IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 February 2024, accepted 8 May 2024, date of publication 15 May 2024, date of current version 18 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3401487

RESEARCH ARTICLE

Self-Competition Particle Swarm Optimization Algorithm for the Vehicle Routing Problem With Time Window

YUFENG WANG[1], XIN CHEN[1], ZHUO SHUANG[1], YING ZHAN[1], KE CHEN[1], AND CHUNYU XU[2,3]

[1]School of Software and Computer Science, Nanyang Institute of Technology, Nanyang 473000, China
[2]Electronic Information School, Wuhan University, Wuhan 430072, China
[3]School of Information Engineering, Nanyang Institute of Technology, Nanyang 473000, China

Corresponding author: Xin Chen (2115925372@nyist.edu.cn)

This work was supported in part by the Key Research Projects of Henan Science and Technology Department under Grant 232102310427 and Grant 232102211058, in part by the Research and Practice Project of Research Teaching Reform in Henan Undergraduate University under Grant 2022SYJXLX114, in part by the Key Research Programs of Higher Education Institutions in Henan Province under Grant 24B520026, in part by the Special Research Project for the Construction of Provincial Demonstration Schools at Nanyang University of Technology under Grant SFX202314, and in part by the Research on Education and Teaching Reform Program of Nanyang Institute of Technology (NYIST) under Grant NIT2023JY-108.

ABSTRACT The vehicle routing problem with time windows (VRPTW) is a well-known NP-Hard combinatorial optimization problem, which is frequently encountered in transportation and logistics scenarios. When traditional algorithms solve this problem, there are some problems, such as slow convergence speed. In this paper, a Self-competitive Particle Swarm Optimization (ScPSO) algorithm is proposed, which regulates the learning direction of the next generation of particles based on their degree of self-competition. When the particle's self-competition degree is low, it is compelled to learn from the individual optimal solution using the self-competition selection probability, thereby achieving rapid convergence. ScPSO uses the nonlinear inertia weight adaptive strategy to adjust the search preference in the search process, and it can balance the relationship between exploration and exploitation. Meanwhile, Random greedy heuristic selection and variable neighborhood search strategies are used in initial solution construction and constraint restriction. Finally, ScPSO is tested on 56 Solomon 100-customers benchmark problems and compared with four state-of-the-art VRPTW algorithms and best-known solutions reported on Solomon's webpage.The running results of the ScPSO algorithm are better than four state-of-the-art VRPTW algorithms and best-known solutions reported on Solomon's webpage. The experimental results show that ScPSO can solve VRPTW problems efficiently.

INDEX TERMS Particle swarm optimization, self-competitive learning strategy, random greedy heuristic selection strategy, variable neighborhood search strategy.

I. INTRODUCTION

Vehicle Routing Problem with Time Windows (VRPTW) is an essential problem in transportation and logistics scenarios. This problem can be formulated as designing a minimal route for a fleet of vehicles to deliver goods to customers. Each vehicle departs from and eventually returns to the warehouse, and each customer can only be visited once by one vehicle. At the same time, the constraints of time window and capacity should be satisfied in the process of vehicle transportation. In addition, there are two objectives to deal with the VRPTW problem: the first is to minimize the total distance of vehicle travel (TD), and the second is to minimize the number of vehicle paths (NV) as much as possible. VRPTW problem is widely used in real life. Goodarzian et al. [1] proposed the first research on optimizing the service time and cost of home health care (HHC) logistics through route balancing through the study of the VRPTW

The associate editor coordinating the review of this manuscript and approving it for publication was Joanna Kołodziej.

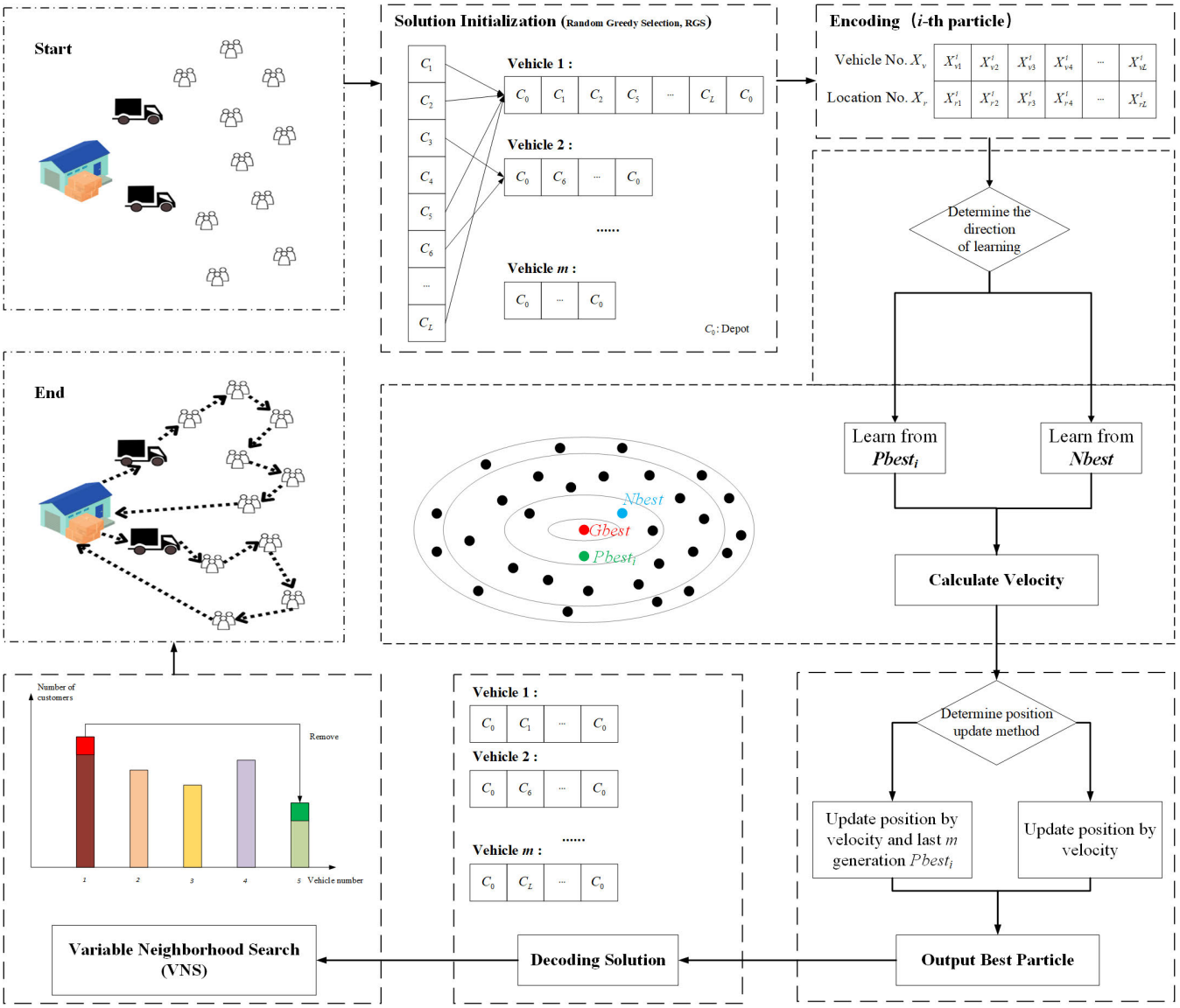


FIGURE 1. The framework of ScPSO.

problem. Kuznietsov et al. [2] proposed A solution to the logistics problem of food and consumer goods distributors in Ukraine through their study of the VRPTW problem. Tao et al. [3] established a mathematical model for vehicle routing optimization of cold chain logistics distribution with the minimum comprehensive cost considering the increasing difficulty of route optimization and the increase of carbon emissions in the process of cold chain logistics distribution. Therefore, the VRPTW problem is closely related to life, and it is of great significance to the research of the VRPTW problem.

Many researchers have made great efforts to solve the VRPTW minimum path problem, and have proposed exact and approximate solutions. There are three main strategies to solve the VRPTW problem in the exact solution algorithm: column generation, dynamic programming, and Lagrangian relaxation. Desrosiers et al. [4] developed an exact method using column generation, a technique used to solve large-scale combinatorial optimization problems by decomposing the problem into a primary problem and a subproblem. Kolen et al. [5] proposed a branch-and-bound dynamic programming method to minimize the total distance traveled by vehicles. This method decomposed the VRPTW problem into a series of sub-problems, which were solved gradually using dynamic programming. Kohl and Madsen [6] transformed the VRPTW problem into a sequence of subproblems that could be decomposed and solved by Lagrangian relaxation and obtained the optimal solution step by step by iterative optimization. This method is effective and feasible in solving VRPTW. Kohl et al. [7] proposed a branch-and-bound approach to minimize the distance of each path, which efficiently searches for the

optimal solution by decomposing the problem into a series of subproblems and using upper and lower bounds to limit the search space. The precise solution algorithm performs effectively in solving small-scale examples, but its efficiency diminishes when tackling large-scale examples. To solve this problem, more and more scholars have begun to use the approximate solution (heuristic algorithm) method to solve the VRPTW problem.

Due to the continuous application of evolutionary computation (EC) in daily life, researchers have undertaken extensive research efforts to develop novel evolutionary algorithms (EAs). Houssein et al. [8] proposed a new bionic optimization algorithm called the Hepatoma algorithm (LCA), which effectively balances local search and global search to explore the search space. Li et al. [9] introduced a new stochastic optimizer known as the Slime Mold Algorithm (SMA), which utilizes adaptive weights to simulate the positive and negative feedback process of slime mold propagation waves based on biological oscillators, thereby forming an optimal path connecting food with excellent exploration ability and utilization propensity. Mohamed et al. [10] drew inspiration from moths facing toward moonlight, leading them to propose a new Moth Swarm Algorithm (MSA) for solving constrained Optimal Power Flow (OPF) problems. The effectiveness and superiority of MSA are verified through experimental comparisons. Yang et al. [11] presented Hunger Games Search (HGS), a general population-based optimization technique characterized by its simple structure, exceptional stability, and highly competitive performance in efficiently solving both constrained and unconstrained problems. Tu et al. [12] inspired by collective predation behavior observed in nature, put forward a novel stochastic optimization algorithm named group predation algorithm that outperforms other algorithms across various search scenarios according to experimental results analysis. Ahmadianfar et al. [13] introduced Vector Weighted Mean (INFO), an innovative optimizer that optimizes different problems by improving update rules as well as vector merging steps within INFO to enhance exploration capability along with the exploitation ability of the algorithm. Heidari et al. [14] proposed a novel population-based nature-inspired optimization algorithm known as the Harris Hawks Optimizer (HHO). This study mathematically models the dynamic patterns and behaviors of Harris eagles to develop an efficient optimization algorithm. Su et al. [15] introduced an effective optimization algorithm called RIME, which is based on the physical phenomenon of fog and ice. By simulating the growth process of soft frost and hard frost in RIME-ICE, this algorithm constructs a search strategy for soft frost and a puncture mechanism for hard frost to achieve both exploration and exploitation behavior in the optimization method. Ahmadianfar et al. [16] proposed a metaphor-free swarm optimization algorithm, the RUNge Kutta optimizer (RUN), which is inspired by the well-known Runge Kutta (RK) method in mathematics. The slope change logic computed by the RK method is used as a promising logical search mechanism for global optimization. It shows superior ability to explore, achieve fast convergence, and avoid local optima.

Since Savelsbergh [17] proved that the VRPTW problem is an NP-Hard combinatorial optimization problem, the research on its algorithms has mainly focused on various heuristics. Traditional region search methods include Route Construction Heuristics, Route Improvement Heuristics, and Composite Heuristics. However, the best solution of traditional regional search methods is often limited by the characteristics of the initial solution or search methods' characteristics, and it can only obtain the local best solution. To improve this shortcoming, there have been significant developments in this field in recent years. Evolutionary algorithms (EC) are a new generation of heuristic solutions. Standard evolutionary algorithms used to solve VRPTW problems include the Tabu Search Algorithm [18], Simulated Annealing algorithm [19], Genetic Algorithm [20], Ant Colony algorithm Optimization [21], and Particle Swarm Optimization Algorithm [22], which can effectively solve the problem of falling into local optimum.

As the path planning problem has been widely used in real life, the related research on path planning has gradually increased, and more and more heuristic algorithms have been used to solve the path planning problem. Tabu search algorithm (TS) is a heuristic search algorithm using local search. It has been widely used because it prohibits repeating previous work and then jumps out of the local optimum. In order to avoid the circulation problem in the search process of the population, Taillard et al. [23] proposed a tabu search heuristic algorithm. The algorithm does not use the local optimum as the stopping criterion and uses the tabu table to realize the principle of only advancing and not retreating. Rochat and Taillard [24]further improved the tabu search algorithm, and compared the experimental results with the best-known solution reported in other previous works. The results show that the proposed algorithm has better solution quality and faster solution efficiency. Genetic Algorithm (GA) is a well-known population-based metaheuristic algorithm that optimizes the population by natural selection, crossover, mutation, and other operations. It has the advantages of wide search space and fast convergence speed, so it is widely used in solving VRPTW problems. Ursani et al. [25] demonstrated better performance in minimizing the number of vehicles or the total distance traveled by using an iterative process between two phases of optimization and de-optimization. Simulated Annealing (SA) gives the search process a time-varying probability jump that eventually tends to zero, which can effectively avoid falling into the local minimum and eventually tends to the global optimum and can be used to solve VRPTW problems. Land Lim [26] proposed a tabu embedded Simulated Annealing (TSA) metaheuristic algorithm, which uses the best k-restart strategy to enhance the local search when performing local search in the neighborhood under constraints, and achieves good

results in solving VRPTW problems. In recent years, the research on VRPTW has also increasing, and at the same time, more and more new algorithms have been applied to solve the path planning problem. Yang et al. [27] proposed an enhanced slime mold algorithm to address the path planning challenges in fire rescue operations involving unmanned equipment. By incorporating a mutation mechanism and dynamic weight coefficient, they successfully resolved the issues of slow convergence and low optimization accuracy encountered in the standard SMA algorithm. Zhang et al. [28] introduced an improved slime mold algorithm (SMA-CSA) to tackle global optimization and capacitated vehicle routing problems. Through the introduction of a Cauchy mutation strategy, they perturbed the optimal solution, thereby increasing the probability of escaping local minima. Additionally, they incorporated an annealing strategy to expand the global search space and enhance exploration during the quest for optimal solutions. Xiang et al. [29], on their part, proposed an enhanced chimpanzee optimization algorithm (MG-ChOA) which was applied to solve spherical VRPTW models effectively. Venkatasubramanian [30] employed a hybrid moth search algorithm for cluster head selection (CH). This approach combined moth search with a differential evolution method while utilizing the tree seed technique for establishing multipath routing (TSA). Alweshah et al. [31] utilized the Harris Hawk Optimization (HHO) algorithm due to its strong detection ability as well as its capability to find optimal paths when attempting to solve VRP problems. Wang et al. [32] proposed a multi-group comprehensive differential evolution algorithm based on contribution, which further improved the algorithm's performance by mutation strategy and grouping mechanism. This kind of algorithm provides a new idea for solving the VRPTW problem.

Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart in 1995. In recent years, the PSO algorithm has developed rapidly. Due to its advantages, such as simplicity and fast convergence, it has been widely applied in various fields, including function optimization, image processing, and geodesy. These advantages of the PSO algorithm have sparked the interest of researchers and encouraged their significant success in solving various NP-Hard problems, including the VRPTW problem. However, most existing particle swarm optimization algorithms operate in continuous spaces, where changes in its physical position define the particle's movement trajectory. In order to apply the PSO algorithm to solve the VRPTW problem, researchers have proposed several variant PSO algorithms in [33], [34], [35], [36], [37], [38], [39], and [40]. In [33], a two-stage VRPTW was designed to address interruption issues. In [34], a particle swarm optimization with local search was introduced. An improved PSO algorithm, Improved PSO (IPSO), was proposed in [35], showing better computation results and accuracy. A multi-adaptive particle swarm optimization (MAPSO) algorithm was presented in [36]. However, the methods proposed in [37], [38], [39], and [40] only solved VRPTW problems with a few customers and could not handle large-scale VRPTW problems. Furthermore, issues such as premature convergence or weak search capabilities persist.

In order to improve the convergence speed of the algorithm for solving VRPTW problems, this paper proposes an algorithm called Self-competitive Particle Swarm Optimization (ScPSO). ScPSO employs a particle self-competitive learning strategy to control particle exploration and exploitation, enabling particles to eliminate weaknesses and learn towards individual optima, thereby accelerating convergence speed; ScPSO uses an inertia weight adaptive strategy to balance the global and local search capabilities of particles, reaching precise search and minimizing the total distance traveled by vehicles. During the initial solutions deconstruction, a random greedy heuristic RGS (random greedy heuristic selection strategy) strategy is used to form a relatively good initial solution. VNS (Variable Neighborhood Search) strategy is used during the solution generation process to fine-tune the generated solution. In ScPSO, four contributions are proposed as follows:

1. A particle self-competitive learning strategy is proposed, which controls the exploration and exploitation of particles through their self-competitive degree (ScD). The ScD of particles is obtained by calculating the ratio of the fitness value of the current particle to the average fitness value of all particles in the contemporary population. When the ScD of the particle is less than 1, it indicates that the fitness value of the particle is better than the average fitness value of the current population, and the search direction of the particle should continue to be maintained. When the ScD of the particle is more significant than 1, it indicates that the fitness value of the particle is inferior to the average fitness value of the current population. When the particle updates its position, it is forced to learn from the individual's optimal solution based on the self-competitive selection probability.

2. A nonlinear inertia weight adaptive strategy is proposed, which balances particles' global search ability and local search ability by adaptively reducing inertia weight. The inertia weight adaptive strategy is achieved by calculating the ratio of the current particle's iteration count to the maximum iteration count. When the ratio is relatively small, expanding the scope and adopting a global search is advisable. As the number of iterations increases, particles gradually shift from global to local search, improving search accuracy.

3. A random greedy selection heuristic (RGS) strategy is proposed. It selects the most optimal customer for delivery by calculating the insertion cost. The RGS strategy calculates the insertion cost of customers to be delivered, selects the optimal customer to be delivered from the sorting results, and forms a good initial solution.

4. A variable neighborhood search (VNS) strategy is proposed. It can make local adjustments to the customer's delivery order, thereby adjusting the vehicle delivery order and total transportation cost. VNS is a domain search conducted on customers who still need to join the delivery path based on delivery costs. Based on the search results, reinsert

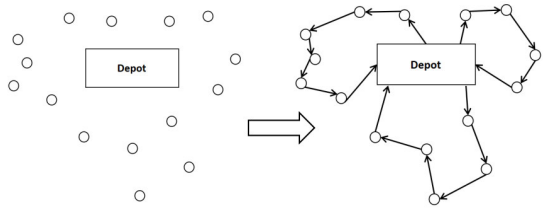


FIGURE 2. One solution for vehicle routing problem.

the delivery path according to the distance relationship between customers.

The structure of this paper is as follows: Section II describes some basic PSO and better-performing algorithms. Section III provides a detailed introduction to the Self-competitive Learning Particle Swarm Optimization (ScPSO) algorithm proposed in this paper. Section IV presents the experimental section, where the proposed algorithm is compared with other algorithms, and parameter effectiveness experiments are conducted. Section V summarizes the paper and gives future work.

II. RELATED WORK

A. VEHICLE ROUTING PROBLEM

Vehicle Routing Problem (VRP) is a crucial NP-Hard combinatorial optimization problem. VRP focuses on the case of a supplier and a path planning with K points of sale, which can be summarized as: for a set of shipping points and receiving points, the organization invokes certain vehicles, arranges appropriate driving routes, and makes the vehicles pass through them in an orderly manner, subject to specified constraints (e.g., Demand and shipment of goods, delivery time, vehicle capacity limit, mileage limit, travel time limit), and strive to achieve specific goals (such as the shortest total mileage of empty vehicles, the lowest total transportation cost, the arrival of vehicles according to a specific time, the minimum number of vehicles used). The Capacitated Vehicle Routing Problem (CVRP) is the basic version of the VRP and is a classical variant of the vehicle routing problem. In this type of problem, each node has a demand, each vehicle has a maximum load, and it is required that the sum of all node demands on the path assigned to each vehicle does not exceed the maximum load of the vehicle. Vehicle Routing Problem with Time Window (VRPTW) is an extension of VRP, produced by adding delivery time constraints to VRP. In VRPTW, nodes are associated with more attributes, and the solution must satisfy more constraints. In this type of problem, given the earliest and latest time of arrival of the vehicle at the destination, it is required that the vehicle must arrive within a specified time window, and an additional penalty fee is incurred either earlier than the earliest time or later than the latest time. The details of a solution for VRP are shown in Figure.2.

In VRPTW, we define $G = (V, E)$ as a directed complete graph, where $V = (c_0, c_1, \cdots, c_N)$ is the node set, and $E = \{\langle c_i, c_j\rangle | c_i, c_j \in V, c_i \neq c_j\}$ is the arc set. c_0 represents the depot, and c_i represents the i-th customer ($i = 1, 2, \cdots, N$). Each edge $\langle c_i, c_j\rangle$ is represented by the Euclidean distance d_{ij} between customer c_i and c_j ($d_{ij} = d_{ji}$). A is the vehicles set, $A = (a_0, a_1, \cdots, a_M)$, M is the maximum number of available vehicles. The time required for vehicles to serve each customer requires a time cycle. Therefore, when the vehicle arrives at the customer earlier than the service start time, it must wait until the service time window starts before providing service to the customer. On the other hand, if the vehicle cannot arrive before the end of the stop time, then the vehicle cannot serve this customer. At this point, the customer should be serviced by another vehicle.

In VRPTW, the definition of decision variables is as follows:

$$x_{ij}^k = \begin{cases} 1, & \textit{if vehicle } k \textit{ travels directly from } c_i \textit{ to } c_j \\ 0, & \textit{otherwise} \end{cases}$$

$$y_i^k = \begin{cases} 1, & \textit{if customer } c_i \textit{ is served by vehicle } k \\ 0, & \textit{otherwise} \end{cases}$$

where x_{ij}^k represents the k-th vehicle travels from customer c_i to customer c_j, y_i^k represents that customer c_i is served by k-th vehicle.

The primary objective is to minimize the total vehicle travel distance, the secondary objective is to minimize the number of vehicle routes NV as much as possible. The fitness evaluation function is as follows:

$$\min Z = C_0 \sum_{k=1}^{K} Q_k + C_1 \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} d_{ij} x_{ij}^k + \sum_{k=1}^{K} F_k \quad (1)$$

where C_0 represents the penalty cost incurred by violating the capacity constraint($C_0 = 10$), C_1 represents the vehicle's exercise cost per unit distance($C_1 = 1$), K is the number of vehicle routes, which also means that K vehicles to be used, $0 < K \leq M$, d_{ij} is the Euclidean distance between customer c_i and c_j, Q_k denotes the value at which the k-th vehicle violates the capacity constraint. F_k is the time penalty cost of k-th vehicle, it is defined by Equation (11).

The constraints are as follows:

1. Depot constraint. All vehicles depart from the depot and return to the depot after completing all distribution tasks.

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{0j}^k = \sum_{k=1}^{K} \sum_{j=1}^{N} x_{j0}^k = K \quad (2)$$

2. Traffic flow constraint. The vehicle flow at the customer point is balanced, and the number of vehicles entering and exiting the customer is equal.

$$\sum_{i=1}^{N} x_{ij}^k = \sum_{j=1}^{N} x_{ji}^k \quad (3)$$

3. Capacity constraint. The load of each vehicle cannot exceed its maximum capacity limit.

$$\sum_{i=0}^{N}\sum_{j=0}^{N} x_{ij}^{k} m_j \le Y \tag{4}$$

where m_j represents the demand for goods from the j-th customer, Y is the maximum load capacity of the vehicle.

4. Customer service constraint. Each customer is only allowed to be visited once.

$$\sum_{k=1}^{K}\sum_{j=1}^{N} x_{j}^{k} = 1 \quad (j = 1, 2, \cdots, N) \tag{5}$$

5. Vehicle travel distance constraint. The distribution distance of each vehicle cannot exceed the maximum travel distance.

$$\sum_{i=0}^{N}\sum_{j=1}^{N} x_{ij}^{k} d_{ij} \le D_{max} \tag{6}$$

where d_{ij} is the Euclidean distance between customer c_i and c_j, D_{max} is the maximum distance traveled by the vehicle.

6. Required number of vehicles constraint. The number of vehicles cannot exceed the maximum number of vehicles.

$$\left\lceil \frac{\sum_{j=1}^{N} m_j}{Y} \right\rceil \le K \le M \tag{7}$$

7. Waiting time constraint.

$$W_i^k = max\left(0, ET_i - t_i^k\right) \tag{8}$$

where W_i^k represents the waiting time for the k-th vehicle to serve the i-th customer, ET_i represents the earliest service time of the i-th customer, t_i^k represents the time when the k-th vehicle arrived at the i-th customer.

8. Travel time constraint.

$$t_{ij} = \frac{d_{ij}}{S} \tag{9}$$

where t_{ij} represents the travel time from the i-th customer to the j-th customer, S represents the speed of the vehicle.

9. Time constraint.

$$t_j^k = t_i^k + W_i^k + T_i + t_{ij} \tag{10}$$

where the i-th customer is the previous customer from the j-th customer, t_i^k is the arrival time of the k-th vehicle at the i-th customer, W_i^k is the waiting time for the k-th vehicle to serve the i-th customer, T_i is the service time of the i-th customer, t_{ij} is the travel time from the i-th customer to the j-th customer.

10. Penalty cost.

$$F_i = \begin{cases} C_2\left(ET_i - t_i^k\right), & \text{if } t_i^k \text{ earlier than the earliest service time} \\ C_3\left(t_i^k - LT_i\right), & \text{if } t_i^k \text{ later than the latest service time} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where C_2 and C_3 are the waiting cost per unit time that arrives earlier than the earliest service time, and the penalty cost per unit time that arrives later than the latest service time, respectively($C_2 = C_3 = 100$). ET_i and LT_i denote the earliest and latest service time of the i-th customer, respectively.

B. THE PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart [22], and its basic concept is derived from the study of the foraging behavior of birds. Each particle can be regarded as a search individual in the n-dimensional search space. The particle's current position is a candidate solution to the corresponding optimization problem, and the particle's flight operation is the individual's search process. The flying speed of the particle can be dynamically adjusted according to the particle's historical optimal position and the population's historical optimal position. Each particle has two attributes: velocity and position. The velocity represents the speed at which it moves, and position represents the direction in which it moves. The process of updating the position of particles is shown in Figure 3. The optimal solution searched by each particle is called the individual, and the best individual in the population is the current global optimal solution. PSO continuously iterates and updates the velocity and position of particles in the population until the termination conditions are met, outputting the optimal individual. The flow chart of PSO is shown in Figure 4, and the pseudocode of PSO is shown in Algorithm 1. The steps of the classical particle swarm algorithm are as follows:

1. Population initialization. The initial parameters are set (maximum number of iterations, problem dimension, initial position and speed of individuals, population size).

2. Calculate the fitness value of each particle. The fitness function is defined, and each particle's fitness value is calculated based on the particle's position.

3. Update global and local optima. In each generation, individual local and global optima are updated based on the fitness values of each particle.

4. Calculate the velocity of particles and update their positions. In PSO, the velocity of particles refers to the direction in which they move in the next iteration, calculated by Eq.12. The position of particles is a feasible solution to the problem, updated by Eq.13.

$$v_i^{t+1} = \omega v_i^t + c_1 \times rand() \times (pbest_i - x_i) + c_2 \times rand() \times (gbest - x_i) \tag{12}$$

where v_i^{t+1} represents the i-th particle's velocity at t-th generation, ω is inertia weight, c_1 and c_2 are acceleration constants(Normally we set $c_1 = c_2 = 2.0$), the former is an individual learning factor, while the latter is a global learning factor. $rand()$ is a random number for [0,1], $pbest_i$ is the i-th particle individual local optimum and $gbest$ is the

Algorithm 1 Particle Swarm Optimization Algorithm

for *each particle i* **do**
  Initialize the velocity v_i and position x_i for the i-th particle
  Evaluate the i-th particle and update $pBest_i$ and $gBest$
while $t < Maxgen$ **do**
  for $i < N$ **do**
    Update the velocity of the i-th particle by Eq.12
    Update the position of the i-th particle by Eq.13
    Evaluate i-th particle
    if $fit(x_i) < fit(pBest_i)$ **then**
      $pBest_i = x_i$
    if $fit(pBest_i) < fit(gBest)$ **then**
      $gBest = pBest_i$
return global best solution;

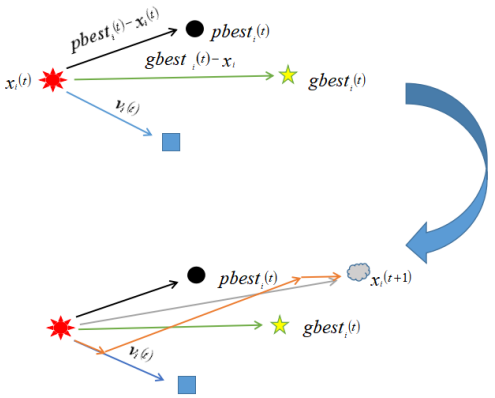


FIGURE 3. The update process of particle's position in PSO.

global optimum.

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{13}$$

5. Calculate the fitness value of particles. Evaluate the newly generated particles and update the individual and global optima.

6. Satisfy the end condition. If PSO does not meet the end condition, repeat steps 4∼5. If the end condition is met, it jumps out of the loop, terminates the search, and outputs the result.

PSO has been widely used in function optimization, image processing, combinatorial optimization, and many other fields because of its simple operation and fast convergence speed. With the expansion of the application range, the PSO algorithm has some problems, such as premature convergence and easy falling into the local extremum, which need to be further improved. PSO algorithm mainly has the following improvement development direction: 1) The parameters are adjusted to balance the global exploration and local exploitation ability. 2) Different types of topological structures can change particles' learning modes to improve the population's diversity. 3) PSO and other optimization algorithms (or strategies) are combined to form a hybrid PSO algorithm. Different types of algorithms were mixed, and the algorithm's

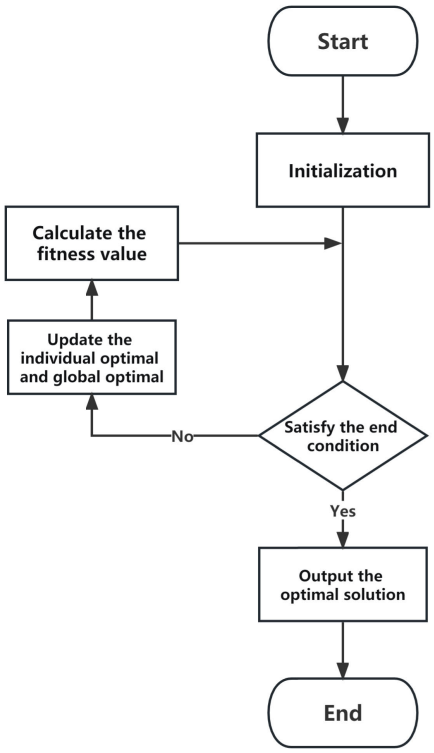


FIGURE 4. The flowchart of PSO.

advantages were combined to improve the performance further. 4) The minimum subpopulation grouping technique is adopted. It is suitable for solving multimodal function and multi-objective function optimization problems. The population was divided into several sub-populations as much as possible to form multiple relatively independent search Spaces, and the synchronous search of multiple extreme regions was realized to avoid premature convergence of the algorithm in solving multimodal function and multi-objective function optimization problems.

III. ENCODING AND DECODING OF POPULATIONS

Two core elements of the PSO are particle's velocity V and the particle's position X. The velocity represents the direction and distance that the particle moves in the next iteration, the position is a solution to the problem to be solved, and a particle represents a feasible solution. Suppose that in the d-dimensional search space, there are N particles, the position of the i-th particle is denoted by: $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$; The velocity of the i-th particle is denoted by: $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$.

When using swarm intelligence algorithms such as PSO to solve discrete problems, it is necessary to map the particle positions in the population to the decision variable value range of the discrete problem one by one. We use the two-row matrix method to construct particles to solve the VRPTW problem. Assuming the number of customers is d, the positions of the constructed particles are a matrix of two rows and L columns. The first row X_v represents the vehicle number

TABLE 1. Initial distribution scheme.

Vehicle 1	2	6
Vehicle 2	3	1
Vehicle 3	4	5

TABLE 2. The initial scheme is converted to particle positions.

Vehicle number	1	1	2	2	3	3
Customer	2	6	3	1	4	5

TABLE 3. Decoding and distribution scheme.

Customer number	1	2	3	4	5	6
X_v	2	1	2	2	3	1
X_r	2	1	1	3	1	2

served for each customer, with the second line X_r representing the execution order of each customer on the corresponding vehicle route. So the position of particles is represented by $[X_v, X_r]$. Firstly, the initial customer distribution scheme is constructed through population initialization, and it is stored in a matrix of NV 1 cell array. Each row of the matrix represents a delivery vehicle, and each cell array stores the sequence of customers to be delivered by that vehicle. Secondly, the initial scheme is transformed into a $2d$ matrix to represent the particle's initial solution. The first row of this matrix indicates the vehicle number, while the second row represents the order in which customers are delivered by each vehicle. Next, the encode operation encodes both the position and velocity of a particle, which are then combined into a $4d$ matrix for iteration purposes. Finally, the decode operation prints out the position matrix of the particle.

Suppose the number of customers is 6 and the number of vehicles is 3, a possible particle is as follows (the customer number is used to make it easier to understand how particles are expressed):

The initial scheme is converted to particle positions:

The position and velocity of a particle are then encoded by the encode operation. The initialization method for a particle's velocity is the same as that for a particle's position. The particle's velocity is $[V_v, V_r]$. Assuming that the number of customers is d and the maximum number of vehicles used is M, the value range of X_v is $[0, L]$, and the value range of X_r is $[0, M]$. The element in each position in V_v should be between $-(M-1)$ and $(M-1)$, and the element in each position in V_r should be between $-(d-1)$ and $(d-1)$.

Finally, the ScPSO algorithm is used to find the optimal solution, and then the position $[X_v, X_r]$ of a particle is represented by the decoding operation as follows:

From Table 3, we can see that customer 1 is served by the second vehicle at the second location, customer 2 is served by the first vehicle at the first location, customer 3 is served by the second vehicle at the first location, customer 4 is served by the second vehicle at the third location, customer 5 is served by the third vehicle at the first location, and customer 6 is served by the first vehicle at the second location. Therefore, the vehicle route (as shown in the Figure.5) can be expressed as: (where 0 represents the depot)

Vehicle 1: $0 \rightarrow 2 \rightarrow 6 \rightarrow 0$
Vehicle 2: $0 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 0$
Vehicle 3: $0 \rightarrow 5 \rightarrow 0$

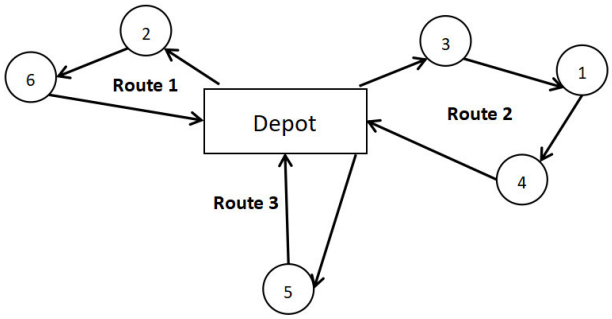


FIGURE 5. The vehicle route map.

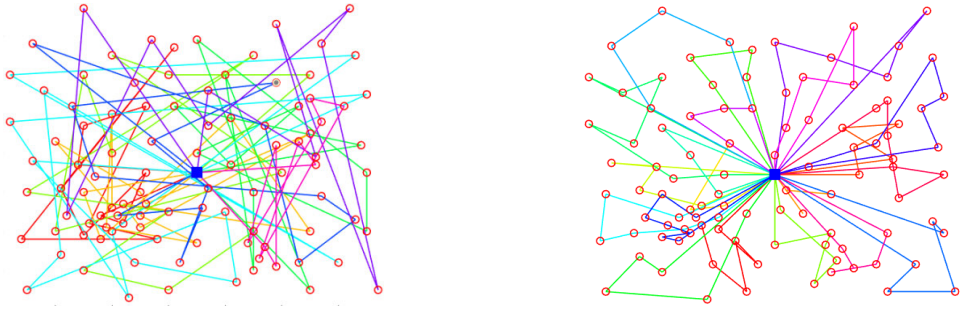
IV. SELF-COMPETITION PARTICLE SWARM OPTIMIZATION ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOW

The solution to the VRPTW problem is a complete directed graph. If the candidate solution finally satisfies the VRPTW time window constraint, it is proved valid. In solving the VRPTW problem, we take the minimization of the total vehicle travel distance TD as the primary goal and solve the service route with the fleet's minimum total vehicle travel distance, as shown in Figure 6.

In order to solve VRPTW, a Self-competition Particle Swarm Optimization (ScPSO) Algorithm is proposed. It can find the optimal vehicle service Routing with the minimum total vehicle travel distance. ScPSO combines position and inertia weight adaptation to expand the search range, avoid premature convergence into local optimum, and achieve accurate search. The updated direction of the particle's velocity was determined by calculating the fitness value ratio of the current particle and the population's average fitness value in each generation. When the particle updates its position, it is forced to learn from the individual optimum according to the self-competition selection probability. Through adaptive adjustment of ω, the global search ability and local search ability of particles are well balanced. The flowchart of ScPSO is shown in Figure.8, and the pseudo-code of the ScPSO is shown in Algorithm 2.

A. POPULATION INITIALIZATION

Constructing a good initial solution is crucial for solving complex problems. An excellent initial solution can significantly improve the speed and performance of the algorithm. We adopted a new method of constructing initial solutions, a random greedy heuristic selection (RGS), as shown in



(a) Initial solution.

(b) A feasible solution after searching.

FIGURE 6. The solution of VRPTW.

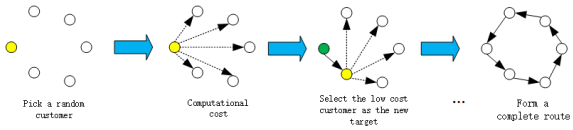


FIGURE 7. Random greedy heuristics selection.

Figure 7. It randomly selects unserved customers, calculates their insertion cost, and makes greedy choices based on the insertion cost. The RGS implementation process consists of three parts: calculating insertion costs, sorting, and greedy selection. First, a random customer is selected from the customer pool that has never been served, and the Cost value required to go from this customer to other customers is calculated, which is called the R policy. Secondly, the customers were sorted in ascending order according to the calculated Cost value, which was called the G strategy. Finally, the customer with a smaller Cost value is selected as the next research target by sorting to minimize the travel distance, which is the S strategy.

B. OBJECTIVE FUNCTION

When solving optimization problems, the objective function significantly impacts particle selection, update, and iteration. When dealing with VRPTW issues, there are two main objectives: NV and the TD traveled by vehicles. This paper mainly focuses on TD. Meanwhile, penalty cost functions are used to solve constraint violations and integrated into the objective function to improve the quality of the solution.

$$Cost = TD + \alpha \times Q + \beta \times W \tag{14}$$

where $Cost$ is an objective function, α is the coefficient of the penalty cost function for violating the capacity constraint, and Q is the sum of the load violations of each route. β is the coefficient of the penalty cost function for violating the time window constraint, and W is the sum of time window constraints violated by the current solution. We set α to 10 and β to 100 in the experiment.

C. VELOCITY UPDATE

In the particle swarm algorithm, the velocity determines the particle's next moving direction and distance. In order to fully utilize the exploration ability of the particles in the landscape, a reasonable range setting of each particle's velocity in each dimension is needed. ScPSO uses the learning probability factor L_i to control the learning direction of the particles in each dimension. By regulating the moving direction of particles in each dimension, ScPSO can improve particles' searching ability and enhance the population's diversity. During the iteration process, each particle's learning probability factor L_i is dynamically adjusted. When the learning probability factor is small, the particle exhibits a stronger inclination towards selecting its own best position in that dimension, thereby enhancing its performance within the population. Consequently, it is imperative for the particle to maintain its current trajectory. Conversely, when the habituation probability factor is large, the particle demonstrates a greater propensity to choose exemplars with superior fitness levels. This indicates that the particle faces a relative disadvantage within the population and should be compelled to learn from other particles exhibiting better characteristics. The learning probability factor L_i is calculated as follows:

$$L_i = L_{min} + (L_{max} - L_{min}) \times \frac{e^{\frac{10(i-1)}{N-1}} - 1}{e^{10} - 1} \tag{15}$$

where L_i is the learning probability of i-th particle, $L_{max} = 0.5$, $L_{min} = 0.05$ and N is population size.

During each iteration of ScPSO, for each dimension j of the i-th particle, a random number $rand()$ is generated. If this random number is less than L_i, this particle learns from other excellent particles $Nbest$ in the corresponding dimension. otherwise, If $rand()$ is greater than or equal to L_i, the particle continues to learn from itself $Pbest$. In addition, in each learning process, the other excellent particle is the best optimal particle of the current population in dimension j, except the i-th particle and $Gbest$. This method can effectively avoid the phenomenon of premature maturation of particles. Meanwhile, each dimension may learn from

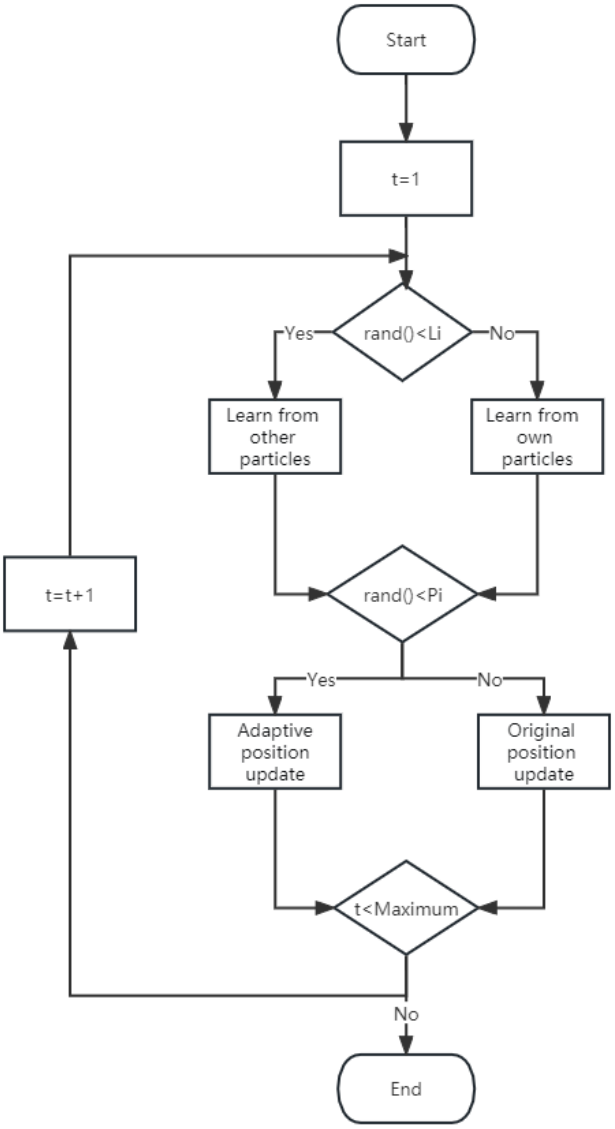


FIGURE 8. Self-competition particle swarm optimization algorithm flow chart.

different samples, which can improve the diversity of the population.

$$V_i^j(t+1) = \omega \times V_i^j(t) + c \times rand() \times (Nbest_i^j(t) - X_i^j(t)) \tag{16}$$

$$V_i^j(t+1) = \omega \times V_i^j(t) + c \times rand() \times (Pbest_i^j(t) - X_i^j(t)) \tag{17}$$

where $V_i^j(t+1)$ is the velocity of i-particle in j-th dimension at $(t+1)$-th generation. ω is the inertia weight, c is the learning factor. $Nbest$ is the best optimal particle in the current population, except the i-th particle and $Gbest$.

D. POSITION UPDATE

ScPSO updates the position of particles based on their self-competition degree. The self-competition degree of particles is obtained by calculating the ratio value of the current particle's fitness value to the population's average fitness in the current generation, which can measure the ranking of the particle in the population. When the self-competition degree of the particle is less than 1, it indicates that the performance of the particle is better than the average of the current population, and the search direction of the particle should continue to be maintained. When the self-competition degree of the particle is greater than 1, it indicates that the particle is inferior to the average of the current population, and the particle should be forced to learn optimally from its surrounding individual particles. The self-competitive learning strategy enhances the learning mechanism of particles. It can help the particles with poor fitness values to continuously adjust their learning direction and force them to learn from the surrounding optimal particles. This strategy can fully leverage the potential capabilities of inferior particles and avoid them falling into local optima.

$$P_i = \frac{exp(fit(x_i\,(t)))}{exp\left(\frac{\sum_{i=1}^{N} fit(x_i(t))}{N}\right)} \tag{18}$$

$$x_i(t+1) = \omega\,(x_i(t) + v_i(t+1)) + (1-\omega)\,\frac{\sum_{i=1}^{m} pbest_i}{m} \tag{19}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{20}$$

where P_i is the self-competition degree of i-th particle, $fit(.)$ is the fitness function, N is the population size, and t is the number of current generation. $x_i(t+1)$ is the position of i-th particle at $t+1$ generation, ω is the inertial weight, m is the size of the particle's influence range, $\frac{\sum_{i=1}^{m} pbest_i}{m} = \frac{pbest_1 + pbest_2 + \cdots + pbest_m}{m}$.

After each particle updates its position, a local search strategy called variable neighborhood search (VNS) performs local fine-tuning on the particles. VNS selects customers who violate time window or load constraints and inserts them into their adjacent vehicle routes, reducing penalty costs. Then, calculate the overall cost of the newly generated route and compare it with the overall cost of the old route to determine whether to perform fine-tuning. The specific operation steps are as follows:

1. Determine whether the particle meets the termination condition. If the particle does not meet the termination condition, go to STEP 2; Otherwise, go to STEP 5.

2. Calculate the value of time window constraints violated by each customer on each vehicle route (VR). This value equals the service time of the vehicle to the customer - the end time of the customer's available service. If this value is greater than 0, it indicates that this customer has violated the time window constraint. The higher the value, the more severe the violation of the time window constraint. Then, VNS selects the customer with the highest violation of time window constraints from each route and stores it in the customer constraint matrix (MT). Among them, the first column records the customer number that violated the

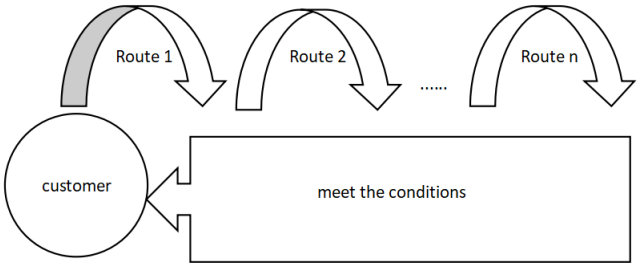


FIGURE 9. Local fine-tuning.

constraint, and the second column records the value that violated the time window constraint. If no customers violate the time window constraint on the route, both columns of the MT corresponding to the route are marked as 0. Go to STEP 3.

3. If there is a value greater than 0 in MT in this route, then in the second column of MT, find the customer with the maximum violation value of the time window constraint, mark it as H, and then remove H from the current route and go to STEP 4.

4. Fine-tuning customer H. Customer H uses VNS for local fine-tuning, searching for new vehicle routes that can reduce penalty costs. VNS selects the vehicle route with the lowest total cost from all search results and inserts customer H into the current vehicle route to form a new one. If no usable vehicle route is found, no further fine-tuning will be performed on customer H. Go to STEP 5.

5. Terminate the loop and the vehicle delivery route is updated.

E. ADAPTATION OF INERTIA WEIGHT

In the basic PSO algorithm, the dynamic adjustment of inertia weight is proposed to balance the global convergence and convergence speed. At the same time, it is pointed out that a larger inertia weight value is conducive to global search, and a smaller inertia weight value is more conducive to local search. Inertial weight ω represents the influence of the previous generation on the current particle, and it can reflect the trust degree of the particle in its current motion state. Each particle moves according to its speed, and the inertia weight can help it maintain its inertia and trend of movement. The larger the inertia weight value, the stronger the ability of particles to explore new areas, that is, the stronger its global optimization ability. On the contrary, the weaker the global optimization ability of particles, the stronger the local optimization ability. A larger inertia weight benefits global search and avoids falling into local optimum. A smaller inertia weight benefits local search and quickly converges to the optimal solution. In the early search stage, particles must have a high global search ability. In the later search stage, particles need to have high local search ability to improve the accuracy of the solution. When the problem space is large, it is necessary to dynamically control the inertia weight to balance the convergence speed and solution accuracy. The adaptive

Algorithm 2 Self-Competition Particle Swarm Optimization Algorithm

```
Initialize each particle's position X_i and velocity V_i;
Evaluate the fitness value of each particle;
Update pbest_i and gbest;
while t < maxgen do
    for i < N do
        for j < D do
            if rand() < L_i then
                Learning from other particles by Eq.16;
            else
                Learning from its own particle by Eq.17;
        Update particle's position by Eq.20;
        Calculate the ratio value of P_i by Eq.18;
        if rand() < P_i then
            Update particle's position by Eq.19;
        else
            Update particle's position by Eq.20;
        Update gBest;
return global best vehicle route;
```

changes in inertia weights ω are as follows:

$$\omega = \omega_{max} - \frac{t}{maxgen}(\omega_{max} - \omega_{min}) \tag{21}$$

where ω_{max} is the maximum inertia weight, ω_{min} is the minimum, t is the current iteration number, and $maxgen$ is the maximum number of iterations.

V. EXPERIMENTAL STUDIES

A. EXPERIMENTAL SETTINGS

In this section, we compared ScPSO with five state-of-the-art algorithms on Matlab 2022b for the 100 customer instances of Solomon's VRPTW benchmark problem. This VRPTW benchmark problem has 56 sub-problems, which are divided into 6 different types (C1, R1, RC1, C2, R2, and RC2). In problem set C, customers show a clustering structure with a relatively concentrated distribution. Therefore, the solution results of different algorithms are similar, and the minimum total distance TD changes smoothly. In problem set R, customers are randomly generated, and TD solved by different algorithms exhibit significant volatility. Problem set RC is a mixture of problem sets C and Problem sets R. Problem sets R1, C1, and RC1 have a short scheduling horizon and allow only a few customers per route (approximately 5 to 10). In contrast, the sets R2, C2, and RC2 have a long scheduling horizon, permitting many customers (more than 30) to be serviced by the same vehicle.

These five state-of-the-art algorithms are S-PSO-VRPTW [41], HGA [42], MS-EAS [43], TSA [26], and best-known solutions reported on Solomon's web page (http://web.cba.neu.edu/ ~msolomon/problems.htm). S-PSO-VRPTW is a set-based PSO to solve VRPTW.

TABLE 4. Comparison results with PSO variant algorithm.

Instance	ScPSO		Mean		S-PSO-VRPTW		Instance	ScPSO		Mean		S-PSO-VRPTW	
	NV	TD	NV	TD	NV	TD		NV	TD	NV	TD	NV	TD
C101	10	828.9369	10	828.9369	10	828.937	R112	11	**1001.9256**	10.96667	1054.046	10	1029.124
C102	10	828.9369	10.06667	851.727	10	829.712	R201	7	**1168.0658**	6.566667	1195.306	4	1274.969
C103	10	**828.0649**	10	887.4657	10	851.373	R202	6	**1043.6839**	5.8	1076.467	3	1247.033
C104	10	**852.3016**	10.06667	948.9704	10	868.521	R203	5	**882.4433**	4.8	921.3485	3	1052.712
C105	10	828.9369	10.03333	841.5001	10	828.937	R204	5	**746.4704**	4.3	775.8192	3	844.161
C106	10	828.9369	10.13333	10.13333	10	828.937	R205	5	**962.9816**	5.366667	996.2831	3	1061.46
C107	10	828.9369	10.06667	882.3518	10	828.937	R206	5	**892.4767**	4.8	927.5445	3	1061.346
C108	10	828.9369	10.2	911.2592	10	828.937	R207	5	**817.1078**	4.366667	850.8668	3	946.778
C109	10	854.783	10	964.0414	10	**828.937**	R208	4	**710.2954**	3.5	739.2849	2	834.721
C201	3	591.5566	3	591.5566	3	591.557	R209	5	**871.4047**	5.033333	901.5015	3	1003.188
C202	3	591.5566	3.433333	610.3584	3	591.557	R210	5	**921.4213**	5.433333	947.402	3	1040.544
C203	3	591.1734	3.733333	611.503	3	591.17	R211	5	**782.6302**	4.366667	804.4532	3	861.323
C204	3	**596.5545**	3.466667	610.5823	3	615.43	RC101	16	1665.2687	16.3	1709.696	15	**1641.204**
C205	3	588.876	3	588.876	3	588.876	RC102	14	**1493.5779**	14.6	1539.999	13	1510.952
C206	3	**588.4928**	3	588.4928	3	588.876	RC103	12	1305.399	12.43333	1365.53	11	**1294.739**
C207	3	**588.2863**	3	588.2863	3	591.35	RC104	11	**1188.3222**	11.26667	1240.399	10	1190.545
C208	3	**588.3238**	3.033333	592.7475	3	588.493	RC105	15	**1553.154**	15.63333	1590.738	14	1603.707
R101	20	1655.8696	19.63333	1670.312	19	**1652.001**	RC106	14	1427.708	13.96667	1495.712	12	**1410.931**
R102	18	**1474.7469**	18.06667	1486.978	17	1500.809	RC107	12	1280.4517	12.63333	1352.822	11	**1249.795**
R103	14	**1227.2734**	14.33333	1251.532	14	1242.216	RC108	11	1210.7384	11.86667	1266.715	11	**1181.87**
R104	10	**1012.9727**	11.16667	1041.72	10	1042.216	RC201	8	**1286.6219**	7.033333	1330.52	4	1423.519
R105	15	**1375.7402**	15.66667	1435.679	14	1385.082	RC202	7	**1116.7943**	5.966667	1160.303	4	1193.591
R106	14	**1259.0205**	13.33333	1296.486	12	1294.869	RC203	5	**945.7553**	5.333333	968.718	3	1123.419
R107	11	**1094.737**	11.8	1127.234	11	1123.981	RC204	4	**791.3987**	4.466667	815.6452	3	894.117
R108	11	**978.494**	10.56667	1008.175	10	1011.682	RC205	7	**1159.4157**	6.733333	1218.843	4	1321.429
R109	12	**1185.7692**	13.23333	1250.132	12	1211.63	RC206	6	**1063.5657**	5.533333	1113.712	3	1307.9
R110	12	**1108.1541**	12.23333	1161.74	11	1190.362	RC207	6	**966.079**	5.666667	1019.749	3	1130.368
R111	11	**1071.8615**	11.93333	1125.228	11	1102.987	RC208	4	**780.7162**	4.733333	823.8457	3	958.236

It considers the discrete search space as the set of arcs of the complete graph defined by the nodes in VRPTW and the candidate solutions as a subset of arcs. HGA is a different hybridization of artificial intelligence-based techniques, including simulated annealing, tabu search, and genetic algorithm for better performance in VRPTW. MS-EAS has a mutation strategy including ant colony mutation and optimal ant mutation to improve the local search ability. TSA is a metaheuristic method based on annealing restart to diversify and enhance local search to solve VRPTW. The best-known results reported on Solomon's web page are obtained from 23 algorithms, including exact, heuristic, and metaheuristic algorithms.

In the experimental parameter settings of ScPSO, the population size N is 50, the maximum iteration number $MaxGen$ is set to 100, the number of runs is 30, $Mean$ represents the average of 30 runs, the lower bound of vehicle number X_v and customer number X_r is 1, the upper bound of X_v is the maximum number of vehicle routes, and the upper limit of X_r is the number of customers served. The learning factor C of the velocity is set to a fixed value of 1.5, the value of $rand()$ is between [0,1], the maximum learning probability $L_{max} = 0.45$, the minimum learning probability $L_{min} = 0.05$. In the rest part of this section, we also conducted parameter sensitivity experiments on the learning factor C in ScPSO. Based on the experimental results, the value of parameter C was determined to be a fixed value of 1.5, updating the average cost is $O(MaxGen \times N)$, the complexity of VNS is $O(MaxGen)$.

B. RESULTS AND COMPARISONS

1) COMPARISON WITH PSO VARIANT ALGORITHM

In order to test the difference between ScPSO and PSO-based algorithms, the S-PSO-VRPTW algorithm was chosen for comparison experiments. The comparison results are shown in Table 4.

From Table 4, the performance of ScPSO is superior to S-PSO-VRPTW in 39 out of 56 problems and similar to S-PSO-VRPTW in 10 problems. In problem set C, there are 17 problems, of which C1 contains 9 problems, and C2 contains 8 problems. Through experiment results, we find that the performance of ScPSO outperforms S-PSO-VRPTW on 6 problems and is similar to S-PSO-VRPTW on 10 problems. In problem set C1, ScPSO outperforms S-PSO-VRPTW on 2 problems and performs similarly on 6 problems. In problem set C2, ScPSO has the best solution performance on 4 problems, and there are 4 problems with similar solution performance to S-PSO-VRPTW. The solution results of the problems are mostly the same because the customer in C has a structure of clusters. Overall, ScPSO outperforms S-PSO-VRPTW. ScPSO employs a local fine-tuning strategy to efficiently search for optimal delivery routes for customers that do not satisfy the time window constraints. These customers are incorporated into the searched routes to minimize the total traveled distance by the vehicles.

In problem sets R, it has a total of 23 problems. Among them, R1 contains 12 problems, and R2 contains 11 problems. Compared with the S-PSO-VRPTW, the running results of

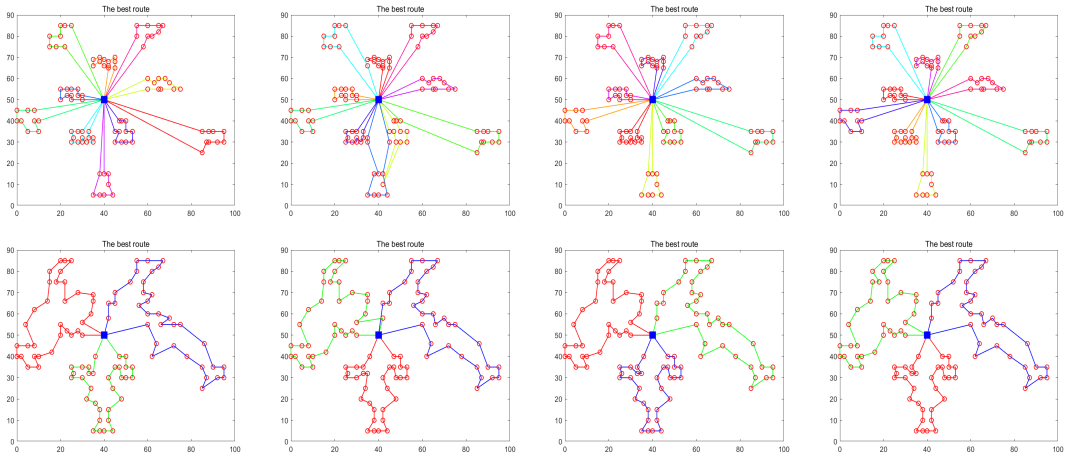


FIGURE 10. The best vehicle routing map of problem sets C.

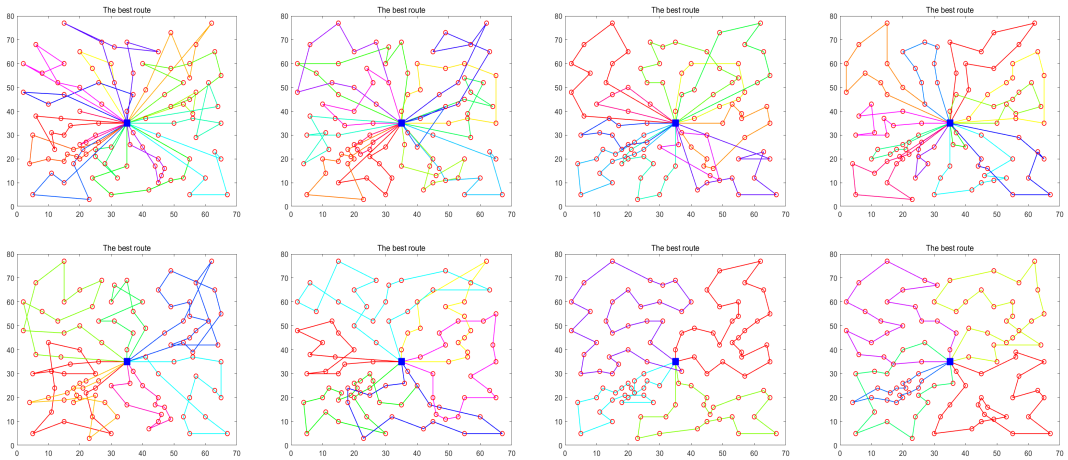


FIGURE 11. The best vehicle routing map of problem sets R.

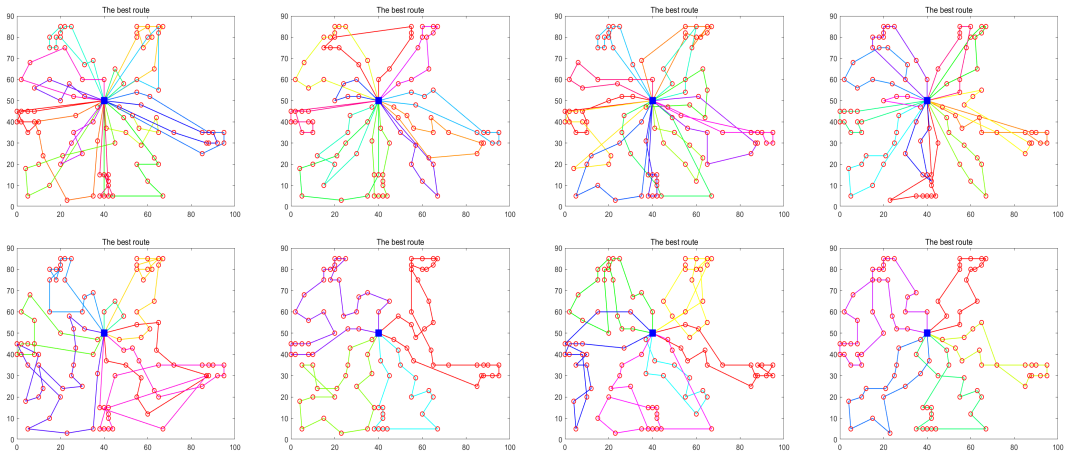


FIGURE 12. The best vehicle routing map of problem sets RC.

the ScPSO algorithm are better than those on 22 problems in the problem sets R. In problem set R1, there are 11 problems in which the ScPSO algorithm performs better than the S-PSO-VRPTW algorithm. In problem set R2, the results of ScPSO are better than the S-PSO-VRPTW algorithm on 11 problems. Because the customers of problem set R

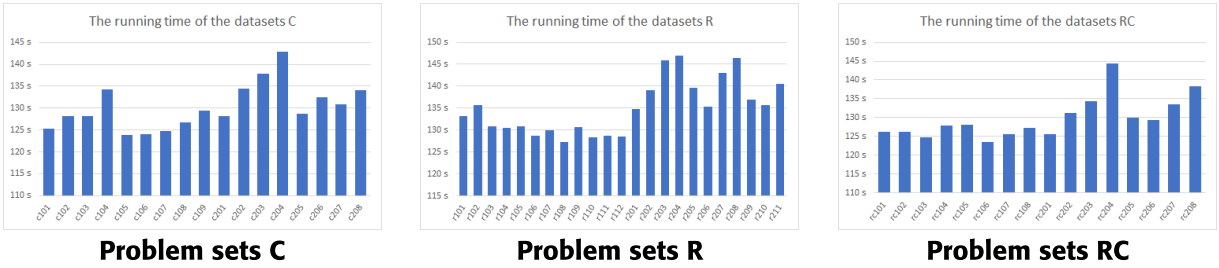


FIGURE 13. The runing time of ScPSO.

show the characteristics of random distribution. ScPSO uses the particle's self-competition learning strategy to adjust the learning direction of particles. It can control the exploration of particles and force particles to learn from the other optimal particles. The particles approach the optimal solution to obtain an optimal route that minimizes the total travel distance.

The problem sets RC has a total of 16 problems. Of these problem sets, RC1 and RC2 both have 8 problems. Compared with the running results of the S-PSO-VRPTW algorithm, the performance of ScPSO is better than the S-PSO-VRPTW algorithm on 11 problems in the problem sets RC. In the problem sets RC1, the performance of ScPSO is better than S-PSO-VRPTW on 3 problems. In the problem sets RC2, the performance of ScPSO is better than S-PSO-VRPTW on 8 problems. The problem sets RC concentrates the customer's characterization of problem sets C and problem sets R. ScPSO plays a good role in dealing with this kind of problem. Because its self-competitive learning strategy and local fine-tuning strategy of particles play a perfect role, the mixed–use of the two strategies can better solve the problem of problem sets RC.

The running time of ScPSO are shown in Figure 13, the customers in Problem sets R are more spread out, so the run time is longer. Problem sets C and Problem sets RC have the property of centralized distribution, so the running time is shorter than that of Problem sets R. In general, ScPSO has more advantages in dealing with randomly distributed customer problems. It outperforms S-PSO-VRPTW in handling VRPTW problems. Overall, ScPSO algorithm has better results in solving VRPTW problem, mainly due to its ability to find an acceptable final solution quickly by constructing an excellent initial feasible solution. Thereby, it can reduce the cost by balancing the particles' global and local search capabilities.

2) COMPARISON WITH OTHER STATE-OF-THE-ART ALGORITHMS ALGORITHMS

In order to comprehend the performance gap between ScPSO and other state-of-the-art heuristics, three meta-heuristic algorithms were selected for comparative experimental analysis. The experimental results are shown in Table 5.

Firstly, ScPSO performs better than the MS-EAS algorithm in most problems. ScPSO outperforms MS-EAS on 52 out of 56 problems and performs similarly on 3 problems. These 52 problems include 13 problems of problem set C, 23 problems of problem set R, and 16 problems of problem set RC. In problem sets C, ScPSO performs better than the MS-EAS algorithm on 5 problems of C1 and 8 problems of C2. In problem sets R, ScPSO performs better than the MS-EAS algorithm on 12 problems of R1 and 11 problems of R2. In problem sets RC, ScPSO performs better than the MS-EAS algorithm on 8 problems of RC1 and 6 problems of RC2. ScPSO utilizes the inertia weights adaptive strategy to gradually make the particles shift from global search to local search in the evolution process, thus improving the search accuracy of ScPSO. Meanwhile, its self-competitive learning strategy enables the particles to eliminate themselves quickly, thus improving the search accuracy of the algorithm.

Secondly, ScPSO outperforms the HGA algorithm on 46 out of 56 problems and performs similarly to the HGA algorithm on 4 problems. Specifically, in problem set C, ScPSO performs better than the HGA algorithm on the 6 problems of problem set C1 and 7 problems of problem set C2. In problem set R, the performance of ScPSO in 11 problems of problem set R1 and 11 problems of problem set R2 is better than that of the HGA algorithm. In problem set RC, ScPSO performs better than the HGA algorithm in all 8 problems of problem set RC2. ScPSO can find an acceptable final solution by constructing an excellent feasible solution and improves the search accuracy by random greedy selection heuristic strategy.

Finally, ScPSO performs better than the TSA algorithm on 29 out of 56 problems and performs similarly to the TSA algorithm on 13 problems. These 29 problems include 19 problems of problem set R and 10 problems of problem set RC. In problem set C, ScPSO performs as well as the TSA algorithm on 7 problems of the problem set C1 and 7 problems of the problem set C2. In problem set R, ScPSO performs better than the TSA algorithm in all 11 problems of the problem set R2. In problem set RC, ScPSO performs better than the TSA algorithm in all 8 problems of the problem set RC2.

Table 5 shows that ScPSO has the best results in solving VRPTW problems. The experimental results show that the solution results of ScPSO and other algorithms on problem set C are similar. Because the distribution of customers in problem set C shows clustering characteristics and is relatively concentrated. ScPSO shows the best experimental

TABLE 5. Comparison with other state-of-the-art algorithms.

Instance	ScPSO		Mean		MS-EAS		HGA		TSA	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C101	10	828.9369	10	828.937	10	828.94	10	828.937	10	828.94
C102	10	828.9369	10.06667	851.727	10	834.53	10	832.672	10	828.94
C103	10	**828.0649**	10	887.4657	10	863.04	10	859.779	10	**828.06**
C104	10	852.3016	10.06667	948.9704	10	890.52	10	893.225	10	**824.78**
C105	10	828.9369	10.03333	841.5001	10	828.94	10	828.937	10	828.94
C106	10	828.9369	10.13333	10.13333	10	830.33	10	836.653	10	828.94
C107	10	828.9369	10.06667	882.3518	10	828.94	10	828.937	10	828.94
C108	10	828.9369	10.2	911.2592	10	837.06	10	861.999	10	828.94
C109	10	854.783	10	964.0414	10	833.61	10	890.92	10	**828.94**
C201	3	591.5566	3	591.5566	4	643.15	3	591.557	3	591.56
C202	3	**591.5566**	3.433333	610.3584	4	627.84	3	613.478	3	**591.56**
C203	3	**591.1734**	3.733333	611.503	4	711.99	3	646.776	3	**591.17**
C204	3	596.5545	3.466667	610.5823	4	727.22	4	605.345	3	**590.6**
C205	3	588.876	3	588.876	3	592.9	4	631.21	3	588.88
C206	3	**588.4928**	3	588.4928	3	600.08	3	648.915	3	588.49
C207	3	**588.2863**	3	588.2863	3	618.74	3	609.157	3	**588.29**
C208	3	**588.3238**	3.033333	592.7475	3	635.21	3	614.951	3	**588.32**
R101	20	1655.8696	19.63333	1670.312	31	2022.12	19	1660.33	19	**1650.8**
R102	18	**1474.7469**	18.06667	1486.978	25	1923.3	18	1486.82	17	1486.41
R103	14	**1227.2734**	14.33333	1251.532	19	1553.54	13	1272.14	13	1292.68
R104	10	**852.3016**	10.06667	948.9704	13	1271.94	11	1073.63	9	1007.31
R105	15	**1375.7402**	15.66667	1435.679	22	1777.58	16	1380.44	14	1381.37
R106	14	**1259.0205**	13.33333	1296.486	18	1579.65	13	1283.06	12	1269.72
R107	11	**1094.737**	11.8	1127.234	15	1293.31	12	1117.81	10	1104.66
R108	11	**978.494**	10.56667	1008.175	12	1189.34	10	995.361	9	986.25
R109	12	1185.7692	13.23333	1250.132	17	1459.08	14	**1101.37**	11	**1208.96**
R110	12	**1108.1541**	12.23333	1161.74	14	1262.72	11	1162.19	10	1159.35
R111	11	1071.8615	11.93333	1125.228	15	1323.24	12	1093.76	11	**1066.32**
R112	11	1001.9256	10.96667	1054.046	12	1148.53	10	1012.85	10	**967.88**
R201	7	**1168.0658**	6.566667	1195.306	17	1537.09	6	1243.18	4	1252.37
R202	6	**1043.6839**	5.8	1076.467	12	1537.09	6	1188.91	4	1084.767
R203	5	**882.4433**	4.8	921.3485	8	1177.65	5	1050.03	3	949.369
R204	5	**746.4704**	4.3	775.8192	5	969.51	4	800.361	2	849.05
R205	5	**962.9816**	5.366667	996.2831	10	1330.71	5	1056.54	3	1032.55
R206	5	**892.4767**	4.8	927.5445	7	1277.98	5	984.648	3	931.62
R207	5	**817.1078**	4.366667	850.8668	6	1150.09	3	920.269	2	905.13
R208	4	**710.2954**	3.5	739.2849	3	1046.26	3	770.691	2	732.8
R209	5	**871.4047**	5.033333	901.5015	7	1296.69	3	902.67	3	930.59
R210	5	**921.4213**	5.433333	947.402	9	1192.05	5	1037.58	3	1018.95
R211	5	**782.6302**	4.366667	804.4532	5	1085.51	4	887.476	3	801.81
RC101	16	1665.2687	16.3	1709.696	22	2228.64	15	1658.96	15	**1658.62**
RC102	14	**1493.5779**	14.6	1539.999	21	1987.83	15	1514.85	13	1513.6
RC103	12	1305.399	12.43333	1365.53	15	1616.17	13	**1149.86**	11	1219.99
RC104	11	1188.3222	11.26667	1240.399	12	1328.29	10	1173.47	10	**1141.09**
RC105	15	**1553.154**	15.63333	1590.738	21	1976.94	16	1585.34	13	1637.62
RC106	14	1427.708	13.96667	1495.712	16	1636.73	15	**1403.1**	11	1424.73
RC107	12	1280.4517	12.63333	1352.822	14	1478.84	13	1290.76	11	**1240.66**
RC108	11	1210.7384	11.86667	1266.715	12	1303.07	10	1157.2	10	**1147.42**
RC201	8	**1286.6219**	7.033333	1330.52	16	1962.33	5	1354.96	4	1425.21
RC202	7	**1116.7943**	5.966667	1160.303	13	1521.47	5	1257.48	3	1374.27
RC203	5	**945.7553**	5.333333	968.718	8	1264.77	5	1063.77	3	1088.53
RC204	4	**791.3987**	4.466667	815.6452	4	1007.49	4	899.347	3	818.66
RC205	7	**1159.4157**	6.733333	1218.843	15	1630.07	8	1236.18	4	1304.64
RC206	6	**1063.5657**	5.533333	1113.712	11	1523.88	5	1133.86	3	1159.03
RC207	6	**966.079**	5.666667	1019.74	8	1523.88	5	1068.73	3	1107.16
RC208	4	**780.7162**	4.733333	823.8457	5	1137.74	5	854.749	3	862.34

results when solving problems consisting of R and RC. Problem set R contains randomly generated customers, while problem set RC is a mixture of problem set C and problem set R. It is mainly attributed to ScPSO employing a self-competitive learning strategy to control the exploration and exploitation of particles. This strategy motivates the particles to learn towards the optimal solution continuously. Also, ScPSO incorporates a local fine-tuning strategy based on variable neighborhood search, which results in higher accuracy of the solution.

3) COMPARISON WITH THE BEST-KNOWN SOLUTIONS

The experiment between ScPSO and best-known solutions is carried out on the Solomon 100-customers benchmark problems. The experimental results of the best-known solutions are reported on Solomon's web page. From Table 6, It can be found that the performance of ScPSO is better than the best-known solution on most benchmark problems.

In the 56 benchmark problems, the solutions of ScPSO are the same as those of the best-known solutions on 14 problems. At the same time, ScPSO has better experimental results

TABLE 6. Compared with the best-known solutions.

Instance	ScPSO		Best Known		Instance	ScPSO		Best Known	
	NV	TD	NV	TD		NV	TD	NV	TD
C101	10	828.9369	10	828.94	R112	11	1001.9256	10	**982.14**
C102	10	828.9369	10	828.94	R201	7	**1168.0658**	9	1252.37
C103	10	828.0649	10	828.06	R202	6	**1043.6839**	8	1191.7
C104	10	852.3016	10	**824.78**	R203	5	**882.4433**	6	939.5
C105	10	828.9369	10	828.94	R204	5	**746.4704**	4	825.52
C106	10	828.9369	10	828.94	R205	6	**962.9816**	5	994.43
C107	10	828.9369	10	828.94	R206	5	**893.1152**	5	906.14
C108	10	828.9369	10	828.94	R207	4	**817.1078**	4	890.61
C109	10	864.5624	10	**828.94**	R208	4	**710.2954**	4	726.82
C201	3	591.5566	3	591.56	R209	5	**871.4047**	5	909.16
C202	3	591.5566	3	591.56	R210	5	**921.4213**	5	939.37
C203	3	591.1734	3	591.17	R211	4	**782.6302**	4	885.71
C204	3	596.5545	3	**590.6**	RC101	16	**1665.2687**	15	1696.95
C205	3	588.876	3	588.88	RC102	14	**1493.5779**	14	1554.75
C206	3	588.4928	3	588.49	RC103	12	1305.399	11	**1261.67**
C207	3	588.2863	3	588.29	RC104	11	1188.3222	10	**1135.48**
C208	3	588.3238	3	588.32	RC105	15	**1553.154**	16	1629.44
R101	20	1655.8696	18	**1650.8**	RC106	14	1427.708	13	**1424.73**
R102	18	**1474.7469**	18	1486.12	RC107	12	1280.4517	12	**1230.48**
R103	14	**1227.2734**	14	1292.67	RC108	11	1210.73841	11	**1139.82**
R104	10	1012.9727	10	**1007.31**	RC201	8	**1286.6219**	6	1406.94
R105	15	**1375.7402**	15	1377.11	RC202	7	**1116.7943**	8	1365.65
R106	14	1259.0205	13	**1252.03**	RC203	5	**945.7553**	5	1049.62
R107	11	**1094.737**	11	1104.66	RC204	4	**791.3987**	4	798.46
R108	11	978.494	10	**960.88**	RC205	7	**1159.4157**	7	1297.65
R109	12	**1185.7692**	13	1194.73	RC206	6	**1063.5657**	7	1146.32
R110	12	**1108.1541**	12	1118.84	RC207	6	**966.079**	6	1061.14
R111	12	**1071.8615**	12	1096.72	RC208	4	**780.7162**	4	828.14

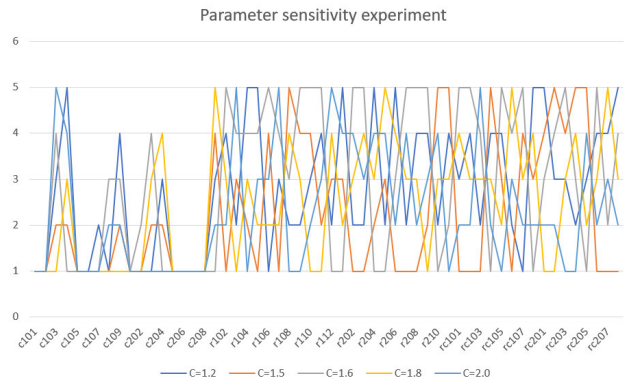


FIGURE 14. Parameter sensitivity experiment.

than the best-known solutions on 25 out of 56 problems. These 29 problems include 7 problems in the problem set R1, 11 problems in the problem set R2, and 11 problems in the problem set RC. These results demonstrate the best performance of the ScPSO in dealing with VRPTW problems.

ScPSO and the well-known solutions run on problem set C with similar results. It may be due to the centralized distribution of customers in problem set C. Different algorithms have fewer differences in solutions during the search process, thus obtaining similar results. However, for problem sets R and RC, the ScPSO algorithm shows a significant advantage. The customer distributions of problem sets R and RC are more complex and random, requiring the algorithms to have more vital search ability and adaptability. The particle self-elimination method of ScPSO can quickly help the particles find the optimal solution's direction and thus find a more reasonable vehicle route. Overall, the ScPSO algorithm shows good performance in solving the VRPTW problem.

C. PARAMETER SENSITIVITY EXPERIMENT

In particle swarm algorithms, the learning factor c plays a crucial role in determining the behavioral patterns of the particles during the search process. This factor c, also known as acceleration factor or acceleration factor, is mainly used to adjust the role of one's experience and population experience in the search process. c represents the weight of the part of the particle's following action derived from its own experience, which is the acceleration weight of the particle's movement towards the individual optimal position. When c is equal

TABLE 7. Parameter sensitivity experiment.

Instance	c=1.2		c=1.5		c=1.6		c=1.8		c=2.0	
	NV	TD	NV	TD	NV	TD	NV	TD	NV	TD
C101	10	828.9369	10	828.9369	10	828.9369	10	828.9369	10	828.9369
C102	10	828.9369	10	828.9369	10	828.9369	10	828.9369	10	828.9369
C103	10	855.6055	10	**828.0649**	10	878.5692	10	**828.0649**	10	881.7063
C104	10	895.7104	10	852.3016	10	**847.1029**	10	891.7965	10	892.5811
C105	10	828.9369	10	828.9369	10	828.9369	10	828.9369	10	828.9369
C106	10	828.9369	10	828.9369	10	828.9369	10	828.9369	10	828.9369
C107	10	863.7031	10	**828.9369**	10	**828.9369**	10	**828.9369**	10	**828.9369**
C108	10	**828.9369**	10	**828.9369**	10	919.7411	10	**828.9369**	10	857.281
C109	10	929.7966	10	864.5624	10	902.5642	10	**828.9369**	10	854.783
C201	3	591.5566	3	591.5566	3	591.5566	3	591.5566	3	591.5566
C202	3	**591.5566**	3	**591.5566**	3	629.3084	3	591.5566	3	**591.5566**
C203	3	**591.1734**	3	**591.1734**	3	624.2748	3	603.3668	3	**591.1734**
C204	3	601.1755	3	**596.5545**	3	**596.5545**	3	607.8113	3	**596.5545**
C205	3	588.876	3	588.876	3	588.876	3	588.876	3	588.876
C206	3	588.4928	3	588.4928	3	588.4928	3	588.4928	3	588.4928
C207	3	588.2863	3	588.2863	3	588.2863	3	588.2863	3	588.28639
C208	3	588.3238	3	588.3238	3	588.3238	3	588.3238	3	588.3238
R101	19	1661.9864	20	1655.8696	20	**1648.7649**	19	1665.2463	20	1657.241
R102	18	1479.3763	18	**1474.7469**	18	1490.2964	18	1477.892	18	1475.6934
R103	14	1226.0205	14	1655.8696	14	1239.2294	14	**1221.5924**	14	1240.9722
R104	11	1048.6446	10	**1655.8696**	12	1043.6518	11	1036.6418	12	1018.1791
R105	15	1435.7415	15	**1375.7402**	15	1414.1768	16	1403.5166	15	1405.1662
R106	11	1048.6446	14	1259.0205	12	1043.6518	11	1036.6418	12	**1018.1791**
R107	12	1117.8243	11	**1094.737**	11	1124.7072	11	1110.0425	11	1134.9509
R108	10	985.6187	11	**978.494**	10	986.5084	10	986.8873	10	982.4342
R109	12	1221.6305	12	**1185.7692**	14	1285.295	13	1227.9125	13	1218.1285
R110	12	1166.4603	12	**1108.1541**	13	1233.898	12	1111.8014	12	1140.0032
R111	12	1140.4122	11	1071.8615	12	1148.9842	12	**1071.3057**	13	1122.2805
R112	11	1015.5154	11	**1001.9256**	11	1015.2742	11	1034.3213	11	1040.9334
R201	6	1198.8614	7	**1168.0658**	7	1172.63	7	1176.7719	6	1182.8035
R202	6	1061.6033	6	**1043.6839**	5	1079.3814	6	1067.9542	6	1076.0484
R203	5	894.1994	5	**882.4433**	6	927.0986	6	903.6413	6	897.7565
R204	5	786.1249	5	746.4704	4	**741.2237**	4	768.6008	4	786.0998
R205	6	973.1752	6	**962.9816**	6	973.0828	6	1002.9201	6	978.5582
R206	5	938.829	4	**893.1152**	6	920.0482	6	920.4073	5	909.5356
R207	5	**813.5473**	5	817.1078	5	845.1376	4	818.5332	4	836.1346
R208	4	750.9974	4	**710.2954**	4	759.8743	4	728.2996	4	720.374
R209	5	898.0731	5	871.4047	6	907.3863	5	**867.8064**	5	881.1839
R210	6	931.3967	5	**921.4213**	7	925.6216	5	932.5647	5	935.3481
R211	4	786.519	4	782.6302	5	776.1797	4	786.1618	5	**771.8661**
RC101	16	1689.8202	16	**1665.2687**	16	1737.5976	16	1696.8879	15	1680.7671
RC102	15	1538.8448	14	**1493.5779**	14	1559.5497	15	1510.1966	14	1509.2999
RC103	12	1347.0947	12	**1305.399**	13	1358.5405	13	1347.7393	12	1360.0605
RC104	11	1249.0529	11	1188.3222	10	**1187.3982**	11	1242.9647	11	1223.7716
RC105	16	1589.9532	15	**1553.154**	15	1638.267	15	1571.418	15	1557.0652
RC106	14	1457.041	14	**1427.708**	14	1523.6504	13	1524.2614	13	1469.9229
RC107	13	1302.7751	12	**1280.4517**	13	1382.6271	13	1345.1163	12	1312.1874
RC108	12	1313.1572	11	**1210.7384**	11	1217.2189	12	1267.457	12	1223.7566
RC201	7	1324.9164	8	**1286.6219**	7	1314.7362	8	1303.3343	7	1309.5098
RC202	7	1127.7505	7	1116.7943	6	1133.7629	7	**1113.128**	7	1124.9225
RC203	5	948.9644	5	**945.7553**	5	989.5031	5	947.6928	6	946.865
RC204	5	805.8255	5	**791.3987**	5	808.6631	5	808.7256	4	793.3481
RC205	8	1188.7487	7	**1159.4157**	7	1160.1392	7	1175.5916	7	1194.8852
RC206	6	1101.6667	6	**1063.5657**	7	1115.1336	6	1098.6345	6	1094.2777
RC207	6	1018.0232	6	**966.079**	5	988.718	5	1041.8155	6	995.5447
RC208	5	819.6428	4	**780.7162**	6	816.944	5	810.0333	5	808.1717

to 0, this particle is called a disinterested particle, which only learns social experience but not its own experience. In this case, the convergence speed of this particle will become faster. However, at the same time, it will also quickly lose

the population diversity and easily fall into the local optimal position without being able to jump out. Therefore, the value of parameter c is crucial to the performance of the particle swarm algorithm. If the value of c is chosen appropriately, it may prevent the algorithm from falling into the local optimal solution. The standard particle swarm algorithm usually selects $c = 2$. However, in the actual problem solving, for different problems, it may be necessary to select different values of c to improve the performance of the algorithm. For the VRPTW problem, we conducted a parameter sensitivity experiment about c, and the experimental results are shown in Table 7.

Figure 14 visualizes the experimental results based on the data in Table 7. In Figure 14, the horizontal coordinates indicate the number of the benchmark problem, while the vertical coordinates indicate the experimental results corresponding to different values of the parameter c on that problem. Where 1 means that the experimental result is ranked first among all the participating algorithms. 5 means that the experimental result is ranked last among all the participating algorithms. By observing Table 5 and Figure 11, it can be found that when the value of parameter c is 1.5, the performance of ScPSO is the most excellent, and better solutions are obtained in all problems. Therefore, in the simulation experiments, we set the parameter c of ScPSO to a fixed optimal value of 1.5.

VI. CONCLUSION AND FUTURE WORK

A new self-competition particle swarm algorithm (ScPSO) is proposed to solve the VRPTW problem. It controls the updating of particle positions by the particles' self-competition degree and self-learning degree. When a particle updates its position, ScPSO forces the current particle's learning direction based on the particle's self-competitive selection probability. Considering the constraints of VRPTW, ScPSO adopts a stochastic greedy selection heuristic to construct the solution. ScPSO locally fine-tunes the solution through a variable domain search strategy during the local search for the solution. It is found that the ScPSO algorithm demonstrates efficiency and effectiveness in solving the VRPTW problem through comparative experiments on 56 benchmark problems from Solomon 100 customers.

Since the ScPSO algorithm is very flexible in its approach, it can be extended to handle vehicle distribution problems and other combinatorial optimization problems in the future. For example, the ScPSO algorithm can be applied to solve the vehicle path problem (VRP), multi-objective vehicle path problem (MO-VRP), and so on.

REFERENCES

[1] F. Goodarzian, A. Abraham, and A. M. Fathollahi-Fard, "A biobjective home health care logistics considering the working time and route balancing: A self-adaptive social engineering optimizer," *J. Comput. Des. Eng.*, vol. 8, no. 1, pp. 452–474, Jan. 2021.

[2] K. A. Kuznietsov, V. A. Gromov, and V. A. Skorohod, "Cluster-based supply chain logistics: A case study of a Ukrainian food distributor," *IMA J. Manage. Math.*, vol. 28, Jul. 2016, Art. no. dpw009.

[3] N. Tao, H. Yumeng, and F. Meng, "Research on cold chain logistics optimization model considering low-carbon emissions," *Int. J. Low-Carbon Technol.*, vol. 18, pp. 354–366, Feb. 2023.

[4] J. Desrosiers, F. Soumis, and M. Desrochers, "Routing with time windows by column generation," *Networks*, vol. 14, no. 4, pp. 545–565, Dec. 1984.

[5] W. J. A. Kolen, "Vehicle routing with time windows," *Oper. Res.*, vol. 35, no. 1987, pp. 266–273, 1987.

[6] N. Kohl and O. B. G. Madsen, "An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation," *Oper. Res.*, vol. 45, no. 3, pp. 395–406, Jun. 1997.

[7] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis, "2-path cuts for the vehicle routing problem with time windows," *Transp. Sci.*, vol. 33, no. 1, pp. 101–116, Feb. 1999.

[8] E. H. Houssein, D. Oliva, N. A. Samee, N. F. Mahmoud, and M. M. Emam, "Liver cancer algorithm: A novel bio-inspired optimizer," *Comput. Biol. Med.*, vol. 165, Oct. 2023, Art. no. 107389.

[9] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime Mould algorithm: A new method for stochastic optimization," *Future Gener. Comput. Syst.*, vol. 111, pp. 300–323, Oct. 2020.

[10] A.-A.-A. Mohamed, Y. S. Mohamed, A. A. M. El-Gaafary, and A. M. Hemeida, "Optimal power flow using moth swarm algorithm," *Electr. Power Syst. Res.*, vol. 142, pp. 190–206, Jan. 2017.

[11] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114864.

[12] J. Tu, H. Chen, M. Wang, and A. H. Gandomi, "The colony predation algorithm," *J. Bionic Eng.*, vol. 18, no. 3, pp. 674–710, May 2021.

[13] I. Ahmadianfar, A. A. Heidari, S. Noshadian, H. Chen, and A. H. Gandomi, "INFO: An efficient optimization algorithm based on weighted mean of vectors," *Expert Syst. Appl.*, vol. 195, Jun. 2022, Art. no. 116516.

[14] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.

[15] H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, M. Mafarja, and H. Chen, "RIME: A physics-based optimization," *Neurocomputing*, vol. 532, pp. 183–214, May 2023.

[16] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, and H. Chen, "RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method," *Expert Syst. Appl.*, vol. 181, Nov. 2021, Art. no. 115079.

[17] M. W. P. Savelsbergh, "Local search in routing problems with time windows," *Ann. Oper. Res.*, vol. 4, no. 1, pp. 285–305, Dec. 1985.

[18] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[19] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 1983, pp. 671–680, 1983.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1995, pp. 439–444.

[21] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. 1st Eur. Conf. Artif. Life*, vol. 142, 1991, pp. 134–142.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN Int. Conf. Neural Netw.*, vol. 4, Nov. 1995, pp. 1942–1948.

[23] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, "A Tabu search heuristic for the vehicle routing problem with soft time windows," *Transp. Sci.*, vol. 31, no. 2, pp. 170–186, May 1997.

[24] Y. Rochat and É. D. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *J. Heuristics*, vol. 1, no. 1, pp. 147–167, Sep. 1995.

[25] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker, "Localized genetic algorithm for vehicle routing problem with time windows," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5375–5390, Dec. 2011.

[26] H. Li and A. Lim, "Local search with annealing-like restarts to solve the VRPTW," *Eur. J. Oper. Res.*, vol. 150, no. 1, pp. 115–127, Oct. 2003.

[27] H. Yang, E. Wang, Y. Cai, and Z. Sun, "Research on fire rescue path optimization of unmanned equipment based on improved slime mould algorithm," in *Proc. IEEE Int. Conf. Depend., Autonomic Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Sep. 2022, pp. 1–6.

[28] X. Zhang, Q. Liu, and X. Bai, "Improved slime mould algorithm based on hybrid strategy optimization of Cauchy mutation and simulated annealing," *PLoS ONE*, vol. 18, no. 1, Jan. 2023, Art. no. e0280512.

[29] Y. Xiang, Y. Zhou, H. Huang, and Q. Luo, "An improved chimp-inspired optimization algorithm for large-scale spherical vehicle routing problem with time windows," *Biomimetics*, vol. 7, no. 4, p. 241, Dec. 2022.
[30] S. Venkatasubramanian, "Optimal cluster head selection-based hybrid moth search algorithm with tree seed algorithm for multipath routing in WSN," in *Proc. Int. Conf. Netw. Commun. (ICNWC)*, Apr. 2023, pp. 1–7.
[31] M. Alweshah, M. Almiani, N. Almansour, S. Al Khalaileh, H. Aldabbas, W. Alomoush, and A. Alshareef, "Vehicle routing problems based on Harris hawks optimization," *J. Big Data*, vol. 9, no. 1, pp. 1–18, Apr. 2022.
[32] Y. Wang, H. Yang, C. Xu, Y. Zeng, and G. Xu, "An integrated differential evolution of multi-population based on contribution degree," *Complex Intell. Syst.*, vol. 10, no. 1, pp. 525–550, Feb. 2024.
[33] R. J. Kuo, M. F. Luthfiansyah, N. A. Masruroh, and F. E. Zulvia, "Application of improved multi-objective particle swarm optimization algorithm to solve disruption for the two-stage vehicle routing problem with time windows," *Expert Syst. Appl.*, vol. 225, Sep. 2023, Art. no. 120009.
[34] F. Belmecheri, C. Prins, F. Yalaoui, and L. Amodeo, "Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows," *J. Intell. Manuf.*, vol. 24, no. 4, pp. 775–789, Aug. 2013.
[35] N. Norouzi, M. Sadegh-Amalnick, and M. Alinaghiyan, "Evaluating of the particle swarm optimization in a periodic vehicle routing problem," *Measurement*, vol. 62, pp. 162–169, Feb. 2015.
[36] Y. Marinakis, M. Marinaki, and A. Migdalas, "A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows," *Inf. Sci.*, vol. 481, pp. 311–329, May 2019.
[37] S. Amini, H. Javanshir, and R. Tavakkoli-Moghaddam, "A PSO approach for solving VRPTW with real case study," *Int. J. Res. Rev. Appl. Sci.*, vol. 4, no. 3, pp. 118–126, 2010.
[38] Y. W. Zhao, B. Wu, W. L. Wang, Y. L. Ma, W. A. Wang, and H. Sun, "Particle swarm optimization for vehicle routing problem with time windows," *Mater. Sci. Forum*, vols. 471–472, pp. 801–805, Dec. 2004.
[39] Q. Zhang, "Research on particle swarm optimization for grain logistics vehicle routing problem," in *Proc. IITA Int. Conf. Control, Autom. Syst. Eng. (CASE)*, Jul. 2009, pp. 212–215.
[40] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimisation for vehicle routing problem with time windows," *Int. J. Oper. Res.*, vol. 6, no. 4, p. 519, 2009.
[41] Y.-J. Gong, J. Zhang, O. Liu, R.-Z. Huang, H. S. Chung, and Y.-H. Shi, "Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach," *IEEE Trans. Syst. Man, Cybern., C, Appl. Rev.*, vol. 42, no. 2, pp. 254–267, Mar. 2012.
[42] K. C. Tan, L. H. Lee, and K. Ou, "Artificial intelligence heuristics in solving vehicle routing problems with time window constraints," *Eng. Appl. Artif. Intell.*, vol. 14, no. 6, pp. 825–837, Dec. 2001.
[43] X. Yan, L. Tan, J. Chen, and B. Li, "A multi-strategy elite ant system algorithm for vehicle routing problem with time window," in *Proc. IEEE 3rd Int. Conf. Inf. Syst. Comput. Aided Educ. (ICISCAE)*, Sep. 2020, pp. 215–220.

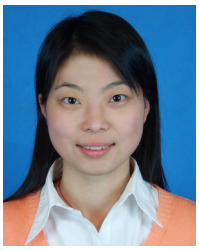
YUFENG WANG received the Ph.D. degree from the Computer School, Wuhan University, Wuhan, China, in 2018. He joined as a Faculty Member with the Software and Computer Science School, Nanyang Institute of Technology, Nanyang, Henan, China, in 2009, where he is currently an Associate Professor. His current research interests include probabilistic modeling of structured data and evolutionary computation.

XIN CHEN is currently pursuing the B.S. degree with the Software and Computer Science School, Nanyang Institute of Technology, Nanyang, Henan, China. His current research interest includes intelligence algorithms.

ZHUO SHUANG is currently pursuing the B.S. degree with the Software and Computer Science School, Nanyang Institute of Technology, Nanyang, Henan, China. His current research interest includes intelligence algorithms.

YING ZHAN received the Ph.D. degree in computer applied technology from Beijing Normal University, Beijing, China. He is currently an Associate Professor with the School of Computer and Software, Nanyang Institute of Technology. His research interests include hyperspectral image classification, machine learning, and deep learning.

KE CHEN received the M.S. degree in computer technology from Northeastern University, in 2013. He is currently a Lecturer with Nanyang Institute of Technology. His research interests include the theory and technology of databases, cloud computing, and data privacy. He is a member of CCF.



CHUNYU XU received the M.S. degree in computer science and technology from the School of Information Engineering, East China University of Technology, Fuzhou, Jiangxi, China, in 2009. She is currently pursuing the Ph.D. degree in artificial intelligence with the School of Electronic Information, Wuhan University. Her research interests include the processing of space exploration data and the application of machine learning to this area.