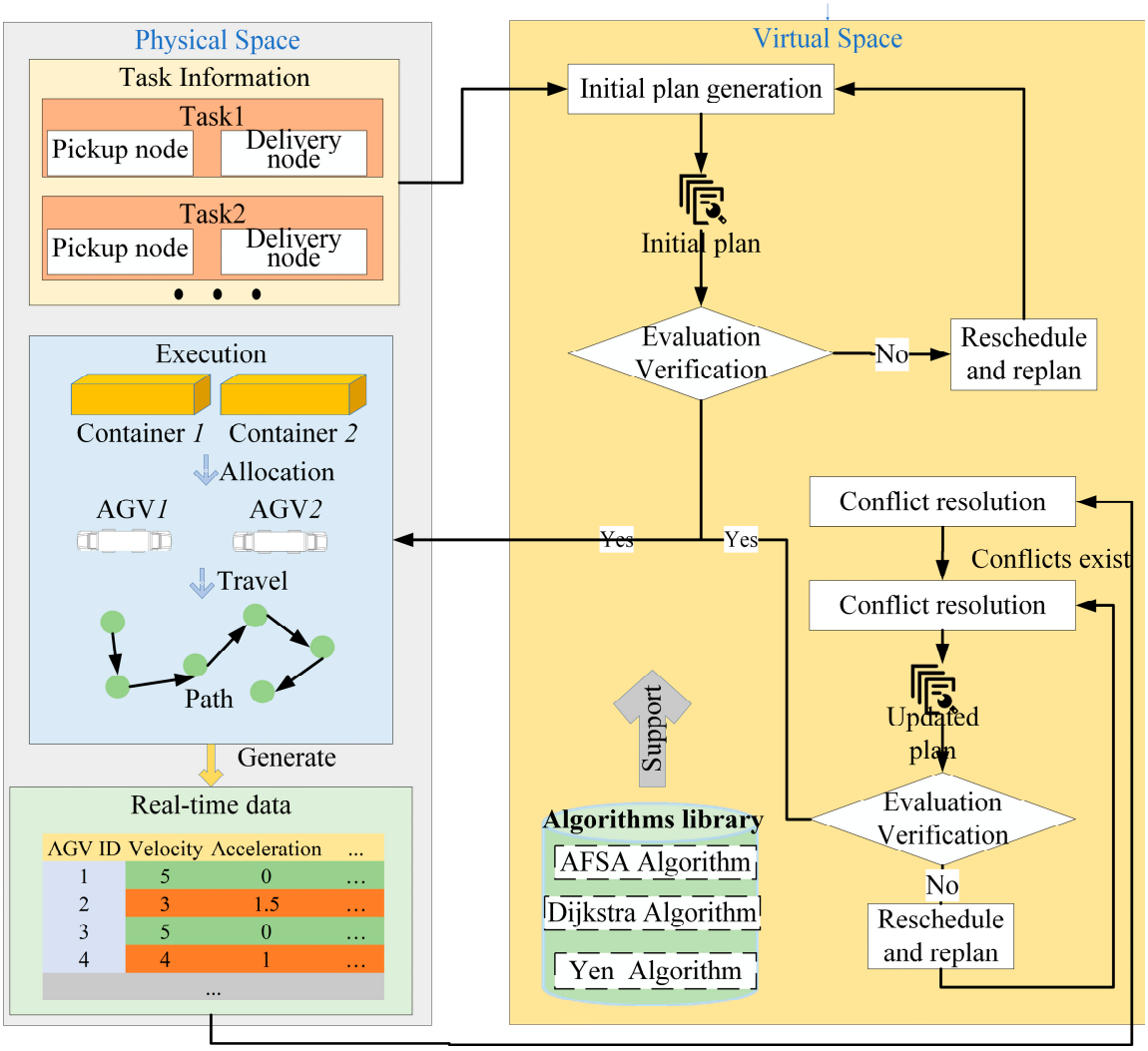


Figure 4. The workflow of digital-twin-driven AGV scheduling and routing.

Step 1: By introducing the digital twin, the AGV transport process is continuously monitored, including AGV speed, AGV acceleration, and AGV position. The real-time data are transmitted to the virtual space.

Step 2: IAFSA-Dijkstra is proposed for the optimal AGV scheduling and routing solution, uncertain factors can be detected and handled through the interaction and fusion of physical and virtual spaces, and a conflict resolution method based on the Yen algorithm is explored to resolve predicted conflicts and drive the evolution of the scheme.

Step 3: The evaluation verification platform evaluates the updated plan. The results of scheduling and path planning are fed into physical space to guide AGV for path planning, which timely responds to changes in the environment.

4.2. IAFSA-Dijkstra Algorithm

A bi-level mixed integer programming model is established in Section 3. The upper level of the model pertains to AGV scheduling, while the lower level of the model concerns collision-free AGV routing. To solve the model, an improved artificial fish swarm algorithm-Dijkstra algorithm (IAFSA-Dijkstra), is presented in this section. The IAFSA is used to optimize AGV scheduling, which determines the start and endpoints of the path. The Dijkstra algorithm is used for collision-free routing, which affects the AGV scheduling process. The flowchart of IAFSA-Dijkstra is shown in Figure 5.

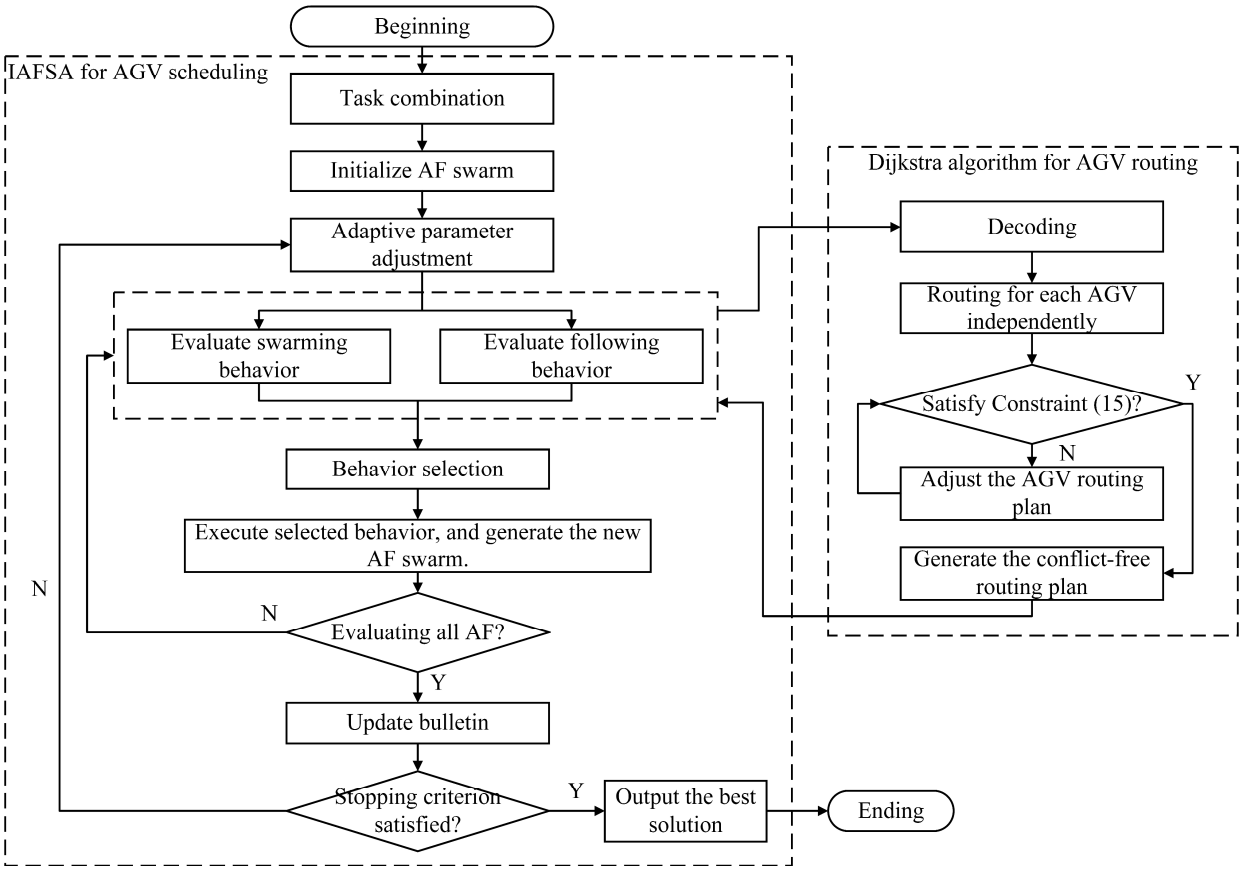


Figure 5. The flowchart of IAFSA-Dijkstra.

AFSA has better global optimization capability owing to its parallel search and the nature of controlling the search direction [50]. There is a more substantial search capability in the early stage of optimization, but the search capability weakens, and it is easily trapped in the local extremum in the later stage. Thus, an adaptive parameter adjustment method is used to improve AFSA. Meanwhile, the encoding method in this work is related to the number of container transport tasks. Therefore, a task combination approach is adopted to combine container tasks according to specific rules to reduce the encoding length.

Encoding and decoding method: A random number encoding method is adopted. The value above each position denotes the container no. The value after rounding in each position denotes the AGV no., and the value range is [0.5, N_{agv} + 0.5), where N_{agv} denotes the number of AGVs. The initial value in each position is randomly generated.

Assume 8 containers and 3 AGVs, and the encoding method is as shown in Figure 6a. Taking the first position as an example, 1.181 is rounded to 1. Therefore, container 1 is assigned to AGV 1. In the same way as for the other positions, AGV 1 is responsible for transporting containers 1, 7, and 8, AGV 2 is responsible for transporting containers 4 and 6, and AGV 3 is responsible for transporting containers 2, 3, and 5. For containers of each AGV, the order after sorting by their value is the transportation order of AGV. Taking AGV 1 as an example, the values of containers 1, 7, and 8 are 1.181, 1.211, 1.203, respectively. Since 1.181 < 1.203 < 1.211, the transportation order of AGV 1 is container 1–container 8–container 7.

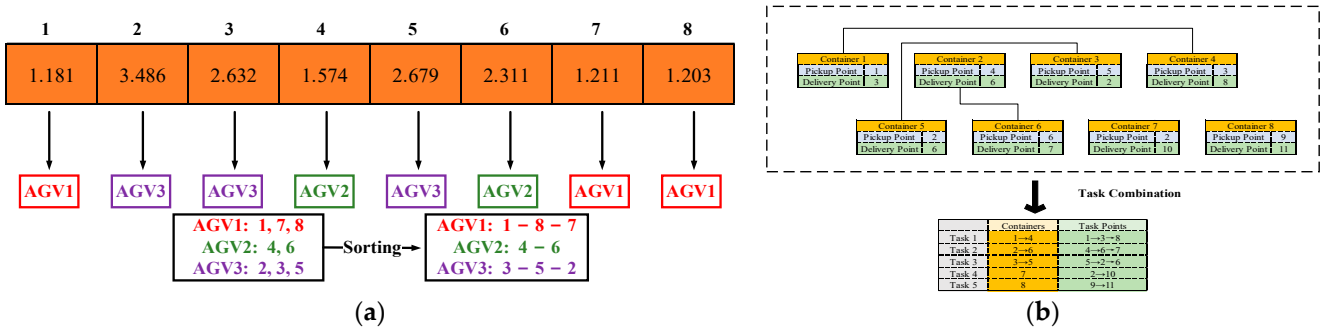


Figure 6. The decoding method and task combination strategy. (**a**) The decoding method; (**b**) Task Combination strategy.

Task combination strategy: Considering the above encoding method, encoding length is related to the number of container transportation tasks. The higher the number of container transportation tasks, the longer the encoding length, which causes a more extended algorithm running time. Therefore, a task combination strategy is introduced to reduce the number of container transportation tasks, shortening the algorithm's running time. The cost of the method is that the number of solution spaces will be reduced.

Task combination is a strategy of combining multiple tasks into one task. If the pickup point of container p and the delivery point of container p' are the same point, two tasks will be combined into one task. The combined task point is the pickup of container p', the delivery point of container p' (viz., the pickup point of container p), and the delivery point of container p in sequence. The task combination with 8 containers as an example is shown in Figure 6b.

Adaptive parameter adjustment strategy: In AFSA, *Visual* and *Step* are essential parameters that affect the movement of the AF. In the early stage of optimization, a larger *Visual* and *Step* can make the AF move quickly and jump out of the local extremum, thus converging faster. Nevertheless, in the later stage of optimization, a larger *Visual* and *Step* will lead to the optimal solution being skipped, which is not conducive to optimization.

In traditional AFSA, *Visual* and *Step* are fixed, which makes it difficult to reconcile the requirements in the early and later stages of optimization. An adaptive parameter adjustment method is used to address the above problem. The value of *Visual* and *Step* will gradually become smaller as the number of iterations increases.

Visual and *Step* can be calculated according to Expressions (18) and (19).

$$Visual(gen) = Visual_{max} \times gen^{log(Visual_{min}/Visual_{max})/log(gen_{max})} \tag{18}$$

$$Step(gen) = Step_{max} \times gen^{log(Step_{min}/Step_{max})/log(gen_{max})} \tag{19}$$

where gen denotes the number of iterations, and $Visual_{max}$, $Visual_{min}$, $Step_{max}$, and $Step_{min}$ denote the maximum and minimum of *Visual* and *Step*, respectively. gen_{max} denotes the maximum number of iterations. $Visual(gen)$ and $Step(gen)$ denote *Visual* and *Step* with the number of iterations.

The procedure of IAFSA is shown in Algorithm 1. First, combine tasks according to task combination strategy and initialize the artificial fish (AF) population. Then, calculate *Visual* and *Step* based on Equations (18) and (19). Meanwhile, evaluate swarming behavior and following behavior and select the optimal behavior to execute. The fitness values of these two behaviors can be obtained based on Algorithm 2. Repeat the above steps until the stopping criterion condition is satisfied.

The Dijkstra algorithm is a path search algorithm to determine the optimal path in one given path network. The traditional Dijkstra algorithm cannot prevent potential AGV conflict; thus, a criterion called "Node visit time (NT)" is presented to record the time at which AGVs visit all nodes, expressed by Equation (20). NT makes it possible to take into account the path information of other AGVs when planning paths for one AGV, effectively avoiding potential AGV conflicts.

$$NT = \{nt_1, \ldots, nt_i, \ldots,\}, nt_i = \{s_i, k_i, t_i\} \tag{20}$$

where nt_i denotes AGV k_i visits to node s_i at moment t_i.

The procedure of the Dijkstra algorithm is shown in Algorithm 2. First, the task assignment result is obtained based on the decoding result. Based on the task assignment results, the start and end nodes of the AGV can be determined so that the optimal route of the AGV can be obtained by Dijkstra algorithm. The plan of AGV k can be denoted by S_k^v, expressed by Expression (21).

$$S_k^v = \left\{ \left(n_{k,1}^v, t_{k,1}^v\right), \left(n_{k,2}^v, t_{k,2}^v\right), \left(n_{k,3}^v, t_{k,3}^v, \tau_{k,L,1}^v\right), \ldots, \left(n_{k,6}^v, t_{k,6}^v, \tau_{k,U,1}^v\right), \ldots, \right\} \tag{21}$$

where the explanations of $n_{k,1}^v$, $t_{k,1}^v$, and $\tau_{k,U,1}^v$ are in Appendix B.

Algorithm 1: IAFSA in AGV scheduling.

Input: Problem data, IAFSA parameters.
Output: The optimal solution $plan_{best}$, the completion time of all tasks Y_{best}.
$Y_{best} \leftarrow \infty, plan_{best} \leftarrow \varnothing$
Task combination.
Initialize AF population $[X_1, \ldots, X_{N_{AF}}]$
while $gen \leftarrow 1$ ***to*** gen_{max} ***do***
  calculate *Visual* and *Step* by Expressions (18) and (19)
  for $i \leftarrow 1$ ***to*** N_{AF} ***do***
    $[X_{i1}, Y_{i1}, plan_{i1}] \leftarrow swarm(X_i)$ // Evaluate swarming behavior,
    $[X_{i2}, Y_{i2}, plan_{i2}] \leftarrow follow(X_i)$ // Evaluate following behavior
    $[X_i, Y_i, plan_i] \leftarrow (Y_{i1} < Y_{i2})?[X_{i1}, Y_{i1}, plan_{i1}] : [X_{i2}, Y_{i2}, plan_{i2}]$
  end for
  $Y_{min} \leftarrow \min\{Y_1, Y_2, \ldots, Y_{N_{AF}}\}$
  $[Y_{best}, plan_{best}] \leftarrow (Y_{min} < Y_{best})?[Y_{min}, plan_{min}] : [Y_{best}, plan_{best}]$
end while

Then, NT is updated based on the routing results and sorted in chronological order. If two AGVs visit the same path nodes and their visit moments do not satisfy Equation (15), their infornamtion are recorded into C^P. The set C^P records the events that do not satisfy Equation (15), expressed by Equation (22).

$$C^P = \{e_1, e_2, \ldots, e_i, \ldots\}, e_i = \{s_i, k_{i,1}, t_{i,1}, k_{i,2}, t_{i,2}\} \tag{22}$$

where e_i denotes AGV $k_{i,1}$ visits to node s_i at moment $t_{i,1}$, AGV $k_{i,2}$ visits to node s_i at moment $t_{i,2}$, and $|t_{i,1} - t_{i,2}| < D_s/v_0$.

If C^P is empty, the time relationship of all AGVs visiting the path node satisfies Equation (15), and there is no potential conflict. Otherwise, if e_1 involves two AGVs, the passage priority is decided according to Equation (23), and the low-priority AGV waits for $(\Delta t + D_s/v_0)$, where Δt denotes the time gap between the high-priority AGV and the low-priority AGV to visit the path node. If e_1 involves multiple AGVs, Equation (23) may lead to a circular waiting problem; thus, Equation (24) is used to decide the passage order, and the low-priority AGV waits for $(\Delta t + D_s/v_0)$.

$$C_{k1} > C_{k2} \rightarrow F_{k1} > F_{k2} \tag{23}$$

$$t_{k1} > t_{k2} \rightarrow F_{k1} > F_{k2} \tag{24}$$

where t_{k1} denotes the moment when AGV $k1$ visits the conflicting node and F_{k1} denotes the passage priority of AGV $k1$.

Algorithm 2: Dijkstra algorithm in AGV routing.

Input: Coding results of IAFSA, problem data
Output: The fitness value of IAFSA Y.
Decode to obtain task assignment results.
for $k \in V$ ***do***
  planning path for AGV k using Dijkstra algorithm, recorded in S_k^v
end for
while True ***do***
  $C^P \leftarrow \varnothing$, $NT \leftarrow \varnothing (s \in N)$
  update NT based on $\{\ldots, S_k^v, \ldots\}$, and sort NT in chronological order
  for $nt_i, nt_{i+1} \in NT$ ***do***
    if $s_i == s_{i+1}$ ***and*** $|t_{i+1} - t_i| < D_s / v_0$ ***and*** $s_i \in N_P$ ***then***
      record $\{s_i, k_i, t_i, k_{i+1}, t_{i+1}\}$ into C^P
    end if
  end for
  if $isEmpty(C^P)$ ***then***
    break;
  end if
  sort C^P in chronological order, $t_m \leftarrow t_{1,2}$
  for $e_i \in C^P (i > 1)$ ***do*** // update e_1
    if $s_i == s_1$ ***and*** $\left|t_{i,1} - t_m\right| < D_s / v_0$ ***then***
      $e_1 = e_1 \cup e_i$, $t_m \leftarrow t_{i,2}$
    end if
  end for
  if e_1 involves two AGVs ***then***
    decide the pass priority according to Equation (23) and update S_k^v
  else
    decide the pass priority according to Equation (24) and update S_k^v
  end if
end while
$Y \leftarrow max(C_1, \ldots, C_k, \ldots) (k \in V)$

4.3. Twin-Data-Driven AGV Conflict Prediction Method

During the AGV transport process, the occurrence of disturbance events (e.g., AGV failure, AGV speed fluctuations, and uncertain QC operation times) will influence the execution of the scheduling and routing plan, resulting in the deviation between the actual plan and the expected plan. As the AGV operation is a continuous process, the accumulation of deviations may affect the subsequent AGV operation and cause AGV conflicts. Therefore, it is necessary to predict the subsequent potential AGV conflicts to facilitate early countermeasures to deal with them. A twin-data-driven AGV conflict prediction method is explored to monitor the deviation between the actual and original plans and then predict AGV conflicts. The parameters in this section are shown in Appendix B. The workflow of the twin-data-driven AGV conflict prediction method is shown in Figure 7.

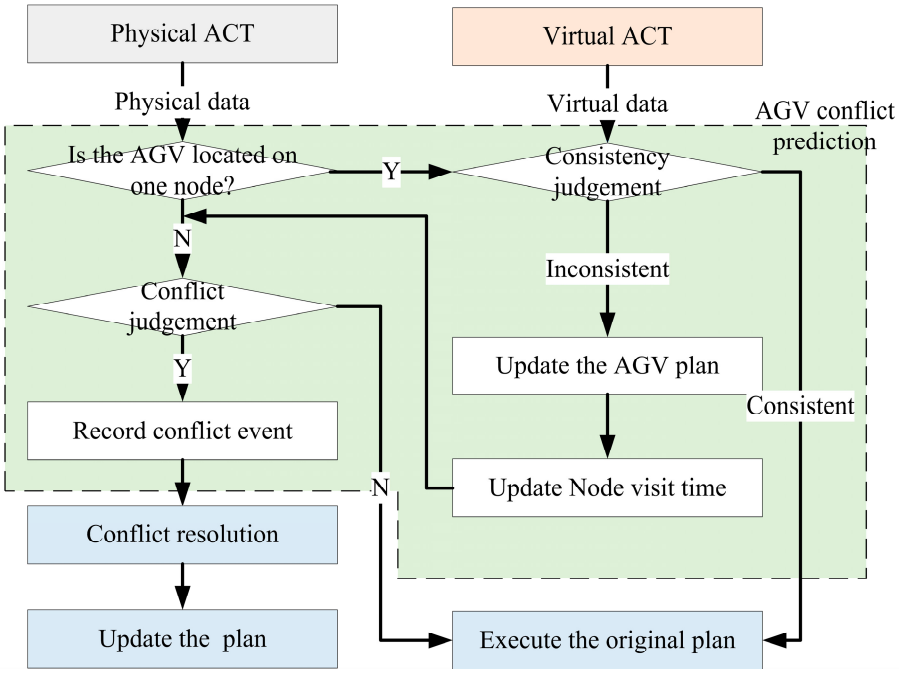


Figure 7. The workflow of twin-data-driven AGV conflict prediction method.

During the AGV transport process, each AGV reports its operation state information to the virtual space in real-time, expressed by Expression (25).

$$info = \{k, t, v_{k,t}, a_{k,t}, pos_{k,t}, phase_{k,t}\} \tag{25}$$

If AGV k is located on the path at moment t (i.e., $pos_{k,t} \notin N$), the time to reach its next node is predicted according to Expression (26). Expression (27) is used for conflict prediction. When AGV k and AGV q satisfy (27), the two AGVs will conflict in their next node. The conflict event $\left\{s, k, t^N_{k,t}, pos_{k,t}, q, t^N_{q,t}, pos_{q,t}\right\}$ is recorded to the conflict set C^N_s. Assume the two conflicts in C^N_s, denoted as $\left\{s, k1, t^N_{k1,t}, pos_{k1,t}, k2, t^N_{k2,t}, pos_{k2,t}\right\}$, $\left\{s, k3, t^N_{k3,t}, pos_{k3,t}, k4, t^N_{k4,t}, pos_{k4,t}\right\}$; if $\left|\min\left(t^N_{k3,t}, t^N_{k4,t}\right) - \max\left(t^N_{k1,t}, t^N_{k2,t}\right)\right| < D_s/v_0$, these two events are combined into one event $\{s, k1, t^N_{k1,t}, pos_{k1,t}, k2, t^N_{k2,t}, pos_{k2,t}, k3, t^N_{k3,t}, pos_{k3,t}, k4, t^N_{k4,t}, pos_{k4,t}\}$. C^N records the conflict events for all AGVs' next path nodes, $C^N = C^N_1 \cup C^N_2 \cup \ldots \cup C^N_s \cup \ldots, s \in N_P$.

$$t^N_{k,t} = \begin{cases} D^N_{k,t}/v_{k,t}, \ if \ a_{k,t} = 0 \\ \infty, \ if \ a_{k,t} < 0 \ and \ \frac{{v_{k,t}}^2}{2\times|a_{k,t}|} \le D^N_{k,t}, \ \forall k \in V \\ \frac{\sqrt{{v_{k,t}}^2 + 2\times a_{k,t}\times D^N_{k,t}} - v_{k,t}}{a_{k,t}}, \ o.w. \end{cases} \tag{26}$$

$$\begin{cases} \left|t^N_{k,t} - t^N_{q,t}\right| < D_s/v_0 \\ s = n_{k,t} = n_{q,t} \\ s \in N_P \end{cases}, \forall k, q \in V \tag{27}$$

If AGV k is located on the node at moment t (i.e., $pos_{k,t} \in N$), the deviations between the plan and the actual operation will be measured to predict the occurrence of conflicts. Aiming to measure the deviation, a criterion called "Consistency deviation (ΔTD)" is

presented, defined as the deflection between the planned AGV operation state and the actual AGV operation state, expressed by Expression (28).

$$\Delta TD = \{\Delta TD_N, \Delta TD_L, \Delta TD_U\} \tag{28}$$

Due to the fluctuating AGV speed during the AGV transport process, the actual time of the AGV visiting one node will deviate from the planned time. The consistency deviation of node visit time ΔTD_N is calculated by Expression (29).

$$\Delta TD_N = n^r_{k,i} - n^v_{k,i} \tag{29}$$

Due to the uncertain operation time of QCs and YCs, one container's actual loading and unloading time will deviate from the planned time. The consistency deviation of loading ΔTD_L and unloading time ΔTD_U is calculated by Expressions (30) and (31).

$$\Delta TD_L = \tau^r_{k,L,j} - \tau^v_{k,L,j} \tag{30}$$

$$\Delta TD_U = \tau^r_{k,U,j} - \tau^v_{k,U,j} \tag{31}$$

When $\Delta TD \neq 0$, the current AGV's times of visiting subsequent nodes need to superimpose ΔTD. Assume that AGV k has a consistency deviation ΔTD_N from the expected plan when visiting the ith node. The updated plan of AGV k can be denoted as $\left\{\ldots, \left(n^r_{k,i}, t^r_{k,i}\right), \left(n^v_{k,i+1}, t^v_{k,i+1} + \Delta TD_N\right), \left(n^v_{k,i+2}, t^v_{k,i+2} + \Delta TD_N\right), \ldots\right\}$.

NT will be updated as the AGV plan is updated and sorted in chronological order. Then, if the time of visiting node s_i by two AGVs in NT satisfies $|t_{i,1} - t_{i,2}| < D_s/v_0$, the conflict event $\{s_i, k_{i,1}, t_{i,1}, k_{i,2}, t_{i,2}\}$ will be recorded to the conflict set C^P. Assume the two conflicts in C^P, denoted as $\{s_i, k_{i,1}, t_{i,1}, k_{i,2}, t_{i,2}\}$, $\{s_i, k_{j,1}, t_{j,1}, k_{j,2}, t_{j,2}\}$; if $\left|t_{j,1} - t_{i,2}\right| < D_s/v_0$, these two events are combined into one event $\{s_i, k_{i,1}, t_{i,1}, k_{i,2}, t_{i,2}, k_{j,1}, t_{j,1}, k_{j,2}, t_{j,2}\}$. C^P records the conflict events of all AGVs' non-next path nodes.

4.4. AGV Conflict Resolution Method Based on Yen's Algorithm

In this section, a conflict resolution method is proposed for resolving AGV conflicts and revising the original plan.

The deviation between the initial plan and the revised plan can affect the stability of the ACT operation. Therefore, the objective of the AGV conflict resolution method is to minimize the time deviation of all AGVs' task completion time between the initial plan and the revised plan, as in (32), while being subject to Equations (12)–(17).

$$\min\left(\sum_{k \in V}\left(C'_k - C_k\right)\right) \tag{32}$$

For those conflicts in C^N, the AGVs are located on the path, and the AGVs will conflict at their next node. Since the AGV transportation area is a directed graph, AGVs can only travel in one direction on a certain path. The conflicts in C^N cannot be resolved by planning a new route—only by the waiting strategy. All conflicting AGVs visit the conflict node sequentially according to the time order, and the time interval between two AGVs visiting the conflicting node is D_s/v_0. Taking one conflict $\left\{s, k1, t^N_{k1,t}, pos_{k1,t}, k2, t^N_{k2,t}, pos_{k2,t}, k3, t^N_{k3,t}, pos_{k3,t}\right\}$ as an example, assume that $t^N_{k1,t} < t^N_{k2,t} < t^N_{k3,t}$; then, AGV $k1$ visits node s at moment $t^N_{k1,t}$, AGV $k2$ visits node s at moment $t^N_{k1,t} + D_s/v_0$ with waiting time $t^N_{k1,t} - t^N_{k2,t} + D_s/v_0$, and AGV $k3$ visits node s at moment $t^N_{k1,t} + 2D_s/v_0$ with waiting time $t^N_{k1,t} - t^N_{k3,t} + D_s/v_0$.

The conflicts in C^P can be resolved by re-routing for AGVs. Yen's algorithm is an algorithm for obtaining multiple shortest paths from a starting point to an endpoint with the idea of a deviated path algorithm in the recursive method. An AGV conflict resolution based on Yen's algorithm is explored to resolve conflicts in C^P. The procedure of Yen's algorithm is shown in Algorithm 3. In Algorithm 3, m paths are planned from the current node to the phase endpoint for AGV k if the conflicting phase is the current phase of AGV k. Otherwise, m paths are planned from the phase start point to the phase endpoint. The waiting strategy is used to resolve the conflict if the conflict still exists in the conflicting phase after re-routing the path. Otherwise, adopt Yen's algorithm to resolve the subsequent conflicts.

Algorithm 3: Yen's algorithm in AGV conflict resolution.

Input: Problem data, real-time information for all AGVs, NT, S_k^v, C^P
Output: the revised plan $plan_{new}$
sort C^P in chronological order, and update e_1
for $k \in e_1$ ***do***
  obtain AGV k's phase at the occurrence of e_1, denoted as $phase_C$
  if $phase_C$ is the current phase of AGV k ***then***
    arrange m paths from the current node to the phase endpoint for AGV k based on Yen's algorithm, and update $\left\{S_{k,1}^v, \ldots, S_{k,m}^v\right\}$
  else
    arrange m paths from the phase start point to the phase endpoint for AGV k based on Yen's algorithm, and update $\left\{S_{k,1}^v, \ldots, S_{k,m}^v\right\}$
  end if
  for $i \leftarrow 1$ ***to*** m ***do***
    while True ***do***
      update NT, and C^P.
      if *is Empty*$\left(C^P\right)$ ***then***
        $f_{k,i} \leftarrow \sum_{k \in V}\left(C_k' - C_k\right)$
        break;
      end if
      if $e_1 \in phase_C$ ***then***
        adopt a waiting strategy to resolve the conflict, and update $S_{k,i}^v$.
      else
        adopt Yen's algorithm to resolve the conflict, and update $S_{k,i}^v$.
      end if
    end while
  end for
end for
select the plan $plan_{new}$ corresponding to $min\ (f_{1,1}, f_{1,2}, \ldots, f_{k,i}, \ldots)$

4.5. Evaluation Verification

Before the initial plan is executed in the physical space, the ACT evaluation verification platform simulates the ACT operation process, thus evaluating the efficiency and feasibility of the plan. The process of AGV evaluation verification is as follows [44].

First, the ACT evaluation verification platform is built based on Unity2021.2.7f1c1, as shown in Figure 8. Then, the process flow and data interface of the ACT are established, and real-time data collected from the physical space is transmitted to the simulation platform through the data interface. Finally, the ACT operation process can be mimicked via the simulation platform, based on the plan, to evaluate the task completion time of each AGV, conflict incidence rate, etc. The evaluation indicators are an essential guide for ACT operation of the physical space. Meanwhile, since these models in the virtual space can be continuously updated according to real-time data, the simulation results can better reflect the actual AGV operation situation.

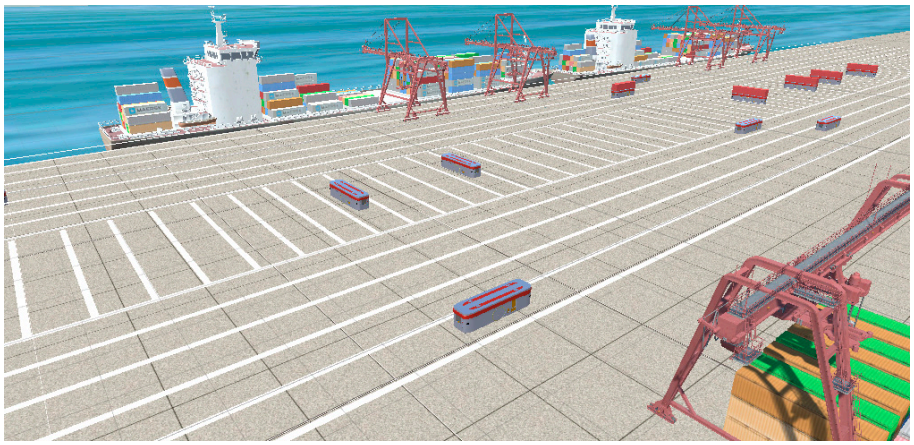


Figure 8. The ACT evaluation verification platform in Unity.

5. Experiments

In the section, some experiments are conducted to verify the benefits of the proposed method. The first experiment of AGV scheduling and routing is aimed to verify the performance of the IAFSA and calculate the optimal scheduling and path planning results, and based on the result of the first experiment, the second experiment of AGV conflict prediction and resolution is conducted to verify the efficiency of DT-based conflict prediction and resolution. All experiments are performed on Windows 10, Intel (R) Core (TM) i7-10750H CPU @ 2.60GHz, 16GB RAM MATLAB 2018a.

As shown in Figure 9, the transportation area is abstracted into the weighted directed graph, including 18 nodes and 44 paths. *n3*, *n5*, and *n7* indicate the three nodes of QCs, *n12*, *n14*, and *n16* represent the three nodes of storage yards, and the other nodes are the path nodes. The number of QCs and YCs is 3. The safe distance D_s = 15 m, and the velocity v_0 = 5 m/s.

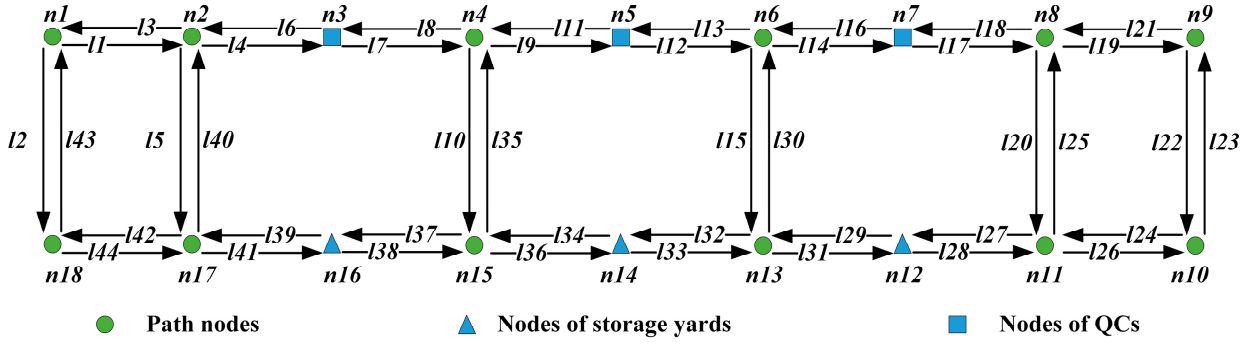


Figure 9. The AGV path in an ACT.

5.1. Verification of AGV Scheduling and Routing

The experiment of AGV scheduling and routing is used to verify the performance of the IAFSA and calculate the optimal scheduling and path planning results.

The experimental data involved in this Section is shown in Table 1, including container no. and the container transportation tasks. The experiment consists of 30 container transportation tasks transported by 6 AGVs. The start nodes of the 6 AGVs are [*n1*, *n5*, *n3*, *n15*, *n7*, *n6*]. The IAFSA parameters are set based on preliminary tests and are set as follows: $Visual_{min}$ = 0.1, $Visual_{max}$ = 1.5, $Step_{min}$ = 0.1, $Step_{max}$ = 1.5, gen_{max} = 300, try_number = 10, δ = 0.618, and N_{AF} = 20.

Table 1. Container transportation tasks.

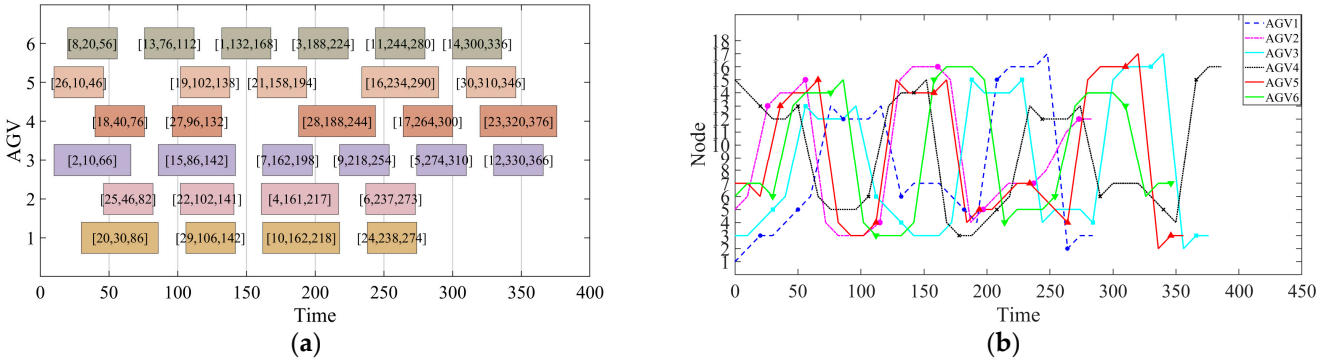
No.	Task	No.	Task	No.	Task	No.	Task	No.	Task
1	[*n3*, *n16*]	7	[*n3*, *n14*]	13	[*n14*, *n3*]	19	[*n3*, *n14*]	25	[*n14*, *n3*]
2	[*n3*, *n12*]	8	[*n7*, *n14*]	14	[*n14*, *n7*]	20	[*n3*, *n12*]	26	[*n7*, *n14*]
3	[*n16*, *n5*]	9	[*n14*, *n5*]	15	[*n12*, *n3*]	21	[*n14*, *n5*]	27	[*n5*, *n14*]
4	[*n16*, *n7*]	10	[*n7*, *n16*]	16	[*n7*, *n16*]	22	[*n3*, *n16*]	28	[*n3*, *n12*]
5	[*n5*, *n16*]	11	[*n5*, *n14*]	17	[*n12*, *n7*]	23	[*n7*, *n16*]	29	[*n12*, *n7*]
6	[*n7*, *n12*]	12	[*n16*, *n3*]	18	[*n12*, *n5*]	24	[*n16*, *n3*]	30	[*n16*, *n3*]

As shown in Figure 6b, the task combination strategy is adopted to combine container tasks according to specific rules to reduce the encoding length. Through the task combination strategy, the 30 container transportation tasks are combined to 16 post-combination tasks, as shown in Table 2, including new task no., task nodes, and post-combination task no.

Table 2. Post-combination tasks.

No.	Nodes	Task	No.	Nodes	Task	No.	Nodes	Task
1	[*n3*, *n16*, *n5*]	1,3	7	[*n7*, *n16*, *n3*]	10,24	13	[*n3*, *n12*, *n7*]	20,29
2	[*n3*, *n12*, *n3*]	2,15	8	[*n5*, *n14*, *n7*]	11,14	14	[*n14*, *n3*, *n16*]	25,22
3	[*n16*, *n7*, *n12*]	4,6	9	[*n7*, *n16*, *n3*]	16,30	15	[*n7*, *n14*]	26
4	[*n5*, *n16*, *n3*]	5,12	10	[*n12*, *n7*, *n16*]	17,23	16	[*n3*, *n12*]	28
5	[*n3*, *n14*, *n5*]	7,9	11	[*n12*, *n5*, *n14*]	18,27			
6	[*n7*, *n14*, *n3*]	8,13	12	[*n3*, *n14*, *n5*]	19,21			

The AGV scheduling Gantt chart is shown according to the scheduling and routing plan in Figure 10a. The numbers in each rectangle indicate the container no., the container loading completion time, and the container unloading start time in sequence. The completion time for all tasks is 386. The path is shown according to the scheduling and routing plan in Figure 10b. The time interval for two AGVs to visit the same node is greater than or equal to 3 s (D_s/v_0 = 15 m/(5 m/s) = 3 s), which satisfies Constraint (15). Thus, the AGV routing is conflict-free.



(**a**) (**b**)

Figure 10. AGV scheduling and routing result of 6 AGVs handling 30 containers (**a**) AGV scheduling result; (**b**) AGV routing result.

To verify the effectiveness of improvement strategies, the running time and completion time of IAFSA and AFSA are compared. We solved each experiment 10 times and took the average value as a result. The results of the experiment are shown in Table 3, the running time denotes the algorithm's execution time when the maximum number of iterations is reached, and completion time denotes the maximum completion time of tasks when the maximum number of iterations is reached. The GAP represents the relative improvement percentage of the IAFSA. In Table 4, N_C denotes the number of containers, N_T denotes the

number of tasks after task combination, and the relative improvement percentage GAP was employed to measure the performance of the IAFSA. The GAP can be calculated by Equation (33):

$$\mathrm{GAP} = \frac{Z_{\mathrm{AFSA}} - Z_{\mathrm{IAFSA}}}{Z_{\mathrm{AFSA}}} \times 100\%, \tag{33}$$

where Z_{AFSA} denotes the running time or the completion time of the AFSA and Z_{IAFSA} denotes the running time or the completion time of the IAFSA.

Table 3. Results comparison between AFSA and IAFSA.

No.	N_C/N_T	AGV	Running Time (s)			Completion Time (s)		
			AFSA	IAFSA	GAP	AFSA	IAFSA	GAP
1	20/12	4	702	293	58.26%	430	398	7.44%
2	30/17	5	1394	565	59.47%	529	467	11.72%
3	30/16	6	1443	573	60.29%	452	408	9.73%
4	50/26	6	2136	684	67.98%	751	656	12.65%
5	100/53	8	6250	1837	70.61%	1174	1072	8.69%
6	100/55	10	8182	2401	70.66%	966	897	7.14%
7	200/103	10	23638	5215	77.94%	1930	1691	12.38%
8	200/103	15	35110	9820	72.03%	1338	1221	8.74%

Table 4. Completion time comparison between different algorithms.

No.	N_C/N_T	AGV	Completion Time (s)			
			GA	PSO	GWO	IAFSA
1	10/6	2	409	386	396	367
2	20/11	3	572	541	546	493
3	30/15	4	643	618	626	550
4	30/15	5	539	502	517	445
5	50/26	6	793	759	769	681
6	80/42	8	975	930	933	871
7	80/42	10	841	790	805	718
8	100/51	10	1035	984	1006	897

For different problem scales, the running time of the AFSA is between 702 and 35110 s, and the running time of the IAFSA is between 293 and 9820 s; the completion time of AFSA is between 430 and 1930 s, and the completion time of IAFSA is between 398 and 1691 s. The running time average GAP between AFSA and IAFSA in the eight experiments is 67.15%. The completion time average GAP between AFSA and IAFSA in the eight experiments is 9.81%.

To verify the effectiveness of IAFSA, IAFSA was compared with other algorithms in terms of completion time, including the genetic algorithm (GA), particle swarm optimization algorithm (PSO), and gray wolf optimization algorithm (GWO). We solved each experiment 10 times and took the average value as a result. The results of the experiment are shown in Table 4. For different problem sizes, the solution quality of IAFSA is better than the other algorithms compared.

5.2. Verification of AGV Conflict Prediction and Resolution

In order to verify the effectiveness of the AGV conflict prediction and resolution method proposed in Section 4, some disturbance events are simulated. The disturbance events considered in this work are uncertain container loading and unloading times. Assume that each container's loading and unloading time obeys a uniform distribution U (7, 13).

As shown in Figure 11, which shows the loading and unloading times of the containers handled by the 6 AGVs, the horizontal coordinates indicate the loading and unloading

process, (1, L) represents the loading node of the first container, (2, U) represents the unloading node of the second container, and so on. The vertical coordinates indicate the loading/unloading time.

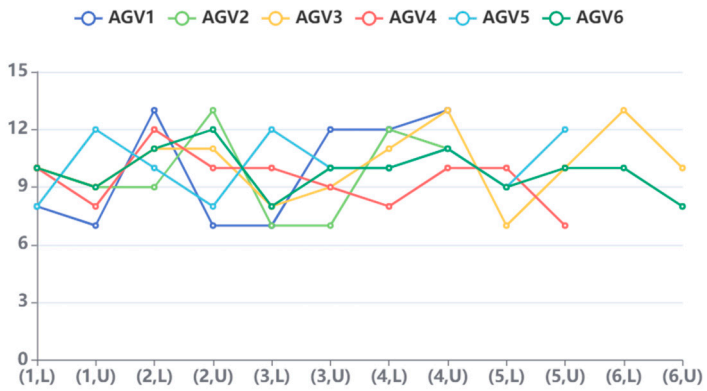


Figure 11. The loading and unloading times of the containers handled by the 6 AGVs.

The uncertain container loading/unloading times lead to AGV conflicts. The details of some predicted AGV conflict events are shown in Table 5. At the 100th second, the loading of AGV2′s second container is completed, and the virtual space predicts that there will be AGV conflicts. The first predicted AGV conflict is Conflict 1 in Table 5. For Conflict 1, AGV2′s route to transport its second container is re-routed to avoid the conflict. The route of AGV2 to transport its second container before re-routing is $\ldots \rightarrow n3 \rightarrow n4 \rightarrow n15 \rightarrow n16 \rightarrow \ldots$, and the route of AGV2 after re-routing is $\ldots \rightarrow n3 \rightarrow n2 \rightarrow n17 \rightarrow n16 \rightarrow \ldots$. A new conflict is caused after re-routing, as in Conflict 2 in Table 5. For Conflict 2, AGV5′s route to transport its third container is re-routed to avoid the conflict. The route of AGV5 to transport its third container before re-routing is $\ldots \rightarrow n14 \rightarrow n15 \rightarrow n4 \rightarrow n5 \rightarrow \ldots$, and the route of AGV5 after re-routing is $\ldots \rightarrow n14 \rightarrow n13 \rightarrow n6 \rightarrow n5 \rightarrow \ldots$. At the 154th second, the loading of AGV1′s third container is completed, and the virtual space predicts that there will be AGV conflicts. The first predicted AGV conflict is Conflict 3 in Table 5. For Conflict 3, both AGV1 and AGV2 keep the original path. AGV1 waits for 1 s after AGV2 visits $n4$ before visiting $n4$. At the 261st second, the loading of AGV4′s fourth container is completed, and the virtual space predicts that there will be AGV conflicts. The first predicted AGV conflict is Conflict 4 in Table 5. For Conflict 4, both AGV4 and AGV6 keep the original path. AGV4 waits for 2 s after AGV6 visits $n13$ before visiting $n13$.

Table 5. The description of AGV conflicts.

Conflict	Time	Node	AGV	Description
1	112–113	$n4$	5-2	AGV5 and AGV2 happen conflict in the 112th–113th second in node $n4$.
2	166–168	$n15$	2-5	AGV2 and AGV5 happen conflict in the 166th–168th second in node $n15$.
3	182–184	$n4$	2-1	AGV2 and AGV1 happen conflict in the 182th–184th second in node $n4$.
4	270–271	$n13$	6-4	AGV6 and AGV4 happen conflict in the 270th–271st second in node $n13$.

In order to handle the AGV conflicts described above, introduce an AGV conflict resolution method based on Yen's algorithm. The procedure of the conflict resolution method is shown in Algorithm 3, and it can solve the path of AGV conflicts well. As shown in Figure 12, the revised path of 6 AGVs handing 30 containers is the result of conflict resolution. After the plan revision, the task completion times for each AGV were 284 s, 278 s, 378 s, 382 s, 358 s, and 344 s.

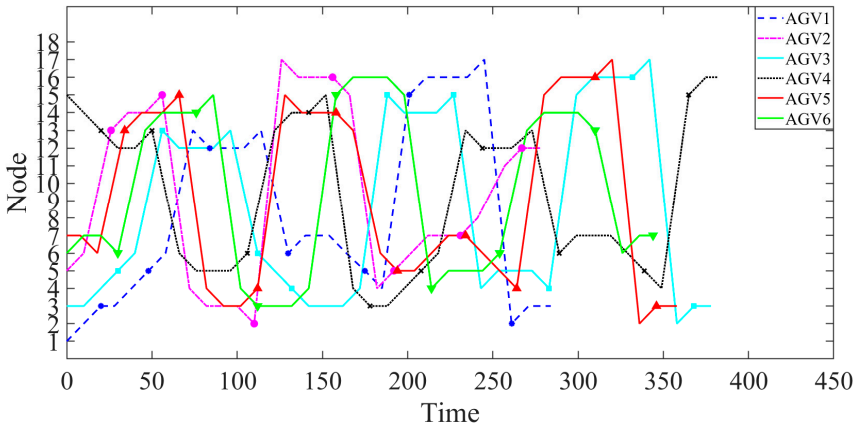


Figure 12. Path of 6 AGVs handling 30 containers after conflict resolution.

Uncertain container loading or unloading time is simulated to verify the efficiency of DT-based conflict prediction and resolution in different problem scales. The results are shown in Figure 13, the traditional method to conflict is the waiting strategy. The DT-based method uses DT technology to predict and resolve AGV conflicts when the plan execution deviates. The vertical axis indicates the time deviation between the initial and revised plans. The difference in the effect of the two methods on the time deviation is not apparent when the problem scales are small. As the number of AGVs and containers increases, the DT-based method has a smaller time deviation than the traditional method.

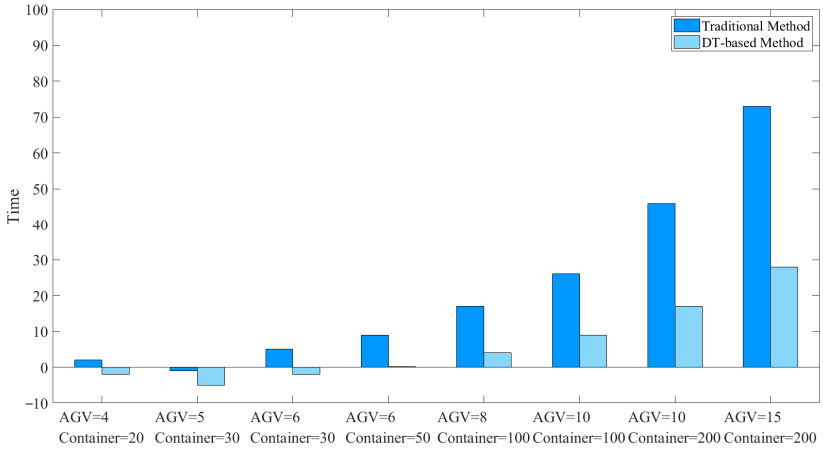


Figure 13. Comparison between DT-based method and traditional method.

6. Conclusions

In this paper, digital-twin-driven AGV scheduling and conflict-free routing is proposed to solve the uncertainties in the physical environment of ACT. The bi-level mixed integer programming model of the AGV scheduling and routing problem is established. The improved artificial fish swarm algorithm Dijkstra (IAFSA-Dijkstra) is proposed to find the optimal AGV scheduling and routing plan. A twin-data-driven conflict prediction method is used for AGV conflict prediction caused by disturbance events, and a conflict resolution method is presented for AGV conflict resolution. The experimental results in different problem scales show that the average solution efficiency and average solution quality of IASFA are better than those of AFSA by 9.81% and 67.15%, respectively. Meanwhile, IAFSA has better solution quality compared with GA, PSO, and GWO. However, the port environment is characterized by continuous, complex, and interlocking components, and the destruction of any link may lead to the deterioration of the entire operation process. Therefore, our future work is to consider the integrated problem of AGVs and other equipment in the port environment to be more in line with the actual situation of the port.

Author Contributions: Conceptualization, P.L. and Y.Z.; methodology, P.L., Y.Z. and X.C. validation, P.L., Y.Z. and X.C.; formal analysis, C.F. and X.C.; investigation, P.L. and Y.Z.; writing—original draft preparation, P.L. and Y.Z.; writing—review and editing, X.C. and C.F.; visualization, Y.Z.; supervision, P.L. and X.C.; project administration, P.L. and J.H.; funding acquisition, P.L. and J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation Committee (NSFC) of China, grant number 52075404, the National Key Research and Development Project of China, grant number 2020YFB1710804, and the Application Basic Frontier Special Project of Wuhan Science and Technology Bureau, grant number 2020010601012176.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are grateful to the subjects for their contributions to the experiment.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Notations and Description in Problem Formulation

Notations	Description
V	Set of AGVs.
C	Set of containers.
N	Set of nodes. $N = N_Q \cup N_Y \cup N_P$.
N_Q	Set of QCs' nodes.
N_Y	Set of storage yards' nodes.
N_P	Set of path nodes.
G	Set of directed line segment between the nodes.
k	Index of AGVs, $k \in V$.
$p,\ p'$	Index of container transport task, $p,\ p' \in C$.
0	The virtual initial container transport task.
f	The virtual last container transport task.
s, e, i, j	Index of nodes, $s, e, i, j \in N$.
C_{max}	All task completion time.
C_k	The completion time of the last task of AGV k.
L_p	The pickup point of container p.
U_p	The delivery point of container p.
O_k	The initial node of AGV k.
v_0	The AGV velocity.
D_s	The safe distance between AGVs.
d_{ij}	The distance from node i to node j, in which $(i, j) \in G$.
t_p^k	The moment when AGV k starts handling container p.
T_{ij}^k	The travel time for AGV k on the connected path of node i and j.
$t_{in,ij}^k$	The moment when AGV k enters the connected path of node i and j.
$t_{out,ij}^k$	The moment when AGV k leaves the connected path of node i and j.
$T_{O_kL_p}^k$	The travel time for AGV k from the initial position of AGV k to the pickup point of container p.
$T_{L_pU_p}^k$	The travel time for AGV k when transporting container p.
$T_{U_pL_{p'}}^k$	The travel time for AGV k from the delivery point of container p to the pickup point of container p'.
τ_L^p	The time to load container p.
τ_U^p	The time to unload container p.
M	A very large positive number.
α_p^k	=1 if container p is assigned on AGV k for transporting; =0 otherwise.
$\beta_{pp'}^k$	=1 if container p and container p' are transported by AGV k consecutively; =0 otherwise.
μ_{ij}^k	=1 if AGV k passes through path from node i to node j, in which $(i, j) \in G$; =0 otherwise.

Appendix B. Parameters in Conflict Prediction Method

Parameters	Descriptions
$n^v_{k,i}$	The ith node that AGV k plans to visit.
$n^r_{k,i}$	The ith node that AGV k actually visits.
$t^v_{k,i}$	The planned time for AGV k to visit its ith node.
$t^r_{k,i}$	The actual time for AGV k to visit its ith node.
$v_{k,t}$	The speed of AGV k at moment t.
$a_{k,t}$	The acceleration of AGV k at moment t.
$pos_{k,t}$	The position of AGV k at moment t.
$phase_{k,t}$	The phase of AGV k at moment t. The traveling process from the pickup point of one container to its delivery point, from the delivery point of one container to the pickup point of the next container, or from the initial position of AGV k to the pickup point of its first container is recorded as one phase.
$D^N_{k,t}$	The distance of AGV k from its next node at moment t, which can be calculated based on the values of $pos_{k,t}$ and $phase_{k,t}$.
$n_{k,t}$	The next node of AGV k at moment t, which can be obtained based on the values of $pos_{k,t}$ and $phase_{k,t}$.
$\tau^v_{k,L,j}$	The planned loading time of the jth task of AGV k.
$\tau^r_{k,L,j}$	The actual loading time of the jth task of AGV k.
$\tau^v_{k,U,j}$	The planned unloading time of the jth task of AGV k.
$\tau^r_{k,U,j}$	The actual unloading time of the jth task of AGV k.

References

1. 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. [CrossRef]
2. Hu, H.; Yang, X.; Xiao, S.; Wang, F. Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning. *Int. J. Prod. Res.* **2021**, *61*, 65–80. [CrossRef]
3. Luo, J.; Wu, Y. Modelling of dual-cycle strategy for container storage and vehicle scheduling problems at automated container terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2015**, *79*, 49–64. [CrossRef]
4. Xu, B.; Jie, D.; Li, J.; Yang, Y.; Wen, F.; Song, H. Integrated scheduling optimization of U-shaped automated container terminal under loading and unloading mode. *Comput. Ind. Eng.* **2021**, *162*, 107695. [CrossRef]
5. Lu, F.; Feng, W.; Gao, M.; Bi, H.; Wang, S. The Fourth-Party Logistics Routing Problem Using Ant Colony System-Improved Grey Wolf Optimization. *J. Adv. Transp.* **2020**, *2020*, 8831746. [CrossRef]
6. Yan, T.; Lu, F.; Wang, S.; Wang, L.; Bi, H. A hybrid metaheuristic algorithm for the multi-objective location-routing problem in the early post-disaster stage. *J. Ind. Manag. Optim.* **2023**, *19*, 4663–4691. [CrossRef]
7. Lu, F.; Yan, T.; Bi, H.; Feng, M.; Wang, S.; Huang, M. A bilevel whale optimization algorithm for risk management scheduling of information technology projects considering outsourcing. *Knowl.-Based Syst.* **2022**, *235*, 107600. [CrossRef]
8. Cai, B.; Huang, S.; Liu, D.; Dissanayake, G. Rescheduling policies for large-scale task allocation of autonomous straddle carriers under uncertainty at automated container terminals. *Robot. Auton. Syst.* **2014**, *62*, 506–514. [CrossRef]
9. Jian, W.; Zhu, J.; Zeng, Q. An Optimization Model of Integrated AGVs Scheduling and Container Storage Problems for Automated Container Terminal Considering Uncertainty. *Symmetry* **2021**, *13*, 1904. [CrossRef]
10. Yue, L.; Fan, H. Dynamic Scheduling and Path Planning of Automated Guided Vehicles in Automatic Container Terminal. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 2005–2019. [CrossRef]
11. Xu, B.; Wang, L.; Li, J. Propagation of Uncertain Events in Multilevel Handlings at Container Terminals from the Perspective of Hypernetwork. *Discret. Dyn. Nat. Soc.* **2021**, *2021*, 6611181. [CrossRef]
12. Zheng, Y.; Yang, S.; Cheng, H. An application framework of digital twin and its case study. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1141–1153. [CrossRef]
13. Rashidi, H.; Tsang, E.P.K. A complete and an incomplete algorithm for automated guided vehicle scheduling in container terminals. *Comput. Math. Appl.* **2011**, *61*, 630–641. [CrossRef]
14. Ma, X.; Bian, Y.; Gao, F. An Improved Shuffled Frog Leaping Algorithm for Multiload AGV Dispatching in Automated Container Terminals. *Math. Probl. Eng.* **2020**, *2020*, 1260196. [CrossRef]
15. Zhao, Q.; Ji, S.; Guo, D.; Du, X.; Wang, H. Research on Cooperative Scheduling of Automated Quayside Cranes and Automatic Guided Vehicles in Automated Container Terminal. *Math. Probl. Eng.* **2019**, *2019*, 6574582. [CrossRef]

16. Angeloudis, P.; Bell, M.G.H. An uncertainty-aware AGV assignment algorithm for automated container terminals. *Transp. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 354–366. [CrossRef]
17. Xin, J.; Negenborn, R.R.; Lodewijks, G. Rescheduling of interacting machines in automated container terminals. *IFAC Proc. Vol.* **2014**, *47*, 1698–1704. [CrossRef]
18. Sahin, C.; Demirtas, M.; Erol, R.; Baykasoğlu, A.; Kaplanoğlu, V. A multi-agent based approach to dynamic scheduling with flexible processing capabilities. *J. Intell. Manuf.* **2017**, *28*, 1827–1845. [CrossRef]
19. Xu, W.; Guo, S.; Li, X.; Guo, C.; Wu, R.; Peng, Z. A Dynamic Scheduling Method for Logistics Tasks Oriented to Intelligent Manufacturing Workshop. *Math. Probl. Eng.* **2019**, *2019*, 7237459. [CrossRef]
20. Zhang, L.; Yan, Y.; Hu, Y.; Ren, W. A dynamic scheduling method for self-organized AGVs in production logistics systems. *Procedia CIRP* **2021**, *104*, 381–386. [CrossRef]
21. Wang, X.; Luo, X.; Han, B.; Chen, Y.; Liang, G.; Zheng, K. Collision-Free Path Planning Method for Robots Based on an Improved Rapidly-Exploring Random Tree Algorithm. *Appl. Sci.* **2020**, *10*, 1381. [CrossRef]
22. Lee, H.; Jeong, J. Mobile Robot Path Optimization Technique Based on Reinforcement Learning Algorithm in Warehouse Environment. *Appl. Sci.* **2021**, *11*, 1209. [CrossRef]
23. Oboth, C.; Batta, R.; Karwan, M. Dynamic conflict-free routing of automated guided vehicles. *Int. J. Prod. Res.* **1999**, *37*, 2003–2030. [CrossRef]
24. Yuan, Z.; Yang, Z.; Lv, L.; Shi, Y. A Bi-Level Path Planning Algorithm for Multi-AGV Routing Problem. *Electronics* **2020**, *9*, 1351. [CrossRef]
25. Xu, C.; Xu, Z.; Xia, M. Obstacle Avoidance in a Three-Dimensional Dynamic Environment Based on Fuzzy Dynamic Windows. *Appl. Sci.* **2021**, *11*, 504. [CrossRef]
26. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
27. Bai, Y.; Ding, X.; Hu, D.; Jiang, Y. Research on Dynamic Path Planning of Multi-AGVs Based on Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 8166. [CrossRef]
28. Onoufriou, G.; Bickerton, R.; Pearson, S.; Leontidis, G. Nemesyst: A hybrid parallelism deep learning-based framework applied for internet of things enabled food retailing refrigeration systems. *Comput. Ind.* **2019**, *113*, 103133. [CrossRef]
29. Zhong, M.; Yang, Y.; Zhou, Y.; Postolache, O. Adaptive Autotuning Mathematical Approaches for Integrated Optimization of Automated Container Terminal. *Math. Probl. Eng.* **2019**, *2019*, 7641670. [CrossRef]
30. Fazlollahtabar, H.; Hassanli, S. Hybrid cost and time path planning for multiple autonomous guided vehicles. *Appl. Intell.* **2018**, *48*, 482–498. [CrossRef]
31. Miyamoto, T.; Inoue, K. Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Comput. Ind. Eng.* **2016**, *91*, 1–9. [CrossRef]
32. Desaulniers, G.; Langevin, A.; Riopel, D.; Villeneuve, B. Dispatching and Conflict-Free Routing of Automated Guided Vehicles: An Exact Approach. *Int. J. Flex. Manuf. Syst.* **2003**, *15*, 309–331. [CrossRef]
33. Xing, L.; Liu, Y.; Li, H.; Wu, C.-C.; Lin, W.-C.; Chen, X. A Novel Tabu Search Algorithm for Multi-AGV Routing Problem. *Mathematics* **2020**, *8*, 279. [CrossRef]
34. Liang, C.; Zhang, Y.; Dong, L. A Three Stage Optimal Scheduling Algorithm for AGV Route Planning Considering Collision Avoidance under Speed Control Strategy. *Mathematics* **2022**, *11*, 138. [CrossRef]
35. Liu, Z.; Meyendorf, N.; Mrad, N. The Role of Data Fusion in Predictive Maintenance Using Digital Twin. In Proceedings of the 44th Annual Review of Progress in Quantitative Nondestructive Evaluation, Provo, UT, USA, 16–21 July 2017; p. 020023. [CrossRef]
36. Guo, J.; Zhao, N.; Sun, L.; Zhang, S. Modular based flexible digital twin for factory design. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1189–1200. [CrossRef]
37. DebRoy, T.; Zhang, W.; Turner, J.; Babu, S.S. Building digital twins of 3D printing machines. *Scr. Mater.* **2017**, *135*, 119–124. [CrossRef]
38. Tao, F.; Zhang, M.; Liu, Y.; Nee, A.Y.C. Digital twin driven prognostics and health management for complex equipment. *CIRP Ann.* **2018**, *67*, 169–172. [CrossRef]
39. Bruynseels, K.; Santoni de Sio, F.; van den Hoven, J. Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm. *Front. Genet.* **2018**, *9*, 31. [CrossRef]
40. Kumar, S.A.P.; Madhumathi, R.; Chelliah, P.R.; Tao, L.; Wang, S. A novel digital twin-centric approach for driver intention prediction and traffic congestion avoidance. *J. Reliab. Intell. Environ.* **2018**, *4*, 199–209. [CrossRef]
41. Zhang, M.; Tao, F.; Nee, A.Y.C. Digital Twin Enhanced Dynamic Job-Shop Scheduling. *J. Manuf. Syst.* **2021**, *58*, 146–156. [CrossRef]
42. Wang, Y.; Wu, Z. Model construction of planning and scheduling system based on digital twin. *Int. J. Adv. Manuf. Technol.* **2020**, *109*, 2189–2203. [CrossRef]
43. Negri, E.; Ardakani, H.D.; Cattaneo, L.; Singh, J.; Macchi, M.; Lee, J. A Digital Twin-based scheduling framework including Equipment Health Index and Genetic Algorithms. *IFAC-PapersOnLine* **2019**, *52*, 43–48. [CrossRef]
44. Han, W.; Xu, J.; Sun, Z.; Liu, B.; Zhang, K.; Zhang, Z.; Mei, X. Digital Twin-Based Automated Guided Vehicle Scheduling: A Solution for Its Charging Problems. *Appl. Sci.* **2022**, *12*, 3354. [CrossRef]

45. Wenna, W.; Weili, D.; Changchun, H.; Heng, Z.; Haibing, F.; Yao, Y. A digital twin for 3D path planning of large-span curved-arm gantry robot. *Robot. Comput. Manuf.* **2022**, *76*, 102330. [CrossRef]
46. Zohdi, T.I. A digital twin framework for machine learning optimization of aerial fire fighting and pilot safety. *Comput. Methods Appl. Mech. Eng.* **2021**, *373*, 113446. [CrossRef]
47. Guo, X.; Peng, G.; Meng, Y. A modified Q-learning algorithm for robot path planning in a digital twin assembly system. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 3951–3961. [CrossRef]
48. Vasanthan, C.; Nguyen, D.T. Combining Supervised Learning and Digital Twin for Autonomous Path-planning. *IFAC-PapersOnLine* **2021**, *54*, 7–15. [CrossRef]
49. Gao, Y.; Chang, D.; Chen, C.-H.; Xu, Z. Design of digital twin applications in automated storage yard scheduling. *Adv. Eng. Inform.* **2022**, *51*, 101477. [CrossRef]
50. Zhang, C.; Zhang, F.; Li, F.; Wu, H. Improved Artificial Fish Swarm Algorithm. In Proceedings of the 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, China, 9–11 June 2014; pp. 748–753.