Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 3785039, 20 pages
https://doi.org/10.1155/2022/3785039

Research Article

Nondominated Sorting Differential Evolution Algorithm to Solve the Biobjective Multi-AGV Routing Problem in Hazardous Chemicals Warehouse

Lini Duan,[1] **Yuanyuan Liu**,[2] **Haiyan Li**,[2] **Kyung-Hye Park**,[3] **and Kerang Cao**[4]

[1]*School of Economics and Management, Shenyang University of Chemical Technology, Shenyang 110142, China*
[2]*School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China*
[3]*College of Economics and Management, Chungnam National University, Daejeon 34134, Republic of Korea*
[4]*College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110142, China*

Correspondence should be addressed to Kerang Cao; caokerang@syuct.edu.cn

Received 27 May 2022; Accepted 24 August 2022; Published 16 September 2022

Academic Editor: Hao Gao

Copyright © 2022 Lini Duan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

For the multiple automated guided vehicle (multi-AGV) routing problems in the warehousing link of logistics, where the optimization objective is to minimize both the number of AGVs used and the maximum pickup time simultaneously, a nondominant sorting differential evolution (NSDE) algorithm is proposed. In the encoding and decoding stages, the pickup point area is divided. AGVs are allocated to each region according to the proposed rule based on avoiding duplicate paths. Meanwhile, the pickup points within the region can be adjusted to optimize the pickup paths and improve the pickup efficiency. The fast nondominated sorting method and elitist selection strategy in the nondominated sorting genetic algorithm II (NSGA-II) are introduced into the differential evolution algorithm, which sorts all the regions to obtain the best Pareto solution set. Lastly, the domination of the proposed NSDE algorithm in Pareto frontier evaluation indicators is verified by some numerical experiments.

1. Introduction

Dangerous chemical accidents have occurred frequently in many countries. Dangerous chemicals have the properties of corrosion, toxicity, and harmfulness, and are easy to explode and combust, so to do a good job in the safety management of the storage, loading and unloading of dangerous chemicals is an important measure to avoid safety accidents. To achieve safety and efficiency in material handling, automated guided vehicle (AGV) system is widely applied in warehouses, manufacturing systems, container terminals, and service industries including hospitals [1]. As a flexible and intelligent warehousing logistics equipment integrating a variety of advanced technologies, AGV has the characteristics of automation, high flexibility, high efficiency, high reliability, and parallel operation [2], which can well meet this logistics demand of dangerous chemicals. Therefore, more and more automatic warehouses of dangerous chemicals use AGV for transportation. This type of transportation problem can be classified as multi-AGV routing problem in an automated warehouse: multiple AGVs start from their respective entrances, drive along passageways pickup the goods at the pickup point mentioned in the order, and finally carry the goods to the exit. AGV routing involves the operation sequence and driving path arrangement of AGV [3]. At present, metaheuristic algorithm is still one of the major ways to solve this kind of complex combinatorial optimization problem, and many scholars have carried out relevant research under various warehouse transportation backgrounds.

AGV handling system has been an essential component of modern manufacturing systems. Kabir and Suzuki [4] conducted a comparative analysis of four AGV routing heuristics for battery management: select the nearest battery station, select minimum delay battery station, select the nearest battery station from the initial point, and select the

nearest battery station to drop-off point. In terms of system productivity, selecting minimum delay battery station performs the best and selecting the nearest battery station from the initial point performs the worst. Considering energy consumption or its environmental impact indicators, Zhang et al. [5] formulated an energy-efficient path planning model for a single-load AGV and selected two optimization objectives, transport distance, and energy consumption. They put forward a particle swarm optimization-based method where an indirect priority-based particle encoding scheme is proposed. For the multi-compartment AGV scheduling problem in matrix workshop, Zou et al. [6] addressed an effective iterated greedy algorithm with some advanced techniques including accelerations for evaluating neighborhood solutions, two improved constructive heuristics based on nearest-neighbour and sweep, an improved destruction procedure, and simulated annealing type of acceptance criterion. Zou et al. [7] then investigated a new AGV scheduling problem with pickup and delivery that simultaneously optimized two objectives of maximizing the overall customers' satisfaction and minimizing the total distribution costs. To solve the problem, an effective multi-objective evolutionary algorithm is proposed, containing a constructive heuristic in population initialization, a multi-objective local search based on an ideal point to improve the capability of exploitation, a two-point crossover operation to exploit helpful information collected in the nondominated solutions, and a restart strategy to prevent trapping into a local optimum. For the problem of collision and deadlock, Zając and Małopolski [8] presented an algorithm introducing a structural online control strategy, which has a considerably high efficiency in three types of road systems: unidirectional, bidirectional, and mixed. For the cell formation and intercellular integrated scheduling problems in the cellular manufacturing system where AGVs are used for transferring exceptional parts, Goli et al. [9] developed a fuzzy mixed-integer linear programming, designing an efficient whale Optimization Algorithm. Drótos [10] designed a centralized approach to maintain a central schedule to guarantee conflict-free operation of a large fleet of vehicles without deadlock or live lock, proposing a method based on local search to support the optimization of plans. For the multi-AGVs dispatching problem in a matrix manufacturing workshop, Zhang et al. [11] provided an improved iterated greedy algorithm, where an AGV route merging strategy and a workshop partition strategy are proposed to reduce travelling distance and the cost of AGVs, two rules to identify infeasible solutions are presented to save the operation time, four effective operators in the local search stage are applied to improve the solution quality, and a repair strategy is used to avoid the local optima. For the fleet sizing and routing problem with synchronization for AGVs with dynamic demands, Aziez et al. [12] used mathematical programming and a powerful matheuristic algorithm to cope with large instances and real-time operations, handling dynamic demand by replanning the routes immediately with the arrival of new requests.

In the automated container terminal, reasonable AGV routing is beneficial to the smoothness of the whole logistics transport. For the integrated scheduling problem of handling equipment and AGV, Yang et al. [13] set up a bilevel programming model to minimize the makespan, where AGV path planning is the lower level, and the integrated scheduling of quay cranes, AGVs, and automatic rail-mounted gantries is the upper level. They designed an effective bilevel general algorithm based on the preventive congestion rule. Aiming at the goal of minimizing multi-AGV delays, Zhong et al. [14] considered conflict-free path planning in simultaneous scheduling and presented a hybrid genetic algorithm-particle swarm optimization algorithm with fuzzy logic controller to adaptive auto-tuning which is more reliable than the genetic algorithm, especially on large-scale problems. Ji et al. [15] investigated the combinatorial optimization of two problems in the synchronous loading and unloading operation pattern, and established a programming model aiming at the layout and operation flow of most container terminals to minimize the completion time of all containers. They proposed two bilevel genetic algorithms based on conflict resolution strategy, introducing elitism preservation strategy, tabu list, and catastrophe operator. Zhou et al. [16] developed an improved anisotropic Q-learning routing algorithm to find the shortest-time paths in the guide-path network of crosslane type, considering AGV real-time status and designing selecting-action strategy based on AGV waiting time estimation.

Similar to manufacturing systems and container terminals, the application of AGVs in automated warehouses has greatly improved the efficiency in handling goods. Xing et al. [17] designed two conflict elimination methods and presented a novel tabu search algorithm. The algorithm adopts two novel neighborhood operators including relocation and exchange operations, which greatly optimizes the order of pickup points in the automatic warehouse and shortens the total travel distance. Fransen et al. [18] proposed a dynamic path planning approach to solve the real-time control problem of grid-based AGV systems in parcel sorting, baggage handling, and semiconductor fabrication and adopted a graph-representation of the system with constantly updating vertex weights, which can recover from deadlock situations. To solve the integration difficulty between scheduling and routing aspects of the multi-AGV problem, Lee et al. [19] established a model of combinatorial auction-based winner determination problem for multiple AGVs and decomposed the model into a two-phase problem, proposing a new genetic algorithm with knowledge-based operators to obtain a better solution. Chen et al. [20] set up an Ant-agent optimized by repulsive potential field for multi-AGV routing, designing a scheme of varying velocity to adjust the AGV velocity through decentralized control, and presenting a novel transition rule to determine the AGV motion state by combining centralized and decentralized control.

Most AGV path optimization problems only optimize one objective. However, there are often multiple conflicting optimization objectives existing in practice. When the number of AGVs used is small, the pickup time will be long; when the requirements for pickup speed are high, more AGVs will need to be started. Therefore, AGV quantity and

maximum pickup time are two conflicting goals, which are also the focus of this paper. A large number of studies have found that the population-based intelligent optimization algorithm can maintain a population of solutions in the iterative process, and has greater advantages in solving multi-objective optimization problems. Differential evolution (DE) algorithm has the advantages of simple structure, easy implementation, fast convergence, and strong robustness and are widely used in engineering practice, such as computer network, mechanical design and robotics, image processing, industrial control, biology, and other fields [21–23].

When the classical differential evolution algorithm is applied to solve optimization problems in some complex environments, it exists the phenomenon of low solution accuracy and poor stability. The improvement for the classical differential evolution algorithm mainly includes: adaptive parameter adjustment mechanism, mutation and crossover strategy adjustment technology, population structure design, and hybrid algorithm evolution mechanism. Zhao et al. [24] added the competitive evolutionary mechanism to mutation operation where the strategy to produce superior individuals are employed in population with larger sizes. Yu et al. [25] put forward an adaptive selection mutation method where individuals are selected by their fitness values and constraint violations, improving the exploitation, and maintaining the exploration. Ahmadianfar et al. [26] created a novel adaptive mutation mechanism, introducing the operators of particle swarm optimization algorithm to improve global search capability. Using the generalized Lehmer mean and linear bias reduction, Stanovov et al. [27] controlled the parameter adaptation bias for fitness improvement-based and distance-based DE algorithms. To exploit the evolutionary state of the population, Gupta and Su [28] proposed a mutation strategy that uses a dynamic number of fitness-based leading individuals, creating a novel way to set control parameters based on each individual's evolutionary state during trial generation. A new mapping mechanism of continuous and discrete variables was proposed by Ali et al. [29]. In addition, a repair method based on k-means clustering is designed to enhance the initial population, and an ensemble of mutation strategies is presented to improve the exploration ability. Wang et al. [30] used the accompanying population whose individuals are composed of suboptimal solutions to optimize the mutation strategy and control parameters, which realized the adaptation of the strategy and parameters of the main population. Besides, they utilized reverse individuals to enhanced population diversity in evolution and radial spatial projection technology to optimize the evolution direction. Tan et al. [31] presented an adaptive mutation operator based on fitness landscape where the relationship between three mutation strategies and fitness landscape features are trained by random forest offline, designing an adaptable parameter mechanism based on historical memory and a linear reduction method of population size. Based on five mutation strategies, Deng et al. [32] developed an optimal mutation strategy to improve local search ability while ensuring global search ability. To ensure diversity of solutions and accelerate convergence, they employed the wavelet basis function and normal distribution function to control the scaling factor and crossover rate, respectively. The crossover rate sorting mechanism introduced by Li et al. [33] enables each individual to be assigned a crossover rate value according to their fitness value, allowing good elements to be more inherited. A dynamic population reduction strategy is employed to enhance convergence speed and balance exploration and exploitation. A selection operator with three candidate vectors was proposed by Zeng et al. [34] for the escaping of the local optimal value when the individual is stuck in a state of stagnation. Sun et al. [35] designed the reverse learning mechanism for generating the initial subpopulations to improve the convergence velocity and maintain population diversity, developing a new multipopulation parallel control strategy to maintain the search efficiency in subpopulations. Deng et al. [36] applied the quantum chromosome encoding to enhance the population diversity and quantum rotation to speed up the convergence speed, using the cooperative coevolution framework to change the large-scale and high-dimensional complex optimization problem to multiple low-dimensional subproblems to improve the solution efficiency. Sun et al. [37] applied a finite-horizon Markov decision process to adaptively control the parameter, avoiding the problem of poor convergence caused by random values or empirical measurements. Houssein et al. [38] proposed a hybrid algorithm of slime mould algorithm and DE Algorithm where an adaptive guided mutation method improve the swarm agents' local search, enhance the diversity of population, and prevent premature convergence.

The performance improvement of the traditional differential evolution algorithm can be implemented by considering the characteristics of complex problems and designing strategies for new coding structure and decoding scheme, mutation operator, crossover operator, and evolution mechanism. To our knowledge, there are no known studies that apply DE algorithms to solve such biobjective AGV routing problem, therefore this paper presents a nondominant sorting differential evolution (NSDE) algorithm with a well-designed encoding and decoding method, introducing an elitist selection strategy. By discussing the characteristics of automated warehouse layout and AGV driving mode, a new encoding and decoding method based on pickup location is designed, which can effectively improve the decoding quality. Nondominated sorting genetic algorithm II (NSGA-II) is one of the most influential and widely applied multi-objective genetic algorithms, based on controlled elitism concepts. In the iteration process, the elite strategy of NSGA-II is introduced to accelerate population convergence and maintain the quality and Pareto diversity of the population.

The remaining sections are organized as follows: First, the mathematical formulation of biobjective multi-AGV routing problem is provided. Then, the NSDE algorithm is detailed. Afterwards, the paper introduces the numerical experiments and results. Lastly, a summary is given.

2. Problem Statement

This section introduces the working environment and process of AGV and the basic concept of multi-objective problem, leading to the proposed biobjective multi-AGV routing problem and mathematical model.

2.1. Workflow Description for AGV. Figure 1 shows the layout of a common automated warehouse which is represented by the grid method [39], divided into 10 rows and 15 columns, a total of 150 square grids, and numbered in order. To maximize the use of warehouse space, the passageway is only designed as the width of one grid, and each shelf occupies two columns. The white area and gray area are driving passageways (transportation passageways) of AGVs and shelves, respectively. The entrances of AGVs are concentrated in the first row in the figure. When AGVs enter the warehouse, they are arranged from left to right, and the exit is located in the grid of the lower right corner.

When AGV works, it starts from its own entrance and runs in the passageway between shelves. According to the requirements of the order, AGV takes out the corresponding goods from the shelves and delivers them to the exit. The goods involved in the order are called the pickup point. When picking up goods, AGV requires driving to the grid next to the pickup point. The grid like this is called the pickup position of the pickup point (for example, the pickup position of pickup point 51 is grid 50), and the passageway located is the pickup passageway of the pickup point.

2.2. Multi-Objective Optimization Problem. In multi-objective optimization problem (MOP), multiple objectives conflict with each other, and the performance of one objective is often improved at the expense of the performance of other objectives. In fact, it is impossible for MOP to have a solution that can achieve the optimal state of all objects, and the solution can usually be obtained from a set of non-inferior solutions, i.e., Pareto solution set [40].

For MOP, suppose there are m optimization objectives $f_i(x)$, $i = 1, 2, \ldots, m$, and each optimization objective is to minimize. If any two feasible solutions a and b satisfy the following relationship, then solution a dominates solution b, which can be expressed as $a > b$. Otherwise, there is no dominating relationship between the two solutions as follows:

(1) For any optimization objective i, $f_i(a) \le f_i(b)$;

(2) There is at least one optimization objective u, $f_u(a) < f_u(b)$ is established.

When there are multiple Pareto optimal solutions, if more information about the problem is unknown, it is difficult to decide which solution is more desirable, i.e., all Pareto optimal solutions are considered to be equally important. Therefore, for MOP, the most important task is to find as many Pareto optimal solutions as possible, mainly completing the following two tasks:

(1) Find a set of solutions as close to Pareto optimal front as possible.

(2) Find a set of solutions that are as different as possible.

2.3. Problem Description and Mathematical Formulation. The biobjective multi-AGV routing problem is described as follows: enough AGVs are available and n goods need to be taken out according to a batch of customer orders. AGVs start from their respective entrances, drive along the passageway, take out the goods involved in the order, and finally transport them to the exit. Collision is not allowed during AGV driving, i.e., any grid can only be occupied by one AGV at the same time. The goal of the problem is to simultaneously minimize two objects, i.e., the number of AGVs used and the maximum pickup time of AGVs, so as to obtain the conflict-free driving path of AGVs. This problem is based on the following assumptions:

(1) The time when the goods is taken out of the shelf and put into the AGV is not considered.

(2) Only focus the case that the quality or volume of goods is small enough and will not exceed the AGV capacity, regardless of capacity limitation of AGV.

(3) Any AGV runs at a constant speed, and turning time and acceleration are ignored.

(4) Both the distance and travel time between adjacent grids are set to 1.

The following notations are introduced for establishing the mixed-integer programming (MIP) model:

r The number of grid rows in the warehouse layout diagram.

c The number of grid columns in the warehouse layout diagram.

r_i The row number of pickup location of the pickup point i (entrance, or exit i), $r_i \in [1, r]$.

c_i The column number of pickup location of the pickup point i (entrance, or exit i), $c_i \in [1, c]$.

n The number of pickup points.

m Maximum number of AGVs, $m = \min\{c, n\}$.

N A collection of nodes that contain entrances, exit and all pickup points. $N = \{0, 1, \ldots, n, n+1\}$. Node 0 is the entrance, nodes $1, \ldots, n$ are the pickup points, and node $n+1$ is the exit.

N' A collection of all pickup points. $N' = \{1, 2, \ldots, n\}$.

M A collection of AGVs. $M = \{1, \ldots, m\}$.

t_{ij} Transport time from node i to node j.

C_k Pickup time of AGV k.

$C_{\max}$ Maximum pickup time.

The decision variable is:

$$x_{ijk} = \begin{cases} 1, & AGV\ k\ accesses\ node\ j\ directly\ after\ node\ i, i \neq j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

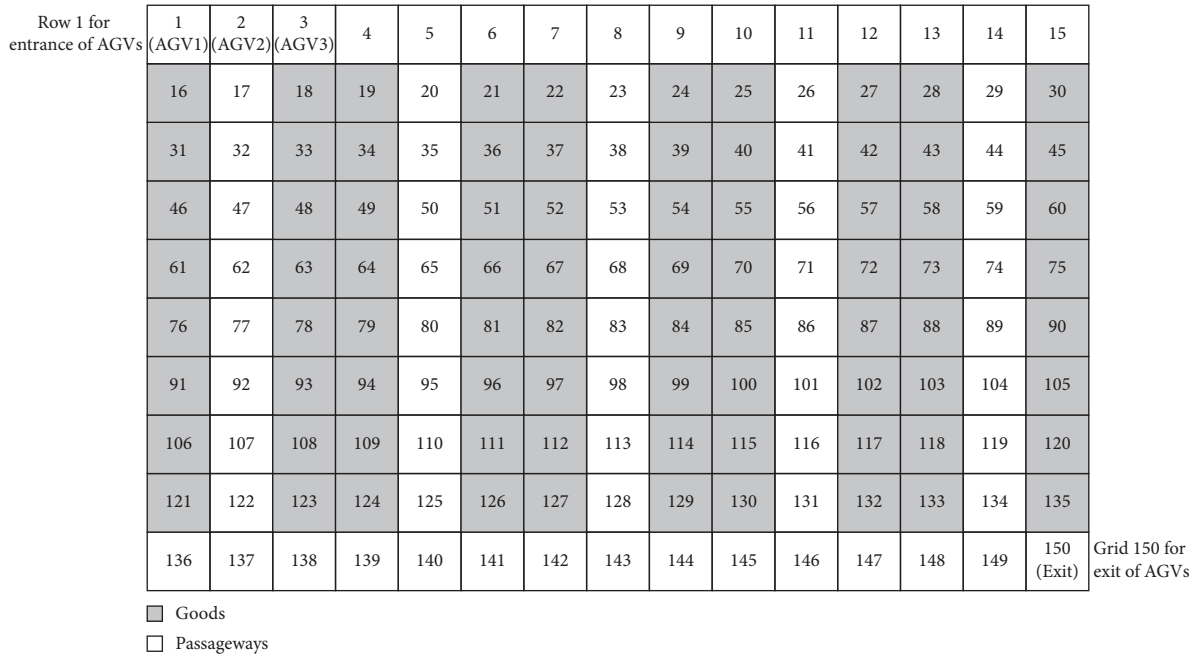

Figure 1: The layout of automated warehouse.

The MIP model of the problem is provided as follows:

$$\begin{aligned}&\text{Minimize}\{f_1(x), f_2(x)\},\\&\text{Subject to,}\end{aligned} \tag{2}$$

$$f_1(x) = \sum_{k=1}^{m}\sum_{j=1}^{n} x_{0jk}, \tag{3}$$

$$f_2(x) = \max_{k\in M}\{C_k\}, \tag{4}$$

$$\sum_{j=1}^{n} x_{0j1} = 1, \tag{5}$$

$$\sum_{j=1}^{n} x_{0jk} \le \sum_{j=1}^{n} x_{0j(k-1)}, k = 2, \ldots, m, \tag{6}$$

$$\sum_{j=1}^{n} x_{0jk} = \sum_{j=1}^{n} x_{i(n+1)k}, k \in M, \tag{7}$$

$$x_{j0k} = 0, j \in N\prime, k \in M, \tag{8}$$

$$x_{(n+1)jk} = 0, j \in N\prime, k \in M, \tag{9}$$

$$x_{0(n+1)k} = 0, k \in M, \tag{10}$$

$$x_{(n+1)0k} = 0, k \in M, \tag{11}$$

$$x_{iik} = 0, i \in N, k \in M, \tag{12}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n} x_{ijk} = \sum_{k=1}^{m}\sum_{i=1}^{n} x_{jik} = 1, j \in N, \tag{13}$$

$$\sum_{k\in M}\sum_{i\in S}\sum_{j\in S} x_{ijk} \le |S|, S\subseteq N', 2 \le |S| \le n-1, \tag{14}$$

$$C_k = \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}x_{ijk}, \tag{15}$$

$$x_{ijk} \in \{0,1\}, i \in N, j \in N, k \in M. \tag{16}$$

The two objectives of optimization are represented by (2): simultaneous minimization of the number of AGVs used and the maximum pickup time; (3) is the objective function of the number of AGV used; (4) is the objective function of maximum pickup time; constraints (4)–(5) ensure that at least one AGV is working; constraint (7) means that entrance and exit need to be arranged for the working AGV, otherwise they will not be arranged; constraints (7)–(10), respectively, illustrate that the following operations are not allowed in the pickup sequence of AGV: return the entrance from the pickup point, return the pickup point from the exit, go directly to the exit from the entrance, and go directly to the entrance from the exit; constraints (11)–(12) indicate that the AGV should leave a node after accessing the node; constraint (14) eliminates subtours; formula (15) is the calculation formula of the pickup time of each AGV; constraints (16) provides decision variable constraint.

What needs illustration is that t_{ij} is the sum of the shortest distance between two nodes i and j plus AGV waiting time, where the waiting time is due to AGV conflict. For concrete details about calculating the shortest path

between two nodes i and j and the solutions of conflicts, reader can consult the study of Xing et al. [17].

3. Nondominated Sorting Differential Evolution Algorithm

The differential evolution algorithm proposed by Storn and Price [41] is a population-based optimization algorithm, one of the global intelligent optimization algorithms to solve continuous optimization problems. For the biobjective problem proposed, a NSDE algorithm is presented. Firstly, an effective encoding and decoding method is designed to eliminate the influence of repeated paths, and then the fast nondominated sorting method and elite selection strategy of NSGA-II are introduced to improve the selection operation. Excellent individuals are selected from the two generations of the population for iteration, so that the Pareto solution set is continuously updated. The procedure of the NSDE algorithm is as follows:

3.1. Encoding and Decoding. Since the driving passageways in an automated warehouse are limited by the position of shelves, and AGV has no capacity limitation and does not need to deal with the situation of exiting, unloading, and reentering if it is full, a relatively superior solution should satisfy the following two rules to reduce duplicate paths:

> Rule 1: The driving direction on the horizontal passageway should be consistent.
>
> Rule 2: Reversals on the vertical passageway should be as little as possible.

Regardless of conflicts, for instance, the pickup sequence (123, 126, 129, and 132) is better than (123, 129, 126, and 132) and the pickup sequence (55, 70, 85, 100) is superior to (55, 85, 70, 100) in Figure 1. Figures 2 and 3 clearly display the path arrangement and driving directions on the horizontal and vertical passageways for these two examples, respectively.

Therefore, applying these two rules can reduce the number of times through the same grid, i.e., decrease repeated paths and reduce the time wasted to some extent. A novel coding and decoding method with the rule of avoiding duplicate paths is designed.

3.1.1. Encoding Method. The pickup points involved in each vertical pickup passageway should be taken out by two AGVs at most. For example, the goods in the upper and lower sections of the vertical passageway can be picked up by two AGVs, respectively. If another AGV is added to work in this passageway, its path would be identical to that of one of the previous two AGVs. For applying in the discrete, the individual in the NSDE algorithm is expressed as a region-based sequence. Therefore, the coding method proposed firstly divides the pickup points related to each vertical pickup passageway into upper and lower areas, which are represented by two numbers, respectively. The pickup points corresponding to each number are arranged in increasing order, so the individual length is equal to the number of vertical passageways multiplied by 2. There are several exceptional cases that need to be explained:

(1) If there is only one pickup point for a certain vertical passageway, it will be assigned to the corresponding number according to the area it is located, and the other number code related to this passageway has no meaning, but only exists for the convenience of calculation.

(2) If the pickup points of a certain vertical passageway are only distributed in the upper half or the lower half, and there are two or more pickup points, in order to make the two digital codes related to this passageway meaningful, the regional number without pickup points contains the nearest pickup point to this region.

(3) If the row number of the warehouse is odd and there are some pickup points in the middle row, they can be distributed to the area that these pickup points are closer to, and if the distances are the same or their pickup passageway only involves these pickup points, they can be distributed to any area.

(4) If not every vertical passageway contains pickup points, the number corresponding to the passageway without pickup points does not represent any pickup point, but it can play the role of dividing AGV in the decoding stage to be introduced below.

Figure 4 indicates the goods numbers of a batch of orders only, and divides the areas. It is assumed that the pickup points are distributed in five vertical passageways, where 1 to 5 represents the pickup points in the top area of each passageway, and 6 to 10 presents the pickup points in the area below. According to the above coding mode, the meaning of each coded number, i.e., the corresponding pickup points can be obtained in Table 1. The current meaning of coded numbers is the initial state before decoding, which will be adjusted during decoding to optimize the path.

3.1.2. Decoding Method. In the decoding phase, the region number in an individual should be converted into the pickup sequence of one or more AGVs, and the picking points should be adjusted across regions to make decoded paths better. In order to reduce the repeated paths on the horizontal passageway, AGV will not turn around on the horizontal passageway when arranging the pickup sequence, i.e., rule 1 should be observed under the background of the warehouse layout in this paper. In the following process, the pickup points represented by each digital code are sorted from small to large. When calculating the pickup time, connect the pickup point sequences represented by the numbers. If the flip sequence can make the pickup sequence closer to the previous pickup point, flip it. The following is a detailed introduction to the decoding process.

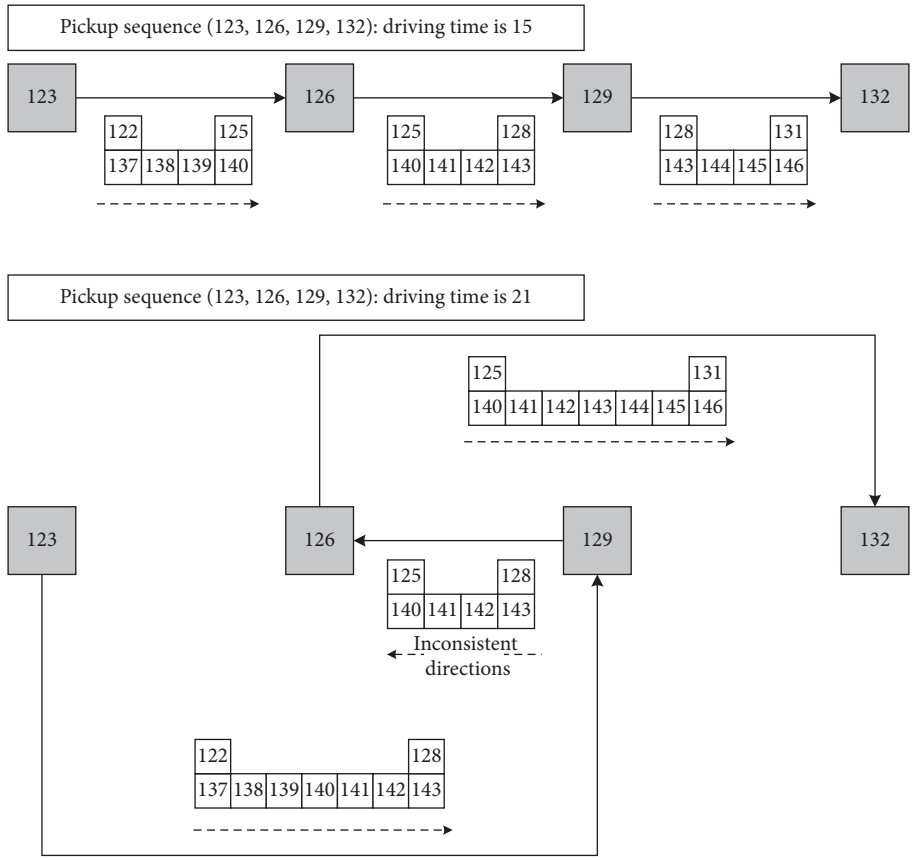

Figure 2: An instance for Rule 1.

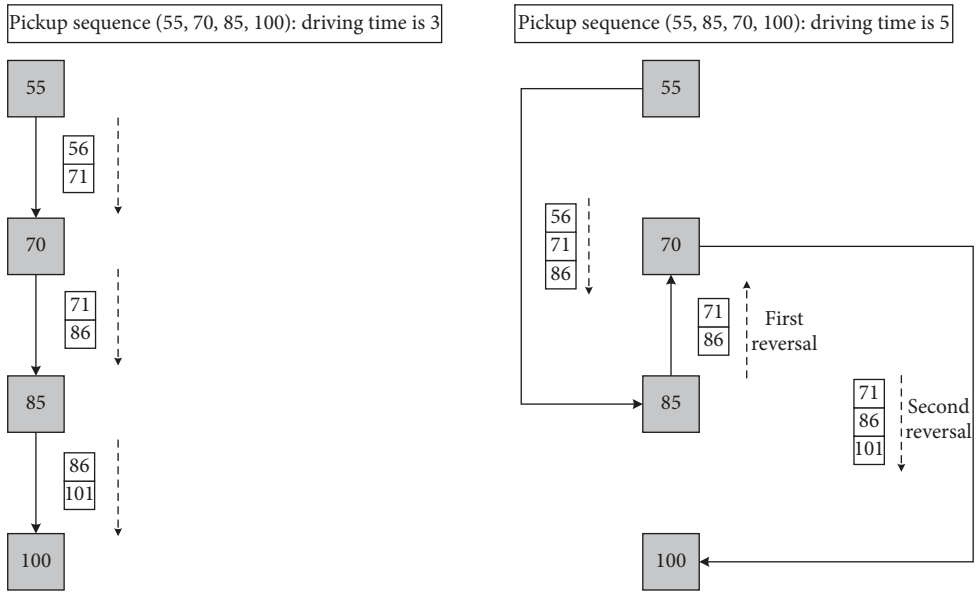

Figure 3: An instance for Rule 2.

(1) Firstly, divide the individuals according to the vertical passageway position and calculate the number of AGVs: compare the serial numbers of the pickup passageways corresponding to the adjacent numbers in the individual in turn, if the preceding value is greater than the following value, divide them here and prepare to start a new AGV. There is no need to arrange AGV for numbers without meaning, which only plays a certain role in the division. Such as the individual [3, 4, 9, 1, 6, 5, 10, 2, 7, 8], the

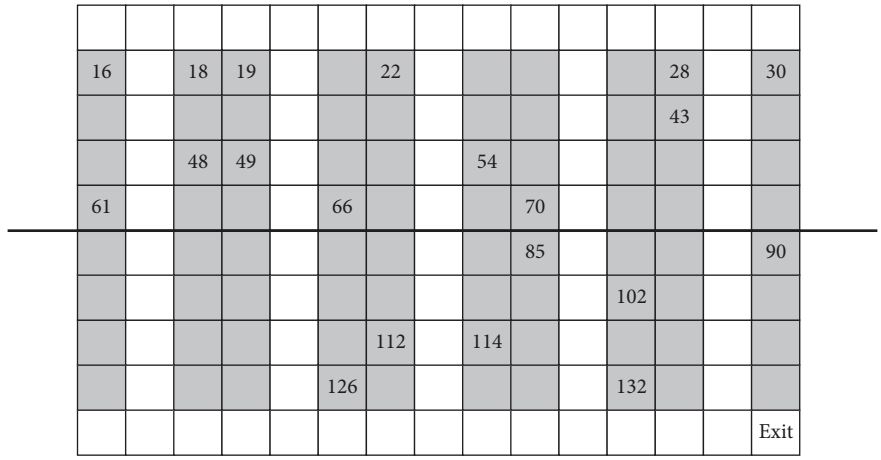

Figure 4: The partition of regions in automatic warehouse.

Table 1: Number coding and meaning before decoding.

Upper region of passageway		Lower region of passageway	
Number coding	Corresponding pickup points	Number coding	Corresponding pickup points
1	16, 18, 48	6	61
2	19, 49, 66	7	126
3	22, 54	8	112, 114
4	70	9	85, 102, 132
5	28, 30, 43	10	90

corresponding vertical passageway numbers and areas are: 3^{up}-4^{up}-4_{down}-1^{up}-1_{down}-5^{up}-5_{down}-2^{up}-2_{down}-3_{down}. According to the size relations, the individual is divided into three sections: 3^{up}-4^{up}-4_{down}, 1^{up}-1_{down} -5^{up}-5_{down}, and 2^{up}-2_{down}-3_{down}.

(2) Assign AGV (entrance) to several sequence fragments after individual division: allocate the entrance from left to right according to the first number (the serial number of the vertical passageway) of each sequence fragment in nondecreasing order. For the example in (1), the following result of allocation can be obtained:

AGV 1: 1^{up}-1_{down} -5^{up}-5_{down};
AGV 2: 2^{up}-2_{down}-3_{down};
AGV 3: 3^{up}-4^{up}-4_{down}.

(3) Adjust pickup points represented by digital code: Since the objective of maximum pickup time needs to be optimized after the determination of AGV quantity, the pickup points represented by each digital code should be adjusted. According to the nondecreasing order of the vertical passageway number, cycle to adjust pickup points: if pickup points of one vertical passageway are allocated to two AGVs, adjust the pickup points of the AGV with a longer pickup time to the other AGV with a shorter pickup time. For each move, the nearest pickup point to the AGV with a shorter pickup time is selected, which actually adjusts pickup points included in the digital code. Choose a solution that minimizes the maximum pickup time of the two AGVs, so as to narrow the pickup time gap between them. An instance is presented as follows: for the allocation result in (2), first check whether two AGVs are allocated for the same aisle from aisle 1 to Aisle 5: only the goods in aisle 3 meet this condition and are allocated to AGV 2 and AGV 3. Suppose digital codes $3^{up} = \{22, 54\}$ and $3_{down} = \{112, 114\}$ and then adjust the encoding: if pickup times of AGV 2 and AGV 3 meets $C_2 > C_3$, the goods represented by 3_{down} in AGV 2 are reduced and the digital codes are attempted to adjust to $3^{up} = \{22, 54, 112\}$ and $3_{down} = \{114\}$. Similarly, if pickup times of AGV 2 and AGV 3 meets $C_2 < C_3$, the goods represented by 3^{up} in AGV 3 are reduced and the digital codes are attempted to adjust to $3^{up} = \{22\}$ and $3_{down} = \{54, 112, 114\}$. If the $\max\{C_2, C_3\}$ becomes smaller after adjustment, the above adjustment will be adopted; otherwise, it will be restored to the original codes, i.e., the scheme with the minimum difference between the pick times of two AGVs.

(4) Adjust entrance: when there is no adjustable scheme in (3), find out the AGV with the longest pickup time: if its entrance is the nearest, stop adjustment and enter (5); otherwise, try to swap other AGV entrance for it: if the maximum pickup time of the two AGVs can be shortened, the shortest solution is selected, and then return (3); otherwise, stop the adjustment and go to (5). An instance is provided as follows: suppose the following results are obtained after step (3) (only the numbers of goods needed for calculation are given):

AGV 1: $1^{up}(16, 18, 48)\text{-}1_{down}\ \text{-}5^{up}\text{-}5_{down}$;
AGV 2: $2^{up}(19, 49, 66)\text{-}2_{down}\text{-}3_{down}$;
AGV 3: $3^{up}\text{-}4^{up}\text{-}4_{down}$.

The pickup time of AGV 1 is assumed to the maximum pickup time, i.e., $C_1 = \max\{C_1, C_2, C_3\}$, then the nearest entrance 2 is tried to arrange for the route of AGV 1, i.e., AGV 1: $2^{up}(19, 49,$ and $66)\text{-}2_{down}\text{-}3_{down}$, AGV 2: $1^{up}(16, 18,$ and $48)\text{-}1_{down}\ \text{-}5^{up}\text{-}5_{down}$. If the $\max\{C_1, C_2\}$ of the two AGVs becomes smaller after adjustment, this adjustment should be taken and all entrances will be further checked similarly; otherwise, the original route will be restored.

(5) Check and resolve conflicts, and finally calculate the two objective values of the individual. (There are two types of AGV conflicts: (1) the uniform direction conflict can be eliminated by waiting; (2) the opposite direction conflict should be solved by exchanging the subsequent pickup points of the collision grid. For concrete details about the solutions of conflicts, reader can consult the study of Xing et al. [17].)

The decoding result of the sample individual in (1) can be obtained: the number of AGVs is 3, the maximum pickup time is 30 and pickup sequences are as follows:

AGV 1: 1-22-54-70-85-102-132–150;
AGV 2: 2-16-18-48-61-28-30-43-90–150;
AGV 3: 3-19-49-66-126-112-114–150.

3.2. Initial Population Generation Method. The target population composes of Λ target individuals. The hth target individual is $X_h^\tau = [x_{h,1}^\tau, x_{h,2}^\tau, \ldots, x_{h,n}^\tau]$, where τ is the current iteration number, $h = 1, 2, \ldots, \Lambda$, and Λ is the population size. Two initial individuals that are easy to calculate are given in the initial population.

In the first individual X_1^0, AGV number is 1. First, sort the pickup points involved in each vertical passageway in increased order and this sequence will make the AGVs drive in one direction when picking up the goods from the same passageway. Then connect these sequences by following the order of the pickup passageway from left to right. The sequence can be reversed if reverse make the sequence closer to the previous pickup point. Finally, the individual corresponding to the obtained AGV pickup sequence is taken as the initial individual X_1^0.

In the second individual X_2^0, AGV quantity is the number of vertical passageways, i.e., each AGV is responsible for only pickup points of one passageway respectively, and takes the goods in increasing order of the pickup point. The individual corresponding to the obtained AGV pickup sequences is taken as the initial individual X_2^0.

The remaining individuals of the initial population are randomly generated. They are divided into two halves at first, and then generated from the two initial individuals X_1^0 and $X_{2,}^0$ respectively. Repeatedly execute the operation of exchanging randomly two gene locations in the new obtained individual until the initial population is generated.

3.3. Mutation, Crossover, and Selection Operations. In the iterative process, individuals in the parent population are successively mutated, crossed, and selected [42]. The detailed processes for these operations are provided in the following subsections:

3.3.1. Mutation Operation. The mutation process of NSDE resembles that of the classic DE algorithm. Four parent individuals and one Pareto individual are randomly selected for mutation operation to generate one mutation individual that is expressed as $V_h^\tau = [v_{h,1}^\tau, v_{h,2}^\tau, \ldots, v_{h,n}^\tau]$. For each target individual in parent population $X_h^{\tau-1}$, the relevant mutation individual is generated as by (17):

$$V_h^\tau = X_{Pareto}^{\tau-1} \oplus Z \otimes \left(X_{a_1}^{\tau-1} - X_{b_1}^{\tau-1}\right) \oplus Z \otimes \left(X_{a_2}^{\tau-1} - X_{b_2}^{\tau-1}\right), \quad (17)$$

where $X_{Pareto}^{\tau-1}$ is a Pareto individual chosen at random from the current Pareto frontier individual set, $X_{a_1}^{\tau-1}$, $X_{b_1}^{\tau-1}$, $X_{a_2}^{\tau-1}$, and $X_{b_2}^{\tau-1}$ are four different parent individuals at iteration $(\tau\text{-}1)$, a_1, b_1, a_2, and b_2 are random integers in $[1, \Lambda]$, and $Z \in [0, 1]$ is a mutation coefficient. The operator $\otimes$ is calculated by equation (18):

$$\begin{aligned} G_h^\tau &= Z \otimes \left(X_a^{\tau-1} - X_b^{\tau-1}\right) \Leftrightarrow g_{h,q}^\tau \\ &= \begin{cases} x_{a,q}^{\tau-1} - x_{b,q}^{\tau-1}, & if\ ran\ d < Z'', \\ 0, & otherwise. \end{cases} \end{aligned} \quad (18)$$

where $G_h^\tau = [g_{h,1}^\tau, g_{h,2}^\tau, \ldots, g_{h,n}^\tau]$ is a temporary individual. The operator $\oplus$ is calculated by (19):

$$\begin{aligned} V_h^\tau &= X_{Pareto}^{\tau-1} \oplus G_{h_1}^\tau \oplus G_{h_2}^\tau \Leftrightarrow v_{h,q}^\tau = X_{Pareto,q}^{\tau-1} \oplus g_{h_1,q}^\tau \oplus g_{h_2,q}^\tau \\ &= \mathrm{mod}\left(\left(X_{Pareto,q}^{\tau-1} + g_{h_1,q}^\tau + g_{h_2,q}^\tau + n - 1\right), n\right) + 1, \end{aligned} \quad (19)$$

where operator mod is modulus. Individual V_h^τ maybe not valid because some encoding elements may be repeated or loss. An instance is provided in Tables 2–4 to explain the mutation process.

3.3.2. Crossover Operation. A mutant individual can increase the perturbation of the target individual and jump out of local optimality. A valid trial individual $U_h^\tau = [u_{h,1}^\tau, u_{h,2}^\tau, \ldots, u_{h,n}^\tau]$ can be obtained by the crossover operation after mutation operation [43]. In the crossover operation process, select parts of the mutant individual V_h^τ and add it into the parent individual $X_h^{\tau-1}$ to obtain a trial individual U_h^τ. A three-point insertion method is applied to enhance the perturbation. The insertion points $p1$, $p2$, and $p3$ are three different locations of the target individual, where three parts of the mutant individual are inserted, and they are different random integers in $[1, n]$. For each gene number of V_h^τ, if a random number $\mathrm{rand}() < Y$, it can be

Table 2: Instance for the mutation operation ($Z = 0.3$) (1).

$X_{a_1}^{\tau-1} - X_{b_1}^{\tau-1}$										
$X_{a_1}^{\tau-1}$	10	7	5	9	1	4	3	8	2	6
$X_{b_1}^{\tau-1}$	9	3	1	10	6	5	7	8	2	4
$X_{a_1}^{\tau-1} - X_{b_1}^{\tau-1}$	1	4	4	−1	−5	−1	−4	0	0	2
$G_{h_1}^{\tau}$ rand	0.18	0.49	0.27	0.54	0.59	0.84	0.33	0.91	0.15	0.63
$X_{a_1}^{\tau-1} - X_{b_1}^{\tau-1}$	1	4	4	−1	−5	−1	−4	0	0	2
$G_{h_1}^{\tau}$	1	0	4	0	0	0	0	0	0	0

Table 3: Instance for the mutation operation ($Z = 0.3$) (2).

$X_{a_2}^{\tau-1} - X_{b_2}^{\tau-1}$										
$X_{a_2}^{\tau-1}$	2	7	10	5	3	6	8	4	9	1
$X_{b_2}^{\tau-1}$	4	1	8	2	3	6	10	7	9	5
$X_{a_2}^{\tau-1} - X_{b_2}^{\tau-1}$	−2	6	2	3	0	0	−2	−3	0	−4
$G_{h_1}^{\tau}$ rand	0.20	0.49	0.19	0.18	0.73	0.41	0.51	0.68	0.57	0.41
$X_{a_2}^{\tau-1} - X_{b_2}^{\tau-1}$	−2	6	2	3	0	0	−2	−3	0	−4
$G_{h_2}^{\tau}$	−2	0	2	3	0	0	0	0	0	0

Table 4: Instance for the mutation operation ($Z = 0.3$) (3).

V_h^{τ}										
$X_{Pareto}^{\tau-1}$	1	3	10	4	5	7	9	2	8	6
$G_{h_1}^{\tau}$	1	0	4	0	0	0	0	0	0	0
$G_{h_2}^{\tau}$	−2	0	2	3	0	0	0	0	0	0
V_h^{τ}	10	3	6	7	5	7	9	2	8	6

added into $X_h^{\tau-1}$, where $Y \in [0, 1]$ is a crossover coefficient; otherwise, it is eliminated. The gene number may exist more than once in U_h^{τ} are deleted from right to left, which ensures that each gene number in U_h^{τ} appears once only.

For V_h^{τ} in Table 4, the crossover operation of $X_h^{\tau-1}$ is displayed in Table 5. The crossover coefficient $Y = 0.6$ and the insertion points are $p1 = 2$, $p2 = 4$ and $p3 = 8$.

3.3.3. Non Dominated Sorting Selection Operation. The selection process of NSDE introduced the fast nondominated sorting method, crowding comparison, and elite selection strategy in NSGA-II. After mutation and crossover, parent population $V^{\tau-1}$ and progeny population U^{τ} are merged into the set C^{τ} whose size is $\Lambda \times 2$. The main process of fast nondominated sorting of individuals in the two generation population set C^{τ} is as follows:

(1) First, judge the dominate relationship among all individuals in C^{τ}. For any individual X in C^{τ}, two parameters S_X and n_X need to be calculated, where S_X is a set that stores all individuals dominated by individual X and its initial value is null; n_X is a variable that records the quantity of all individuals in C^{τ} that can dominate X and its initialization value is 0.

(2) Perform fast nondominated sorting, calculate the nondominated rank (layer) rank_X of each individual, and use the set F_i to store the result of each rank after sorting. When $n_X = 0$, no individual in C^{τ} can dominate X and all individuals conforming to $n_X = 0$ in C^{τ} are taken out and put into set F_1 that is used to collect all individuals with the highest nondominated rank in C^{τ}. Then assign each individual in F_1 the nondominated rank $\text{rank}_X = 1$. For the individual set S_X dominated by each individual X in F_1, the corresponding parameter $n_{X'}$ of each individual X' in S_X minus 1. If $n_{X'} - 1 = 0$, X' is the individual with the highest nondominated rank in current individual set C^{τ}. Then delete individual X' from C^{τ} and add it into set F_2. Similarly, continue the above operation until C^{τ} is empty to stop [44]. Tables 6 and 7 is an example of fast nondominated sorting, in which Table 6 represents the objective function value and dominance relation of individuals in the two generations of population, and Table 7 represents the calculation results of nondominated sorting rank.

After the fast nondominated sorting of individuals in C^{τ}, each individual is divided into different levels. In the selecting process of the new target population, it is necessary to inherit as many genes of excellent individuals as possible to the next generation to ensure the good distribution of the population. To judge the environment density of individual, the crowding distance is used to calculate the Euclidean distance between the individual and two adjacent individuals. The greater the crowding distance, the better the individual distribution.

The calculation method of crowding distance is as follows: The individuals are sorted in ascending order according to each objective function value of them. The crowding distance of the first and last individuals are all set to infinity, and the crowding distance cd_i of individual X_i ($2 \le i \le \Lambda \times 2-1$) is calculated by the equation below.

Table 5: Instance for crossover Operation ($Y = 0.6$).

V_h^{τ}										
$X_h^{\tau-1}$	9	6	2	7	1	10	8	3	4	5
U_h^{τ}	9	6	2	7	1	10	8	3	4	5
V_h^{τ}	10	3	6	7	5	7	9	2	8	6
rand	0.25	0.87	0.63	0.44	0.63	0.95	0.51	0.69	0.51	0.71
$V_h^{\tau'}$	10			7			9		8	
$U_h^{\tau'}$										
$U_{h_1}^{\tau}$	9	6	2	7	1	10	8	3	4	5
$V_{h_1}^{\tau}$	10	7	9	8						
U_h^{τ}	9	6 10	7	2 7	9	1	10 8	3	8 4	5
$U_{h_1}^{\tau}$										
U_h^{τ}	9	6 10	7	2 7	9	1	10 8	3	8 4	5
U_h^{τ}	9	6	10	7	2	1	8	3	4	5

Table 6: Dominance relation in C^{τ}.

Individual in C^{τ}	Objective function value	n_X	S_X
X_1	(1, 100)	0	{}
X_2	(2, 72)	0	$\{X_3, X_5, X_7\}$
X_3	(2, 84)	1	$\{X_5\}$
X_4	(3, 67)	0	$\{X_5, X_7\}$
X_5	(4, 85)	3	{}
X_6	(6, 60)	0	{}
X_7	(3, 77)	2	{}
X_8	(5, 61)	0	{}

Table 7: Nondominant sorting results.

Nondominated rank	Sorting result				
F_1	X_1	X_2	X_4	X_6	X_8
F_2	X_3	X_7			
F_3	X_5				

$$cd_i = \sum_{j=1}^{n_f} \frac{f_j(i+1) - f_j(i-1)}{f_j^{\max} - f_j^{\min}}. \tag{20}$$

For the jth objective, $f_j(i-1)$ and $f_j(i+1)$ are the objective function values of the $(i-1)$th and $(i+1)$th individuals after sorting, and $f_j^{\min}$ and $f_j^{\max}$ are the minimum and maximum values of the jth objective function value, respectively. n_f is the number of objective functions.

The selection operation of NSDE algorithm adopts the elite strategy, i.e., integrates the two generations of population set C^{τ} after fast nondominated sorting and crowding distance calculation, and half of them are reserved as the next parent population. Priority is given to individuals in the higher level, and those with large crowding distance are preferred in the process of selecting individuals in the same level to ensure the diversity of population. For the example in Table 6, four individuals need to be selected. The crowding distance in first layer F_1 are calculated as $cd_1 = \infty$, $cd_2 = 0.45$, $cd_4 = 0.475$, $cd_6 = \infty$, and $cd_8 = 0.575$. The four individuals with the largest crowding distance are selected as X_1, X_4, X_6, X_8.

3.4. Algorithm Flowchart. The specific process of NSDE algorithm is as follows:

Step 1: Set Algorithm parameters: population size Λ, mutation coefficient Z, crossover coefficient Y and maximum iteration $\tau_{\max}$ and set the Pareto solution set to be null.

Step 2: Generate initial population and update Pareto solution set.

Step 3: Set the number of iterations $\tau = 0$ and evolution begins.

Step 4: Mutation and crossover operations are performed to produce progeny individuals.

Step 5: Execute a nondominated sorting on the collection C^{τ} which merges the current population and the offspring population to obtain nondominated layers F_i ($i = 1, 2, 3, \ldots$).

Step 6: The individuals in each nondominated layer F_i ($i = 1, 2, 3, \ldots$) are put into the parent population in turn to replace the previous ones. For each F_i, calculate whether it will exceed the population size before putting it in population: if it exceeds, calculate the number of individuals n' that can be put into the target population, calculate the crowding distance for each individual in F_i, select the best n' individuals, and go to Step 7.

Step 7: Update Pareto solution set. If $\tau < \tau_{\max}$, $\tau = \tau + 1$ and return to Step 3, otherwise go to Step 8;

Step 8: Output Pareto solution set and stop.

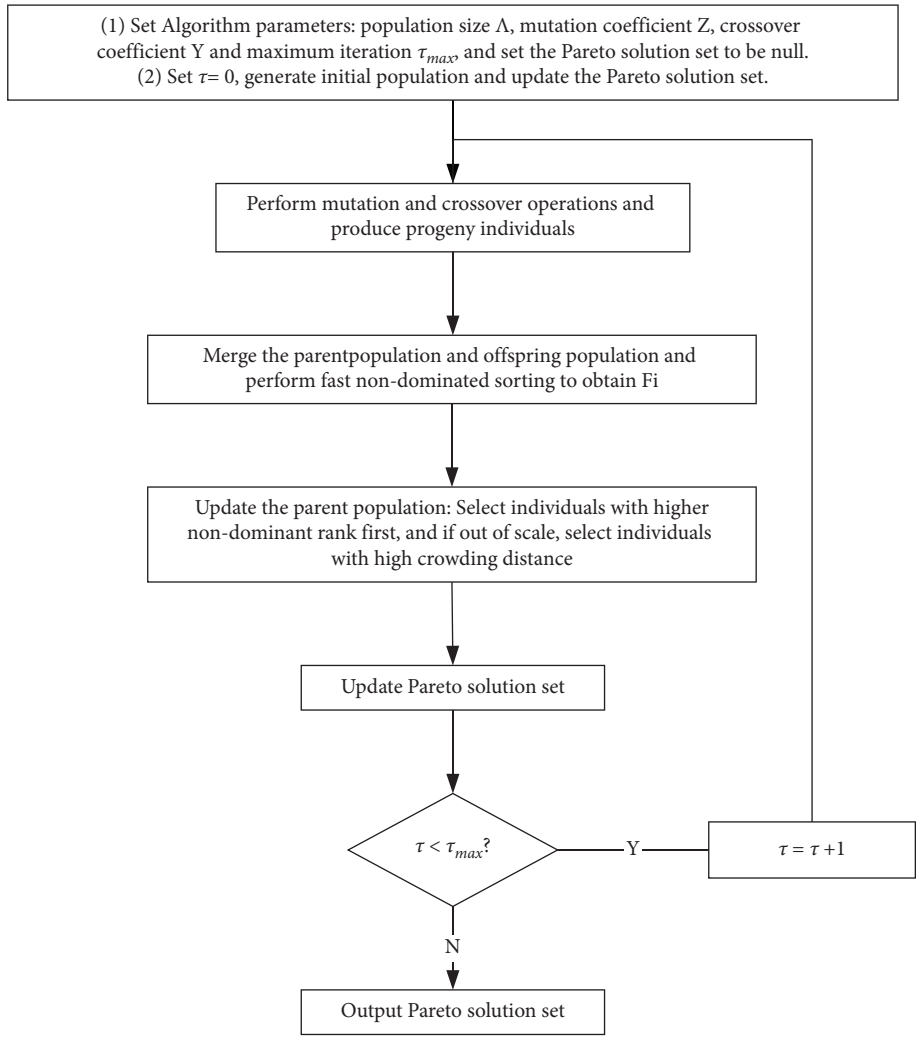

Figure 5: NSDE algorithm flowchart.

The flowchart of NSDE algorithm is expressed in Figure 5.

4. Computational Results

In order to verify the performance of the proposed NSDE algorithm, this section compares it with Gurobi and multi-objective tabu search (MOTS) algorithm [17], respectively. All algorithms are implemented in C++ programming language and run on a PC with an Intel Corei7 CPU (2.6 GHz × 12) and 8 GB RAM. The relevant parameters of NSDE algorithm are measured by orthogonal experimental design method: mutation coefficient: 0.1, crossover coefficient: 0.3, population size: 50, and maximum number of iterations: 400. The relevant parameters of MOTS algorithm are set by orthogonal experimental design method: tabu list length: 20, maximum number of iterations: 500. The process of the orthogonal experimental design for the NSDE algorithm is presented in Appendix .

The performance indicators for multi-objective evolutionary algorithms are generally categorized into four groups based on performance criteria, i.e., counting indicators, convergence indicators, diversity indicators, and comprehensive indicators [45, 46], in which each indicator is selected in this paper. Four evaluation indexes used to evaluate the Pareto solution set obtained by multi-objective algorithms are chosen as follows:

(1) Overall nondominated vector generation (ONVG): number of vectors in Pareto front [47], and a larger ONVG value indicates more superior performance (this metric is actually applied to count the quantity of approximate Pareto solutions obtained by the correlation experimental algorithms);

(2) Convergence metric (CM): an indicator to evaluate convergence. It measures the dominant relationship between the two Pareto fronts E_1 and E_2. $C(E_1, E_2)$ reflects the percentage of solutions in E_2 dominated by E_1 or equal to E_1. A larger $C(E_1, E_2)$ value indicates better convergence of Pareto fronts E_1. The calculation formula is as follows [48]:

$$C(E_1, E_2) = \frac{\left|\{b \in E_2 | \exists a \in E_1, a \succ b \text{ or } a = b\}\right|}{\left|E_2\right|}. \quad (21)$$

(3) Spacing metric (SM): an indicator to evaluate the uniformity of the solution set distribution. The smaller the SM value, the more uniform the distribution. The calculation formula is as follows:

$$SM = \sqrt{\frac{1}{|E|}\sum_{a \in E}\frac{\left(d_a - \overline{d}\right)^2}{\overline{d}}}, \quad (22)$$

where d_a is the Euclidean distance between solution a and the solution nearest to a, E is the Pareto front, and $\overline{d} = \sum_{a \in E} d_a/|E|$ is the average distance [49].

(4) Hypervolume (HV): the sum of the hypervolume of the hypercube composed of all nondominated solutions and nadir point. The convergence, distribution uniformity and universality of the algorithm results can be evaluated simultaneously [50].

4.1. Comparison of NSDE and Gurobi. In the third-party optimizer evaluation conducted by Decision Tree for Optimization Software, the world's most famous professional optimizer evaluation website, Gurobi shows faster optimization speed and accuracy, which has been proved to have obvious advantages over other large-scale optimizers. The Gurobi optimization solver can be used to solve multi-objective optimization problems, which is one of its greatest strengths. This section compares the performance of NSDE and Gurobi. In order to adapt to the solution of the proposed problem, Gurobi's solving process is designed as follows: since the objective of AGV number cannot be solved directly by Gurobi's function, the number of AGVs is fixed and attempted from small to large, and the proposed problem is transformed into a single-objective optimization problem. Then check the optimal solution and the feasibility of the corresponding path. Finally, form the Pareto solution set to output. The layout of the automated warehouse is set as follows: the number of columns of goods (excluding passageways) is fixed at 20, the number of rows of goods r is taken in the set {20, 30}, and the number of picking points $n = \{10, 20, 30, 40\}$. These eight combination experiments are carried out, and five groups of random data are tested for each combination. Because the solution time of Gurobi is too long, after recording the running time of each NSDE experiment, Gurobi runs the same time to stop. The relevant experimental results are shown in Table 8 which records the number of Pareto solutions and running time obtained by the two algorithms under different scales.

When $n = 10$ to 40, the number of Pareto solutions obtained by NSDE algorithm is significantly more than that of Gurobi, and when $n = 30$ and 40, Gurobi cannot even obtain a feasible solution to solve the presented problem. It takes some time in Gurobi to preprocess and on judge the conflict of the searched solutions. Therefore, Gurobi's solving capability is limited. For the scale of $n = 30$ and above, Gurobi cannot get the same number of solutions as NSDE or even a feasible solution in the same time, but NSDE still has good performance.

Table 8: Comparison of experimental results between NSDE and Gurobi.

		$r = 20$			$r = 30$		
		NSDE	Gurobi	Time (s)	NSDE	Gurobi	Time (s)
$n = 10$	Trail 1	8	2	0.956	7	2	1.265
	Trail 2	7	2	0.951	6	2	1.15
	Trail 3	8	2	1.006	6	3	1.166
	Trail 4	7	2	0.973	7	3	1.393
	Trail 5	7	2	0.956	6	2	1.266
	Average	7.4	2	0.968	6.4	2.4	1.248
$n = 20$	Trail 1	10	1	1.562	10	1	1.686
	Trail 2	10	1	1.126	10	1	1.725
	Trail 3	10	1	1.505	10	1	1.825
	Trail 4	10	1	1.304	10	1	1.408
	Trail 5	10	1	1.379	10	1	1.908
	Average	10	1	1.375	10	1	1.710
$n = 30$	Trail 1	10	0	1.843	10	0	1.884
	Trail 2	10	0	1.682	10	0	2.102
	Trail 3	10	0	1.726	10	0	2.012
	Trail 4	10	0	1.599	10	0	1.826
	Trail 5	10	0	1.594	10	0	2.067
	Average	10	0	1.689	10	0	1.978
$n = 40$	Trail 1	10	0	2.152	10	0	2.2
	Trail 2	10	0	2.112	10	0	2.473
	Trail 3	10	0	2.083	10	0	2.465
	Trail 4	10	0	2.095	10	0	2.237
	Trail 5	10	0	2.109	10	0	2.049
	Average	10	0	2.110	10	0	2.285

4.2. Comparison of NSDE and MOTS. This section modifies the novel tabu search algorithm proposed by Xing et al. [17], so that it can solve the multi-objective problem, which can be used as a comparison algorithm of NSDE to evaluate the performance. The modified NTS algorithm, i.e., a MOTS algorithm, introducing the judgment of dominant relationship. In the comparative experiment between NSDE and MOTS, the layout of the automated warehouse is set as follows: the number of columns of goods (excluding passageways) is fixed at 20 columns, and the row number of goods r is taken in the set {20, 30}. Eight combination experiments with the number of pickup points $n = \{60, 120, 140, 180\}$ are carried out, 10 groups of random data are tested for each combination, and the relevant experimental results are recorded. Four evaluation indexes are used for performance comparison, including ONVG, CM, SM, and HV. Therefore, in the selection of HV reference point, the point when both objectives take the minimum value is selected and smaller HV value indicate the superiority of the algorithm. Because MOTS is time-consuming, after recording the running time of each NSDE experiment, MOTS runs the same time to stop. The relevant experimental results are shown in Tables 9–12 and Figures 6–9. These tables, respectively, record ONVG, CM, SM, HV, and running time obtained by the two algorithms under different scales.

Table 9: Comparison of experimental results between NSDE and MOTS ($n = 60$).

r		ONVG		CM		SM		HV		Time (s)
		NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	
20	Trail 1	10	10	0.60	0.70	7.51	7.72	274	266	2.56
	Trail 2	10	10	0.70	0.70	7.45	8.41	285	284	2.56
	Trail 3	10	10	0.40	0.90	6.52	9.78	285	268	2.73
	Trail 4	10	10	0.50	0.80	7.60	6.89	290	298	2.71
	Trail 5	10	10	0.70	0.90	7.54	8.66	276	263	2.79
	Trail 6	10	10	0.60	0.70	7.68	7.60	289	289	2.87
	Trail 7	10	10	0.60	0.60	7.82	7.78	292	284	2.54
	Trail 8	10	10	0.60	0.60	6.86	7.53	269	277	2.85
	Trail 9	10	10	0.80	0.40	7.48	8.23	270	275	2.78
	Trail 10	10	10	0.40	0.80	7.66	7.94	288	277	2.70
	Average	10	10	0.59	0.71	7.41	8.05	281.8	278.1	2.71
30	Trail 1	10	10	0.80	0.50	8.46	9.07	404	409	3.10
	Trail 2	10	10	0.90	0.50	9.58	9.66	398	406	2.96
	Trail 3	10	10	0.70	0.50	9.19	8.16	414	445	3.05
	Trail 4	10	10	0.50	0.80	8.91	11.43	418	400	2.85
	Trail 5	10	10	0.80	0.40	7.98	7.08	389	490	2.99
	Trail 6	10	10	0.50	0.70	9.21	11.54	447	425	3.12
	Trail 7	10	10	0.70	0.50	9.03	9.37	405	413	2.91
	Trail 8	10	10	0.80	0.40	7.96	9.89	400	405	2.76
	Trail 9	10	10	0.80	0.40	8.37	7.51	418	497	2.83
	Trail 10	10	10	0.70	0.60	9.24	9.49	413	409	3.03
	Average	10	10	0.72	0.53	8.79	9.32	410.6	429.9	2.96

Table 10: Comparison of experimental results between NSDE and MOTS ($n = 100$).

r		ONVG		CM		SM		HV		Time (s)
		NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	
20	Trail 1	10	10	0.60	0.70	7.56	8.38	293	288	4.62
	Trail 2	10	10	0.90	0.40	7.48	9.79	287	309	4.18
	Trail 3	10	10	0.70	0.70	7.59	8.66	283	287	4.20
	Trail 4	10	10	0.70	0.70	7.54	8.78	289	294	4.59
	Trail 5	10	10	0.70	0.60	7.75	9.23	299	304	4.53
	Trail 6	10	10	0.80	0.50	7.86	9.59	282	296	4.26
	Trail 7	10	10	0.80	0.40	7.11	7.37	280	324	4.10
	Trail 8	10	10	0.80	0.50	7.64	9.23	294	301	4.08
	Trail 9	10	10	0.70	0.60	7.67	8.20	298	300	4.46
	Trail 10	10	10	0.50	0.70	7.37	7.70	295	286	4.27
	Average	10	10	0.72	0.58	7.56	8.69	290	298.9	4.33
30	Trail 1	10	10	0.90	0.30	9.17	10.74	431	457	4.81
	Trail 2	10	10	0.50	0.70	9.46	10.79	434	429	4.87
	Trail 3	10	10	0.70	0.60	9.09	10.23	443	436	4.85
	Trail 4	10	4	1.00	0.00	9.53	6.67	436	308	4.85
	Trail 5	10	10	0.60	0.60	9.65	10.41	438	425	5.07
	Trail 6	10	10	0.60	0.50	9.01	8.13	432	471	4.84
	Trail 7	10	10	0.70	0.60	9.55	9.36	439	480	4.81
	Trail 8	10	10	0.70	0.50	9.50	8.18	449	488	5.08
	Trail 9	10	10	0.80	0.60	9.30	10.54	424	435	4.66
	Trail 10	10	10	0.70	0.50	9.41	9.13	459	467	4.98
	Average	10	9.4	0.72	0.49	9.37	9.42	438.5	439.6	4.88

From the Pareto solution number finally obtained, all experimental results meet the relation ONVG(NSDE) ≥ ONVG(MOTS), which means that more Pareto solutions can be obtained by NSDE. There is only one experiment that MOTS cannot obtain the same number of Pareto solutions as NSDE. This is because NSDE algorithm is based on population evolution, the distribution of solutions is wider, and the performance is more stable. From the convergence indicator CM, column NSDE and column MOTS, respectively, represent their dominated degree over the other algorithm, i.e., C(NSDE, MOTS) and C(MOTS, NSDE). There are $61/80 = 76.3\%$ of the cases

Table 11: Comparison of experimental results between NSDE and MOTS ($n = 140$).

r		ONVG		CM		SM		HV		Time (s)
		NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	
20	Trail 1	10	10	0.90	0.50	7.68	9.39	294	298	6.03
	Trail 2	10	10	0.60	0.70	7.72	8.53	308	308	5.73
	Trail 3	10	10	0.90	0.40	7.61	8.58	296	302	6.17
	Trail 4	10	10	0.70	0.50	7.53	7.93	293	295	6.26
	Trail 5	10	10	0.60	0.60	7.71	7.39	290	315	6.55
	Trail 6	10	10	0.80	0.70	7.60	8.25	293	291	6.57
	Trail 7	10	10	0.70	0.80	7.52	8.28	296	293	6.30
	Trail 8	10	10	0.80	0.70	7.70	7.60	305	329	6.74
	Trail 9	10	10	0.80	0.50	7.37	8.03	300	306	6.09
	Trail 10	10	10	0.60	0.80	7.66	7.69	292	283	6.63
	Average	10	10	0.74	0.62	7.61	8.17	296.7	302	6.31
30	Trail 1	10	10	0.70	0.50	9.28	9.19	448	497	6.65
	Trail 2	10	10	0.80	0.40	9.43	10.81	454	459	6.66
	Trail 3	10	10	0.70	0.50	9.65	9.55	449	480	6.67
	Trail 4	10	10	0.60	0.50	9.20	9.33	443	468	6.99
	Trail 5	10	10	0.50	0.80	9.53	10.52	450	445	7.43
	Trail 6	10	10	0.80	0.70	9.38	9.69	439	436	6.95
	Trail 7	10	10	0.50	0.80	9.30	9.93	460	440	7.06
	Trail 8	10	10	0.90	0.20	9.63	10.73	447	486	7.05
	Trail 9	10	10	0.90	0.40	9.08	8.10	446	511	6.66
	Trail 10	10	10	0.80	0.50	9.44	10.18	447	463	6.76
	Average	10	10	0.72	0.53	9.39	9.80	448.3	468.5	6.89

Table 12: Comparison of experimental results between NSDE and MOTS ($n = 180$).

r		ONVG		CM		SM		HV		Time (s)
		NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	NSDE	MOTS	
20	Trail 1	10	10	0.70	0.50	7.80	8.31	303	309	8.77
	Trail 2	10	10	0.90	0.50	7.89	7.93	298	302	7.61
	Trail 3	10	10	0.80	0.50	7.59	6.55	305	348	8.95
	Trail 4	10	10	0.70	0.70	7.89	8.41	301	312	8.86
	Trail 5	10	10	0.90	0.40	7.70	7.10	296	338	8.02
	Trail 6	10	10	1.00	0.80	7.77	6.34	300	336	9.95
	Trail 7	10	10	0.60	0.80	7.71	8.51	298	297	8.50
	Trail 8	10	10	0.80	0.60	7.71	5.39	298	360	9.16
	Trail 9	10	10	0.60	0.70	7.75	7.03	303	315	9.77
	Trail 10	10	10	0.80	0.50	7.74	7.75	298	320	9.45
	Average	10	10	0.78	0.60	7.75	7.33	300.0	323.7	8.90
30	Trail 1	10	10	0.70	0.60	9.64	8.21	447	511	9.54
	Trail 2	10	10	0.80	0.60	9.73	7.39	454	530	11.27
	Trail 3	10	10	0.70	0.50	9.63	9.88	449	451	10.46
	Trail 4	10	10	0.80	0.40	9.47	10.51	443	463	9.94
	Trail 5	10	10	0.80	0.70	9.71	11.07	454	454	9.60
	Trail 6	10	10	0.67	0.30	9.56	9.70	454	419	9.76
	Trail 7	10	10	1.00	0.40	9.59	11.64	444	491	10.67
	Trail 8	10	10	0.50	0.50	9.78	11.30	462	429	10.42
	Trail 9	10	10	0.70	0.70	9.58	9.89	457	453	11.92
	Trail 10	10	10	0.60	0.70	9.55	9.04	439	496	9.57
	Average	10	10	0.73	0.54	9.62	9.86	450.3	469.7	10.31

when C(NSDE, MOTS) ≥ C(MOTS, NSDE), so the convergence of the NSDE algorithm is better and more stable than MOTS. From the distribution uniformity indicator SM, there are 57/80 = 71.3% of cases when NSDE < MOTS, so solutions of NSDE have better distribution uniformity than MOTS. From the perspective of the comprehensive evaluation index HV, i.e., a comprehensive measure of convergence and distribution uniformity and universality of the algorithm, the case of NSDE ≤ MOTS is 23/80 = 68.8%, indicating that the overall performance of NSDE is better. The novel encoding and decoding method can eliminate the impact of duplicate paths to generate

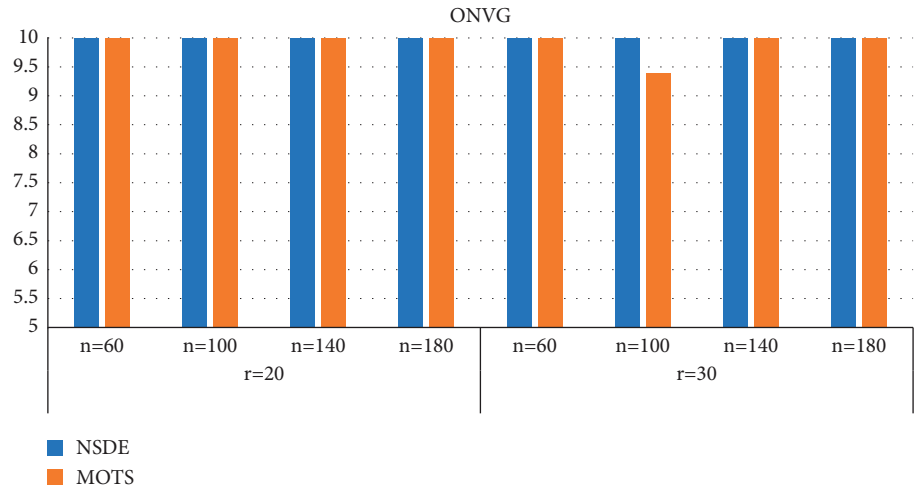

Figure 6: Comparison of average ONVG between NSDE and MOTS.

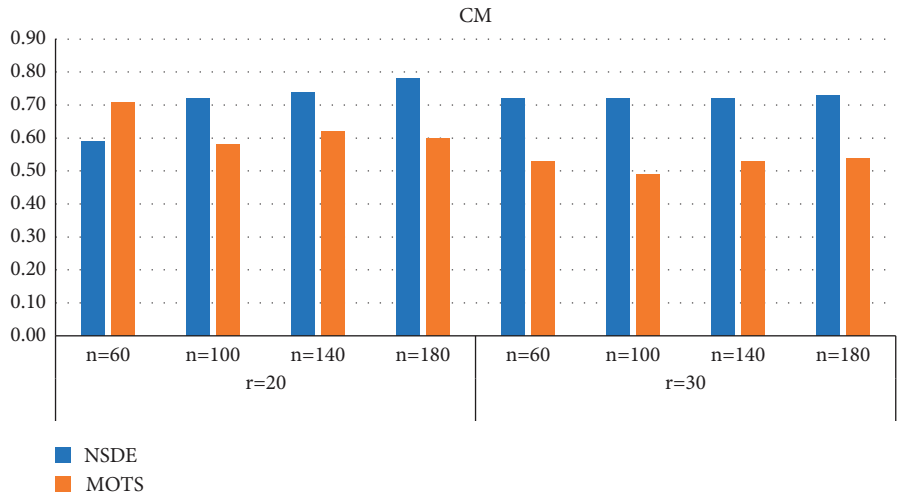

Figure 7: Comparison of average CM between NSDE and MOTS.

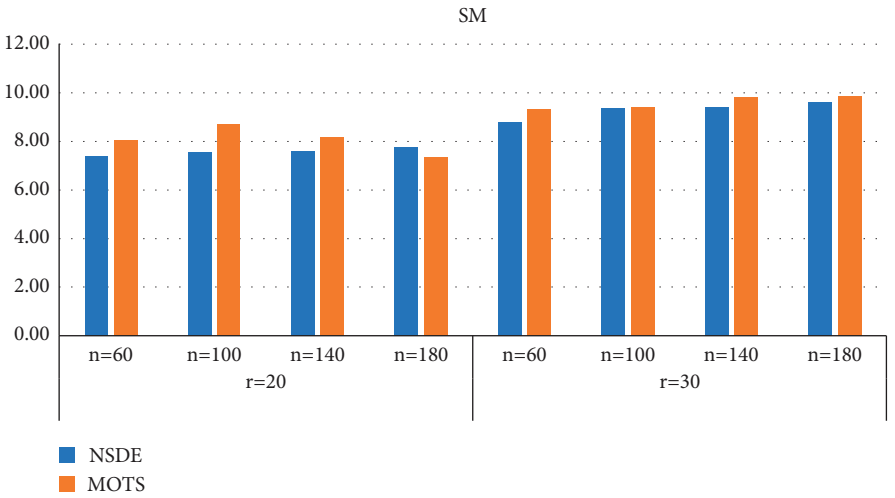

Figure 8: Comparison of average SM between NSDE and MOTS.

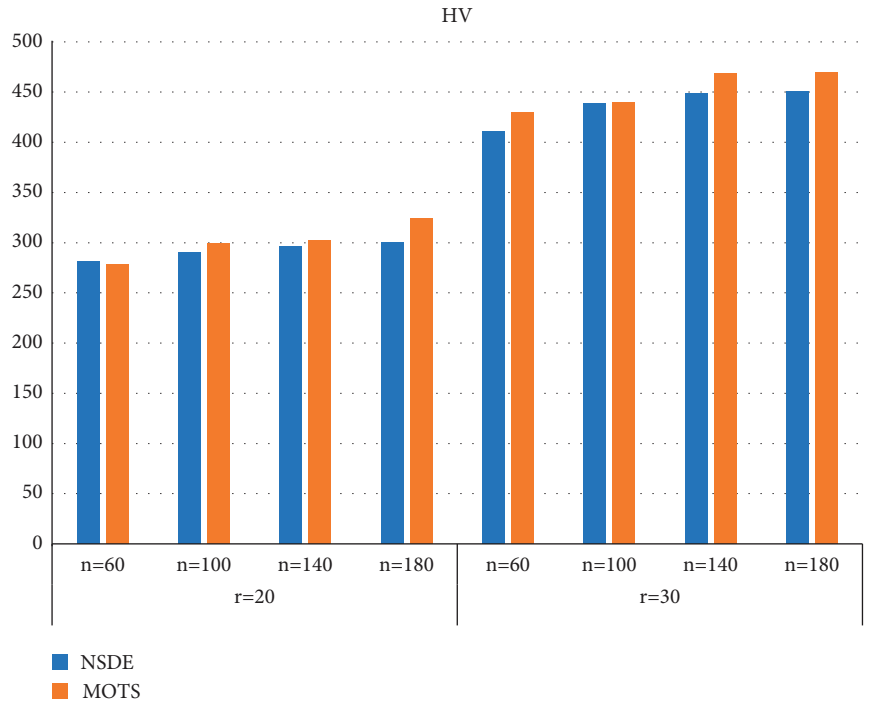

Figure 9: Comparison of average HV between NSDE and MOTS.

Table 13: Orthogonal factor level table of the NSDE algorithm.

Factors	Mutation coefficient	Crossover coefficient	Population size	Maximum iteration
Level 1	0.1	0.1	50	100
Level 2	0.3	0.3	100	200
Level 3	0.5	0.5	150	300
Level 4	0.7	0.7	200	400
Level 5	0.9	0.9	250	500

Table 14: Orthogonal experimental design table of the NSDE algorithm.

Factors	Mutation coefficient	Crossover coefficient	Maximum iterations	Experimental result
Trail 1	0.1	0.1	50	100
Trail 2	0.1	0.3	100	200
Trail 3	0.1	0.5	150	300
Trail 4	0.1	0.7	200	400
Trail 5	0.1	0.9	250	500
Trail 6	0.3	0.1	100	300
Trail 7	0.3	0.3	150	400
Trail 8	0.3	0.5	200	500
Trail 9	0.3	0.7	250	100
Trail 10	0.3	0.9	50	200
Trail 11	0.5	0.1	150	500
Trail 12	0.5	0.3	200	100
Trail 13	0.5	0.5	250	200
Trail 14	0.5	0.7	50	300
Trail 15	0.5	0.9	100	400
Trail 16	0.7	0.1	200	200
Trail 17	0.7	0.3	250	300
Trail 18	0.7	0.5	50	400
Trail 19	0.7	0.7	100	500
Trail 20	0.7	0.9	150	100
Trail 21	0.9	0.1	250	400
Trail 22	0.9	0.3	50	500
Trail 23	0.9	0.5	100	100
Trail 24	0.9	0.7	150	200
Trail 25	0.9	0.9	200	300

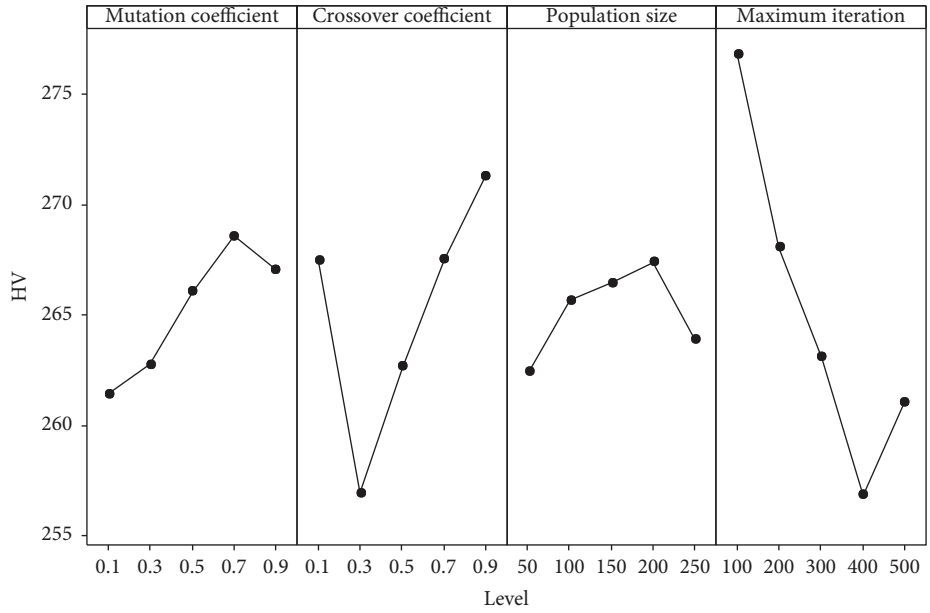

Figure 10: The main effect graph of mean for the NSDE algorithm.

better Pareto solutions, avoid too many inferior individuals in the population, and then greatly reduce the calculation time. According to the average level of the four metrics in Figures 6–9, NSDE is superior to MOTS in most cases. In general, NSDE performed well in all aspects, proving the advantages of NSDE's population-based evolution mechanism and elite selection strategy.

5. Conclusions

This paper studies the biobjective multi-AGV routing problem in an automatic warehouse. In order to make use of the greater advantages of a population-based intelligent optimization algorithm in solving multi-objective optimization problems, a NSDE algorithm is designed. Firstly, a novel encoding and decoding method is designed to eliminate the influence of repeated path, prevent too many bad solutions, and save time. Then the fast nondominated sorting method and elite selection strategy of NSGA-II are introduced to improve the selection operation of differential evolution algorithm, so that better individuals in the offspring population can be retained to continuously update the Pareto solution set. Finally, the effectiveness of the algorithm is proved by numerical experiment.

Some limitations of the study include as follows: When the proposed NSDE algorithm runs at a late stage, the population may be updated slowly sometimes. In this case, future studies will consider the design of a population restart strategy to further improve computing performance. Due to the variability of practical problems, it will be continued to study the following aspects in the future: for the multi-AGV path optimization problem, the limited capacity of AGV when the quality or volume of goods is large, and solution method of the online state will be considered. In these cases, the types of conflict may be more complex.

Appendix

A. Parameters setting for the NSDE algorithm

The four factors that influence the performance of the NSDE algorithm were tested. The levels required to test for each factor are presented in Table 13, and the orthogonal experimental design table is shown in Table 14. Each trail included 10 random tests when $n = 60$, and the average HV metric value of 10 random tests was recorded at column experimental result. According to the valley point of the main effect graph in Figure 10, the level of each factor is determined as follows: mutation coefficient: 0.1, crossover coefficient: 0.3, population size: 50, and maximum number of iterations: 400.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was funded by the Korea-China Young Researchers Exchange Program (2020) (30,000,000KRW); the Science Foundation of Shenyang University of Chemical Technology under Grants no. LJ2020032 (30,000RMB) and NO. LQ2020020 (30,000RMB); and 2021 Natural Science Foundation of Liaoning Province under Grant NO. 2021-MS-259 (50,000RMB).

References

[1] T. Nishi, Y. Hiranaka, and I. E. Grossmann, "A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles," *Computers & Operations Research*, vol. 38, pp. 876–888, 2011.

[2] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, pp. 677–709, 2006.

[3] T. Le-Anh and M. B. M. De Koster, "A review of design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 171, no. 1, pp. 1–23, 2006.

[4] Q. S. Kabir and Y. Suzuki, "Comparative analysis of different routing heuristics for the battery management of automated guided vehicles," *International Journal of Production Research*, vol. 57, pp. 624–641, 2019.

[5] Z. Zhang, L. Wu, W. Zhang, T. Peng, and J. Zheng, "Energy-efficient path planning for a single-load automated guided vehicle in a manufacturing workshop," *Computers & Industrial Engineering*, vol. 158, Article ID 107397, 2021.

[6] W.-Q. Zou, Q.-K. Pan, and M. F. Tasgetiren, "An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop," *Applied Soft Computing*, vol. 99, Article ID 106945, 2021.

[7] W.-Q. Zou, Q.-K. Pan, and L. Wang, "An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery," *Knowledge-Based Systems*, vol. 218, Article ID 106881, 2021.

[8] J. Zając and W. Małopolski, "Structural on-line control policy for collision and deadlock resolution in multi-AGV systems," *Journal of Manufacturing Systems*, vol. 60, pp. 80–92, 2021.

[9] A. Goli, E. B. Tirkolaee, and N. S. Aydın, "Fuzzy integrated cell formation and production scheduling considering automated guided vehicles and human factors," *IEEE Transactions on Fuzzy Systems*, vol. 29, pp. 3686–3695, 2021.

[10] M. Drótos, P. Györgyi, M. Horváth, and T. Kis, "Suboptimal and conflict-free control of a fleet of AGVs to serve online requests," *Computers & Industrial Engineering*, vol. 152, Article ID 106999, 2021.

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

[12] I. Aziez, J.-F. Côté, and L. C. Coelho, "Fleet sizing and routing of healthcare automated guided vehicles," *Transportation Research Part E: Logistics and Transportation Review*, vol. 161, Article ID 102679, 2022.

[13] Y. Yang, M. Zhong, Y. Dessouky, and O. Postolache, "An integrated scheduling method for AGV routing in automated container terminals," *Computers & Industrial Engineering*, vol. 126, pp. 482–493, 2018.

[14] M. Zhong, Y. Yang, Y. Dessouky, and O. Postolache, "Multi-AGV scheduling for conflict-free path planning in automated container terminals," *Computers & Industrial Engineering*, vol. 142, Article ID 106371, 2020.

[15] S. Ji, D. Luan, Z. Chen, and D. Guo, "Integrated scheduling in automated container terminals considering AGV conflict-free routing," *Transportation Letters*, vol. 13, no. 7, pp. 501–513, 2021.

[16] P. Zhou, L. Lin, and K. H. Kim, "Anisotropic Q-learning and waiting estimation based real-time routing for automated guided vehicles at container terminals," *Journal of Heuristics*, 2021.

[17] L. Xing, Y. Liu, H. Li, C. C. Wu, W. C. Lin, and X. Chen, "A novel tabu search algorithm for multi-AGV routing problem," *Mathematics*, vol. 8, no. 2, Article ID 279, 2020.

[18] K. J. C. Fransen, J. A. W. M. van Eekelen, A. Pogromsky, M. A. A. Boon, and I. J. B. F. Adan, "A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems," *Computers & Operations Research*, vol. 123, Article ID 105046, 2020.

[19] C. W. Lee, W. P. Wong, J. Ignatius, A. Rahman, and M.-L. Tseng, "Winner determination problem in multiple automated guided vehicle considering cost and flexibility," *Computers & Industrial Engineering*, vol. 142, Article ID 106337, 2020.

[20] J. Chen, X. Zhang, X. Peng, D. Xu, and J. Peng, "Efficient routing for multi-AGV based on optimized Ant-agent," *Computers & Industrial Engineering*, vol. 167, Article ID 108042, 2022.

[21] F. Xu, H. Li, C. M. Pun et al., "A new global best guided artificial bee colony algorithm with application in robot path planning," *Applied Soft Computing*, vol. 88, Article ID 106037, 2020.

[22] H. Gao, Z. Fu, C.-M. Pun, J. Zhang, and S. Kwong, "An efficient artificial bee colony algorithm with an improved linkage identification method," *IEEE Transactions on Cybernetics*, vol. 52, pp. 4400–4414, 2022.

[23] M. Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential Evolution: a review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, Article ID 103479, 2020.

[24] F. Zhao, L. Zhao, L. Wang, and H. Song, "An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion," *Expert Systems with Applications*, vol. 160, Article ID 113678, 2020.

[25] X. Yu, C. Li, and J. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowledge-Based Systems*, vol. 204, Article ID 106209, 2020.

[26] I. Ahmadianfar, A. Kheyrandish, M. Jamei, and B. Gharabaghi, "Optimizing operating rules for multi-reservoir hydropower generation systems: an adaptive hybrid differential evolution algorithm," *Renewable Energy*, vol. 167, pp. 774–790, 2021.

[27] V. Stanovov, S. Akhmedova, and E. Semenkin, "Biased parameter adaptation in differential evolution," *Information Sciences*, vol. 566, pp. 215–238, 2021.

[28] S. Gupta and R. Su, "An efficient differential evolution with fitness-based dynamic mutation strategy and control parameters," *Knowledge-Based Systems*, vol. 251, Article ID 109280, 2022.

[29] I. M. Ali, D. Essam, and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," *Swarm and Evolutionary Computation*, vol. 52, Article ID 100607, 2020.

[30] M. Wang, Y. Ma, and P. Wang, "Parameter and strategy adaptive differential evolution algorithm based on accompanying evolution," *Information Sciences*, vol. 607, pp. 1136–1157, 2022.

[31] Z. Tan, K. Li, and Y. Wang, "Differential evolution with adaptive mutation strategy based on fitness landscape analysis," *Information Sciences*, vol. 549, pp. 142–163, 2021.

[32] W. Deng, J. Xu, Y. Song, and H. Zhao, "Differential evolution algorithm with wavelet basis function and optimal mutation

strategy for complex optimization problem," *Applied Soft Computing*, vol. 100, Article ID 106724, 2021.

[33] S. Li, Q. Gu, W. Gong, and B. Ning, "An enhanced adaptive differential evolution algorithm for parameter extraction of photovoltaic models," *Energy Conversion and Management*, vol. 205, Article ID 112443, 2020.

[34] Z. Zeng, M. Zhang, T. Chen, and Z. Hong, "A new selection operator for differential evolution algorithm," *Knowledge-Based Systems*, vol. 226, Article ID 107150, 2021.

[35] Y. Song, D. Wu, W. Deng et al., "MPPCEDE: multi-population parallel co-evolutionary differential evolution for parameter optimization," *Energy Conversion and Management*, vol. 228, Article ID 113661, 2021.

[36] W. Deng, S. Shang, X. Cai et al., "Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization," *Knowledge-Based Systems*, vol. 224, Article ID 107080, 2021.

[37] J. Sun, X. Liu, T. Bäck, and Z. Xu, "Learning adaptive differential evolution algorithm from optimization experiences by policy gradient," *IEEE Transactions on Evolutionary Computation*, vol. 25, pp. 666–680, 2021.

[38] E. H. Houssein, M. A. Mahdy, M. J. Blondin, D. Shebl, and W. M. Mohamed, "Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems," *Expert Systems with Applications*, vol. 174, Article ID 114689, 2021.

[39] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robotics and Autonomous Systems*, vol. 59, pp. 801–812, 2011.

[40] X. Lei and Z. Shi, "Overview of multi-objective optimization methods," *Journal of Systems Engineering and Electronics*, vol. 15, no. 2, pp. 142–146, 2004.

[41] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[42] L.-N. Xing, Y. Liu, H. Li, C. C. Wu, W. C. Lin, and W. Song, "A hybrid discrete differential evolution algorithm to solve the split delivery vehicle routing problem," *IEEE Access*, vol. 8, pp. 207962–207972, 2020.

[43] D. Bai, J. Liang, B. Liu, M. Tang, and Z.-H. Zhang, "Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates," *Computers & Industrial Engineering*, vol. 112, pp. 336–347, 2017.

[44] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[45] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 44, pp. 2391–2404, 2014.

[46] J. Ding, C. Yang, Q. Xiao, T. Chai, and Y. Jin, "Dynamic evolutionary multiobjective optimization for raw ore allocation in mineral processing," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 36–48, 2019.

[47] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the 2000 Congress on Evolutionary Computation*, pp. 204–211, La Jolla, USA, July 2000.

[48] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: nsga-II," in *Parallel Problem Solving from Nature PPSN VI*vol. 1917pp. 849–858, Berlin/Heidelberg, Germany, 2000.

[49] J. R. Schott, *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*, Doctoral dissertation, Massachusetts Institute of Technology, Cambridge(MA), 1995.

[50] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 257–271, 1999.