A neighborhood comprehensive learning particle swarm optimization for the vehicle routing problem with time windows

Qichao Wu[a,b], Xuewen Xia[a,b,*], Haojie Song[a,b], Xing Xu[a,b], Yinglong Zhang[a,b], Fei Yu[a,b], Hongrun Wu[a,b], Hui Zeng[c]

[a]*College of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, Fujian, China*
[b]*Key Lab of Intelligent Optimization and Information, Minnan Normal University, Zhangzhou 363000, Fujian, China*
[c] *XinJiang Institute of Engineering, Urumqi 830091, Xinjiang Uygur Autonomous Region, China*

Abstract

Vehicle routing problem with time windows (VRPTW), which is a typical NP-hard combinatorial optimization problem, plays an important role in modern logistics and transportation systems. Although the particle swarm optimization (PSO) algorithm exhibits very promising performance on continuous problems, how to adapt PSO to efficiently deal with VRPTW is still challenging work. In this paper, we propose a neighborhood comprehensive learning particle swarm optimization (N-CLPSO) to solve VRPTW. To improve the exploitation capability of N-CLPSO, we introduce a new remove-reinsert neighborhood search mechanism, which consists of remove operator and reinsert operator. When performing the remove operator, the probability of adjacency between two customers is calculated by an information matrix (IM), which is constructed based on the customers' time-space information and elite individuals' local information. When executing the reinsert operator, the IM and a cost matrix (CM), which is introduced to record the cost of customer insertion, are used to find an optimal insert position. Moreover, to enhance the exploration of N-CLPSO, a semi-random disturbance strategy is proposed to prevent degradation of the population, in which the longest common sequences (LCS) of elites are saved. The N-CLPSO algorithm is tested on 56 Solomon benchmark instances, and the algorithm obtained yielded 22 optimal solutions. The simulation results and comparison results illustrate that the proposed algorithm outperforms or can compete with the majority of other 3 PSO variants as well as other 9 state-of-the-art algorithms.

Keywords: Vehicle routing problem with time windows, Particle swarm optimization, Neighborhood search, Longest common sequence

1. Introduction

For logistics industries, the rise of e-commerce means more orders and parcels to be processed [1]. Thus, it has become a research focus of logistics issues that ensuring goods are delivered within a specified time with a lower cost. Based on in-depth analyses of the logistics problem, one can observe that study of Vehicle Routing Problems (VRP) is a feasible and

*Corresponding author

promising method to deal with the problem. Generally, a good VRP solution can consider many aspects, such as complete customer service on time, reduce distribution costs while improving customer satisfaction, and so on. Therefore, designing an efficient and reliable algorithm to deal with VRP problems has become a key research priority in logistics and distribution.

The VRP, which is a typical combinatorial optimization problem, was first proposed by Dantzig et al. in 1954 when they studied the optimal path of gasoline transportation trucks [2]. Since there is a high correlation between the VRP and the reality of logistics, it has attracted much attention of researches in the logistics field. During the last few years, various VRP variants, such as the Capacitated VRP (CVRP) [3] and the Heterogeneous Fleet VRP (HFVRP) [4], have proposed aiming to imitate different real applications. One of the most significant variations is the VRP with time windows (VRPTW) [5], in which users' access time and vehicles' capacity limits are considered in VRP aiming to make the problem more realistic.

Initially, some exact algorithms are adopted to solve VRPTW [6, 7]. However, due to VRPTW's NP-hardness, solving large-scale VRPTW with the exact algorithms is exceedingly time-consuming. Thus, in recent years, various heuristic and meta-heuristic algorithms for solving VRPTW problems are of great interest to researchers. For example, the Simulated Annealing (SA) [8], Taboo Search (TS) [9], Ant Colony Optimization (ACO) [10], Particle Swarm Optimization (PSO) [11], and Genetic Algorithm (GA) [12] have been proved to be effective in solving VRPTW problems. However, as the complexity and scale of VRPTW increase, the convergence speed and optimal results of the algorithms are still unsatisfactory. Hence, how to speed up the convergence and improve the solutions' accuracy needs to be further studied.

PSO algorithm [13], proposed by Kennedy and Eberhart in 1995, has been widely used for continuous function optimization and achieved many promising achievements in both theoretical studies [14, 15, 16, 17, 18] and engineering applications [19, 20, 21, 22]. However, PSO also has some shortcomings, including (1) premature convergence for complicated multimodal problems, and (2) low precision of final solutions. The main reasons for the former shortcoming of the PSO algorithm is that the population diversity may disappears rapidly during optimization process. Hence, many improvements intending to keep the population diversity have been proposed by researchers [23, 24]. To overcome the latter drawback of the PSO algorithm, various excellent and efficient local search strategies have been applied to some PSO variants [25, 26]. Although these strategies significantly enhance the comprehensive performance of PSO on continues problems, they cannot be directly applied in combinatorial optimization problems, such as VRPTW. Thus, in recent years, how to improve these strategies in PSO and adapt it to VRPTW attract researchers' attention.

Based on the analysis above mentioned, this paper proposes a neighborhood comprehensive learning PSO (N-CLPSO) algorithm for solving the VRPTW. The motivation and novelties applied in N-CLPSO are briefly introduced as follows.

(1) The original CLPSO is mainly used to solve continuous problems. To adapt the N-CLPSO to solve VRPTW, we redefine the velocity and position of the particles and corresponding update rules of them [11].

(2) Regarding the convergence performance of the algorithm, we improve the velocity update strategy [27] of the original CLPSO using a method that determines the learning probability and learning exemplars based on their fitness values. In addition, we also use the vehicle insertion strategy to reduce the number of vehicles as much as possible, and to speed up the algorithm's convergence.

(3) Although some local information of populations can be used to guide the evolution of populations [28, 29], to our knowledge, no studies directly apply local information of nodes to

insertion search. Therefore, we propose the concept of a local information matrix that not only considering local information about the customers location, time windows, and elite individuals in each generation, but also evaluates the incremental cost caused by the insertion operator.

(4) To overcome degradations caused by blind random perturbations, we propose a semi-randomly perturbation strategy based on the longest common subsequence (LCS) idea. Based on the strategy, some elite sequences in LCS can be preserved while other unpromising sequences are destroyed. Thus, the diversity of the particles can be enhanced, and the solutions quality can be improved.

The rest of this paper is organized as follows. Section 2 provides a short introduction to the PSO algorithm and the VRPTW model, and Section 3 reviews the study of VRPTW. N-CLPSO and new introduced strategies are detailed in Section 4. Next, the experimental results and analyses are reported in Section 5. Finally, a summary and future work of this study are presented in Section 6.

2. Related work

2.1. PSO

In the standard PSO, states of each particle i in the t^{th} generation can be described by two vectors, i.e., a position vector $X_i^t = \left[x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,D}^t\right]$ and a velocity vector $V_i^t = \left[v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,D}^t\right]$, where

D denotes a dimension of the problem to be optimized. X_i^t is considered as a candidate solution, and V_i^t is regarded as the search direction and step size of particle i in the t^{th} generation. In the process of population search, each particle adjusts its flight path by its own historical best position $PB_i^t = \left[pb_{i,1}^t, pb_{i,2}^t, \ldots, pb_{i,D}^t\right]$ and the population's historical best position $GB_i^t = \left[gb_{i,1}^t, gb_{i,2}^t, \ldots, gb_{i,D}^t\right]$. The specific update rules of V_i^t and X_i^t are defined as Eq.(1) and Eq.(2), respectively.

$$v_{i,j}^{t+1} = wv_{i,j}^t + c_1 r_1 \left(pb_{i,j}^t - x_{i,j}^t\right) + c_2 r_2 \left(gb_j^t - x_{i,j}^t\right) \tag{1}$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1} \tag{2}$$

where w denotes an inertia weight, which is used to control the influence of the current velocity on the latest velocity; c_1 and c_2 are two constants that determine the learning weight on PB_i^t and GB_i^t respectively; r_1 and r_2 are two random numbers uniformly distributed in the interval [0, 1].

To improve the global search capability of the basic PSO, Liang et al. [30] proposed a CLPSO in which a new velocity update rule described as Eq.(3) is used to prevent premature convergence.

$$v_{i,j}^{t+1} = wv_{i,j}^t + c_1 r_1 \left(p_{f(i),j}^t - x_{i,j}^t\right) \tag{3}$$

where $f(i)$ represents a particle index that guides particle i to fly in the j^{th} dimension, and the particle $f(i)$ can be any particle including particle i itself. To select $f(i)$ in each dimension, CLPSO will generate a random number r. Subsequently, r is compared with Pc_i defined as Eq.(4), which is the learning probability of the control particle to learn from itself or others.

$$Pc_i = 0.05 + 0.45 * \frac{\left(\exp\left(\frac{10(i-1)}{N-1}\right) - 1\right)}{(\exp(10) - 1)} \qquad i = 1, 2, \ldots, N \tag{4}$$

where N is the population size.

If r is greater than Pc_i, the particle i learns toward its own personal history. On the contrary, the particle i selects the particles $f(i)$ as its learning exemplar by binary tournaments selection. Using this learning strategy, particles can learn not only from themselves, but also from the optimal features of other particles. These features allow the particles to have more learnable samples and a larger potential flight space. Thus, CLPSO can utilize the helpful information in the population more efficiently, and then can generate higher quality solutions. Experimental results in [31] manifest that CLPSO has good performance for discrete optimization. Hence, we will use CLPSO as the basic framework to solve VRPTW problems.

2.2. Mathematical definition of VRPTW

VRPTW is to find the lowest cost route to serve multiple consumers with the same size fleet within a certain time window. The total demand for service provided by each vehicle must not exceed the total capacity of the vehicle, and each customer is served by a vehicle only once during a defined time window. In other words, a vehicle must wait until the start of the time window if it approaches a customer before the start of the customer's time window. Similar to this, a customer cannot be served if a vehicle arrives at their location after the end of their time window.

VRPTW is a problem in which a fleet of K vehicles serve M customers. Each vehicle has a constant capacity Q. The depot v_0 is the start and end point of each route. The vertex v_i is defined as a customer, $i \in \{1, 2, \ldots, M\}$. The demand of customer v_i located at (x_i, y_i) is q_i and the delivery time window of it is $[b_i, l_i]$, where b_i and l_i refer to the earliest and latest time when the customer starts the service, respectively. If a vehicle arrives at the customer v_i earlier than b_i, it must wait until the start of the time window to serve the customer. On the other hand, if the vehicle does not arrive before l_i, it cannot serve the customer v_i. The service time of each customer is s_i. The depot is located at (x_0, y_0) with demand $q_0 = 0$ and the time window $[0, l_0 \geq \max(b_i)]$. For simplicity, the time cost that a vehicle traveling from customer i to customer is represented by the Euclidean distance between nodes $\left(c_{i,j} = c_{j,i}\right)$, where $i \neq j$, $i, j \in \{1, 2, \ldots, M\}$.

Generally, there are two objectives in VRPTW, which are defined as Eq.(5) and Eq.(6), respectively. The primary goal is to minimize the number of vehicles (NV) and the secondary goal is to minimize the total distance (TD) with the same number of routes. VRPTW can be mathematically formulated as follows.

The goal of the VRPTW is to minimize

$$\min NV = K \tag{5}$$

and

$$\min TD = \sum_{i=0}^{M} \sum_{j=0}^{M} \sum_{k=1}^{K} c_{i,j} * x_{i,j}^{k} \tag{6}$$

s.t.

$$\sum_{i=0}^{M} x_{i,j}^k = y_j^k \quad \forall k = 1, \ldots K, \quad \forall j = 1, \ldots, M \tag{7}$$

$$\sum_{i=0}^{M} x_{i,j}^k = y_i^k \quad \forall k = 1, \ldots K, \quad \forall i = 1, \ldots, M \tag{8}$$

$$\sum_{k=1}^{K} y_i^k = 1 \quad \forall i = 1, \ldots M \tag{9}$$

$$\sum_{i=0}^{M} y_i^k * q_i \leq Q \quad \forall k = 1, \ldots K \tag{10}$$

$$\sum_{k=1}^{K} y_0^k = K \tag{11}$$

$$t_i + w_j + s_i + c_{i,j} = t_j \quad \forall i, j = 1, \ldots M, i \neq j \tag{12}$$

$$b_j \leq t_j \leq l_j \quad \forall j = 1, \ldots M \tag{13}$$

$$w_j = \max\left\{b_i - (t_i + s_i + c_{i,j}), 0\right\} \quad \forall i = 1, \ldots M \tag{14}$$

where

$$x_{i,j}^k = \begin{cases} 1, \text{ if vehicle } k \text{ treavels directly from } i \text{ to } j \\ 0, \text{ otherwise} \end{cases}$$

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

Constraints Eq.(7)-Eq.(9) mean that each customer will be served by a vehicle and that each customer can be served by only one vehicle. Constraint Eq.(10) means that each vehicle cannot carry more than capacity Q. Constraint Eq.(11) means that all routes start from the depot. E-qs.(12)-(14) define the time window constraint, where t_i is the vehicle arrival time at node i; w_j is the waiting time after the vehicle arrives at customer j; S_i is the service time; and $c_{i,j}$ is the time cost between nodes i and j.

3. Literature review

VRP and its variants have been extensively studied based on different intelligence algorithms in the past decades. For instance, ACO inspired by the foraging behavior of real ants is a popular probabilistic algorithm to solve VRPTW. Considering the customer service time, Wang [32] et al. designed a multi-ant system with local search, which combines the Multi_Ant System algorithm and four local search operators to improve the solution quality. Gupta [33] et al. proposed an improved ACO to solve the VRPTW problem, which uses new pheromones to reset and update the function to enhance the global search capability of the algorithm, and improve the optimization path by the 2-opt method. Zhang [34] et al. designed a solution strategy based on ACO and three variational operators to solve a multi-objective VRP problem with flexible time windows.

GA is a heuristic algorithm that simulates genes, chromosomes and the genetic evolution of organisms. Due to its strong search performance and good extensibility, GA has been widely used in VRPTW. Zhang [35] et al. used VRPTW as a research object and proposed a hybrid multi-objective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search (HMOEA-GL). Khoo [12] et al. proposed an two-phase distributed ruin-and-recreate genetic algorithm (RRGA) to solve VRPTW by a ruin-and-recreate strategy combined with GA. This combination of algorithms harnesses the strength of the search diversification and intensification, thereby producing very high-quality solutions.

PSO, as a typical swarm intelligence optimization algorithm, imitates the foraging behaviour of a flock of birds, and it was mainly applied to continuous problems at the beginning of its proposal. In recent years, some scholars started to improve the standard PSO to make it applicable to the optimization of combinatorial optimization problems. Reong [36] et al. summarized the last 20 years of PSO algorithms for solving VRP and related variants, including solving VRPTW. Saksuriya [37] et al. proposed a novice local search, ruin and recreated procedure, and PSO three-module hybrid heuristic algorithm to solve VRPTW. Salehi [38] et al. proposed a multi-objective PSO and a multi-objective particle swarm variable neighborhood search algorithm to solve the real-time collaborative feeder vehicle path problem (RTCFVRP) with flexible time windows and achieved better performance. Ding [39] et al. established a time-window electric vehicle distribution path problem (EVRPTW-DC) for driving cycles based on typical driving cycles in suburban and urban areas, considering vehicle load, driving distance, and speed. Then an adaptive PSO was designed to solve the problem.

In studies on VRP-related variants, the local neighborhood search strategy has become a critical factor in determining individuals to jump out of the local optimum. Therefore, designing efficient neighborhood search strategies has received much attention from researchers recently. For instance, Liu [40] et al. designed an efficient algorithm based on the combination of large-neighborhood search and GA. The algorithm uses a constrained relaxation scheme to extend the search space by neighborhood search of existing infeasible solutions. It initiates a GA to explore the undiscovered space when the search falls into a local optimum. Yu [10] et al. introduced a neighborhood search operator in the ACO and protect the population diversity by forbidden search. Alinaghian [41] et al. presented a mathematical model for a hybrid green vehicle routing problem (TD-FSMGVRP). This study utilizes an improved adaptive large domain search algorithm to solve the TD-FSMGVRP problem simultaneously considering multiple dimensions of vehicle fixed cost, driver cost, fuel cost, and greenhouse gas emission cost and achieves superior performance.

4. Proposed method

In this section, Section 4.1 presents the framework of N-CLPSO, and Section 4.2 describes the details of it. Section 4.3 introduces the vehicle insertion strategy. Section 4.4 shows the guided reinsertion operator based on local information, and Section 4.5 details the remove-reinsert-based neighborhood search strategy. Diversity retention strategies based on elite fragments are described in Section 4.6.

4.1. Framework of N-CLPSO

N-CLPSO is a combination of CLPSO and a new proposed local search strategy. Although, CLPSO has a good global search capability when optimizing continues problems, the encoding and related operators of it need to be redefined according to the distinct characteristics of VRPTW.

We use the vector X_i^t to denote the position of particle i in the t^{th} generation. $x_{i,d}^t$ denotes the set $d_{k,l} = \{< k, d >, < d, l >\}$ of a set of arcs in the d-dimension of X_i^t, which indicates that the left and right neighbors of node d in particle i in the t^{th} generation are k and l, respectively. To ensure that each position is a valid solution, the position $x_{i,d}^t$ has three constraints: (1) $d \in (0, 1, 2, \dots, n)$, n represents the city number; (2) $k, l \in \{0, 1, \dots, d-1, d+1, \dots, n\}$ and (3) $k \neq l$. The constraint (1) ensures that each element is a valid city, the constraint (2) makes each city will not be adjacent to itself, and the constraint (3) enables that each city's neighbouring points are not duplicated. Taking a VRPTW with four cities as an example, a route sequence $0 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 \rightarrow 0$, can be represented by 5 arcs: $(0_{1,2}, 1_{3,0}, 2_{0,4}, 3_{4,1}, 4_{2,3})$. Meanwhile, there exists a set of probability sets $v_{i,d}^t = [< u, v > /p(u, v) \mid < u, v > \in A_d]$ in dimension d of the velocity vector V_i^t, where A_d denotes the set of all possible adjacent arcs to node d and $p(u, v) \in [0, 1]$ is the probability corresponding to each arc $< u, v >$. The framework of N-CLPSO is demonstrated by Fig.1

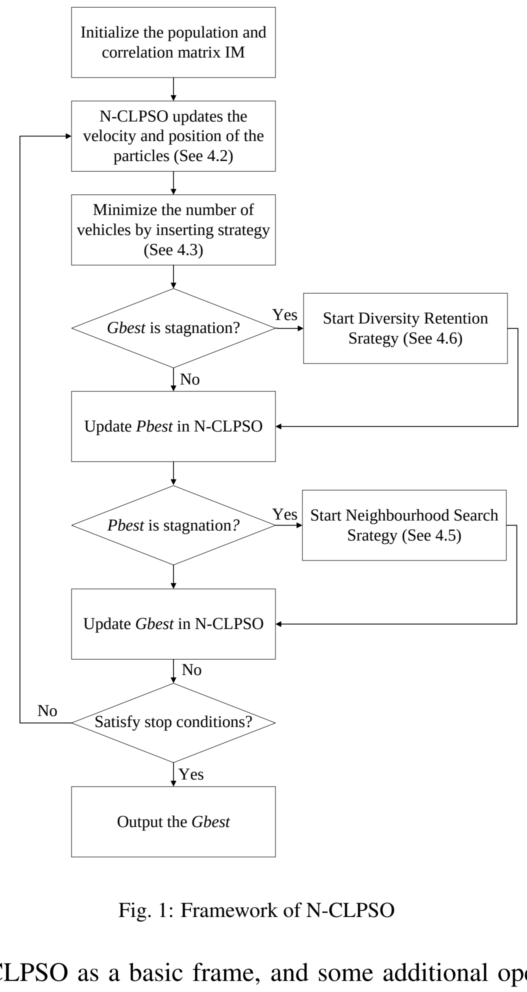
Initialize the population and correlation matrix IM
N-CLPSO updates the velocity and position of the particles (See 4.2)
Minimize the number of vehicles by inserting strategy (See 4.3)
Gbest is stagnation?
Yes
Start Diversity Retention Srategy (See 4.6)
No
Update *Pbest* in N-CLPSO
Pbest is stagnation?
Yes
Start Neighbourhood Search Srategy (See 4.5)
Update *Gbest* in N-CLPSO
No
No
Satisfy stop conditions?
Yes
Output the *Gbest*

Fig. 1: Framework of N-CLPSO

N-CLPSO selects CLPSO as a basic frame, and some additional operators are introduced, i.e., a vehicle insertion strategy (see Section 4.4), a diversity retention strategy (see Section 4.6), and a neighborhood search strategy (see Section 4.5). After the N-CLPSO velocity and position update (see Section 4.2), we use a vehicle insertion strategy to minimize the number of vehicles in a feasible solution. Then, the diversity retention strategy and the neighborhood search strategy are executed while *Gbest* and *Pbest* are stagnant, respectively. After that, the global optimal solution of the population *Gbest* is updated. The above steps are repeatedly executed until the stopping conditions are satisfied.

It has been pointed out in Section 2 that VRPTW has two objectives. In this study, we combine a new decision method [11] to deal with the number of vehicles NV and the total distance TD of VRPTW. To better present the priority of the NV objective over the TD objective, we normalize TD in the range of [0,1] weighted with NV. The objective function of N-CLPSO is defined as Eq.(15), where $NV\left(X_i^t\right)$ and $TD\left(X_i^t\right)$ denote the number of vehicles and the total distance corresponding to particle X_i, respectively.

$$\text{fitness}\left(X_i^t\right) = NV\left(X_i^t\right) + \text{ normalize }\left(TD\left(X_i^t\right)\right) \tag{15}$$

$$\text{normalize}(x) = \frac{\arctan(x)}{\frac{\Pi}{2}} \tag{16}$$

4.2. Basic operators in N-CLPSO

In N-CLPSO, based on the above definition of position and velocity [11], we use Eq.(3) to update the velocity. The related operators in Eq.(3) are defined as Eqs.(17)-(20, respectively. $c * v_{i,d}^t$ and $v_{i,d}^t + v_{j,d}^t$ denotes the probability $p(u, v)$ variation of the arc. $x_{i,d}^t - x_{j,d}^t$ means the set solving difference set and $c * (x_{i,d}^t - x_{j,d}^t)$ stands for converting a crisp set into a set with probability:

$$\begin{gathered} c * v_{i,d}^t = \{< u, v > /p'(u, v) \mid < u, v >\in A_d\} \\ p'(u, v) = \begin{cases} 1, \text{ if } c * p(u, v) > 1 \\ c * p(u, v), \text{ otherwise} \end{cases} \end{gathered} \tag{17}$$

$$v_{i,d}^t + v_{j,d}^t = \left\{< u, v > \left|\max\left(p_i(u, v), p_j(u, v)\right)\right| < u, v >\in A_d\right\} \tag{18}$$

$$x_{i,d}^t - x_{j,d}^t = \mathrm{U}_d = \left\{< u, v >\mid < u, v >\in x_{i,d}^t \text{ and } < u, v >\notin x_{j,d}^t\right\} \tag{19}$$

$$\begin{gathered} c * U_d = \{< u, v > /p'(u, v) \mid < u, v >\in A_d\} \\ p'(u, v) = \begin{cases} 1, \text{ if } < u, v >\in U_d \text{ and } c > 1 \\ c, \text{ if } < u, v >\in U_d \text{ and } 0 < c < 1 \\ 0, \text{ if } < u, v >\notin U_d \end{cases} \end{gathered} \tag{20}$$

For instance, we assume that $v_{i,1}^t = \{< 1, 2 > /0.3, < 1, 4 > /0.5, < 4, 1 > /0.6\}$, $x_{i,1}^t = \{< 5, 1 >, < 1, 2 >\}$, $p_{f(i),1}^t = \{< 1, 4 >, < 5, 1 >\}$, $w = 0.4$, $c_1 = 2.0$, $r_1 = 0.3$. Then, we have $w * v_{i,1}^t = \{< 1, 2 > /0.12, < 1.4 > /0.2, < 4, 1 > /0.24\}$ and $p_{f(i),1}^t - x_{i,1}^t = \{< 1, 4 >\}$ and $c_1 * r_1 * \left(p_{f(i),1}^t - x_{i,1}^t\right) = \{< 1, 4 > /0.6\}$. Finally, the new velocity $v_{i,1}^t = w * v_{i,1}^t + c_1 * r_1 * \left(p_{f(i),1}^t - x_{i,1}^t\right) = \{< 1, 2 > /0.12, < 1, 4 > /0.6, < 4, 1 > /0.24\}$ can be obtained.

In order to speed up the algorithm's convergence, this paper applies a new velocity update strategy [27], which sets the learning probability according to the particle's fitness and selects different learning samples according to the characteristics of the particle itself, show as Eq.(21) and Eq.(22).

$$Pc_i = \frac{sc_i}{2 * N} \tag{21}$$

$$n = 2 + \text{round}\left(\frac{\text{ceil}\left(\frac{N}{2}\right) - 2}{N * sc_i}\right) \tag{22}$$

where N is the size of population, sc_i is the ranking of particle i in terms of its fitness value in the population, and n is the number of selected exemplars.

Unlike the traditional CLPSO, in which a particle selects its learning exemplar based on its index, a particle in N-CLPSO adjusts its learning probability and the number of learning exemplars based on its fitness values. From Eq. (21) and Eq. (22), we can observe that the learning probability Pc and the number of selected exemplars can be adapted according to a particle's fitness and the size of the population. Concretely, the higher fitness a particle has, the smaller the Pc and the n will be. Thus, the elite particles can pay more attention to self-learning, which is beneficial for keeping their promising property. On the contrary, if the individual fitness value is lower, the Pc and the n will be larger, so the particle has a greater probability of learning from other particles. Hence, the inferior particles have more chances to obtain favorable knowledge from other particles.

In N-CLPSO, the original position update process in CLPSO is replaced by a new process, which is detailed in Algorithm 1. The position of the particle can be obtained by three set: $S_V = \{m \mid <k,m> \in V_i$, and $<k,m>$ satisfies $\Omega\}$, $S_x = \{m \mid <k,m> \in X_i$, and $<k,m>$ satisfies $\Omega\}$, $S_a = \{m \mid <k,m> \in A$, and $<k,m>$ satisfies $\Omega\}$. It can be seen that the arcs $<k,m>$ come from the sets V_i, X_i and A respectively, while satisfying the constraints. We set a random number $r \in [0,1]$ in order to ensure that arcs with larger probability are more likely to be selected, and only arcs with probability greater than r in V_i^t will be added to V_i. Then, the set X_i and A represent the set of all possible arcs from the current position X_i^t and the search space.

As in Algorithm 1, the new position X_i^{t+1} is set to the empty set before the position update. According to the constraint, each vehicle departs from the depot and iteratively selects the next customer to be served. Suppose, k is the customer being served by the vehicle and the next customer m to be visited by the vehicle. If there is an available node in S_V, we select m from S_V. Otherwise, we select m from S_x. When there is no available node in both S_V and S_x, we select m from S_a. After m is selected, arc $<k,m>$ is added to X_i^{t+1}. The search process is repeatedly until all customers have been visited.

When there are no available nodes in the sets S_v, S_x, and S_a, the constraints of VRPTW cannot be satisfied. Hence, a new path needs to be created, i.e., a new sub-tour needs to be opened. Specifically, a depot node needs to be inserted after the current customer point k, and the next customer point m to be served is reselected using the depot as the starting point, thus ensuring the feasibility of X_i^{t+1}. Finally, the updated X_i^{t+1} will replace the current position X_i^t. In addition, we also use the heuristic selection method NNH [11] to speed up the convergence of the algorithm.

Algorithm 1 Position update in N-CLPSO

```
Input: $X_i$; $V_i$; $A$;
Output: $X_i^{t+1}$
1:  $X_i = \phi$;$k = 0$;
2:  $S_V = \{m \mid < k, m > \in V_i$, and $< k, m >$ satisfies $\Omega\}$;
3:  $S_x = \{m \mid < k, m > \in X_i$, and $< k, m >$ satisfies $\Omega\}$;
4:  $S_a = \{m \mid < k, m > \in A$, and $< k, m >$ satisfies $\Omega\}$;
5:  while Customers do not have complete access do:
6:      if $S_V \neq \phi$ then
7:          select m in $S_V$, and add $< k, m >$ to $X_i^{t+1}$
8:          $k = m$;
9:          update $S_V$,$S_x$,$S_a$;
10:     else if $S_x \neq \phi$ then
11:         select $m$ in $S_x$, and add $< k, m >$ to $X_i^{t+1}$
12:         $k = m$;
13:         update $S_V$,$S_x$,$S_a$;
14:     else if $S_a \neq \phi$ then
15:         select m in $S_a$, and add $< k, m >$ to $X_i^{t+1}$
16:         $k = m$;
17:         update $S_V$,$S_x$,$S_a$;
18:     else
19:         $k = 0$;
20:         update $S_V$,$S_x$,$S_a$;
21:     end if
22: end while
```

4.3. Vehicle insertion strategy

It is well known that the number of vehicles is one of crucial factors determining a VRP's difficulty. In reality, each additional vehicle is likely to increase the cost significantly. Therefore, in this paper, a simple insertion scheme is applied to reduce the number of vehicles after each particle completes its position update. The vehicle insertion strategy is shown in Algorithm 2.

Take Fig.2 as an example, where "1" is a depot and "2"-"7" are 6 customers. The route $r = \{1, 2, 3, 1, 5, 6, 1, 4, 7, 1\}$ means that there are 3 sub-tours, i.e., $Vc(1) = \{1, 2, 3, 1\}$, $Vc(2) = \{1, 5, 6, 1\}$, and $Vc(3) = \{1, 4, 7, 1\}$. First, we remove sub-tour $Vc(1)$ from route r to obtain an intermediate route r^*. After that, the nodes $\{2, 3\}$ in $Vc(1)$ need to be inserted into the r^* through a guided insertion strategy (see Section 4.4). When all nodes are inserted successfully, as shown in Fig.2(a), a new route r consisting 2 new sub-tours $Vc(1)$ and $Vc(2)$ can be obtained, and continue to remove the path $Vc(1)$. If any node is not inserted in the r^* as shown in Fig.2(b), the route keeps the sub-tour in the state before insertion, and removes path $Vc(2)$. Such operations are repeated until all sub-tours being visited. The time complexity of the vehicle insertion strategy is directly related to the number of sub-tours K. If the insertion time is m, the time complexity of this strategy is $O(K * m)$.

Algorithm 2 Vehicle insertion strategy

Input: Routing r, Vehicle Collection Vc

Output: New Routing Nr

```
1:  $n = number(Vc)$; //Record total number of vehicles
2:  $i = 1$;
3:  while $i <= n$ do:
4:      $ins = Vc(i)$; //Add the node with vehicle $i$ to the set to be inserted
5:      $r^* = r - Vc(i)$; //Remove the vehicle $i$ from the path r
6:      Insert $ins$ into $r^*$ using Algorithm 3;//See section 4.4
7:      if $ins$ all inserted successfully then
8:          $r = r^*$;
9:          $n = n - 1$;
10:     else
11:         $r = r$;
12:         $i = i - 1$;
13:     end if
14: end while
15: $Nr = r$;
```

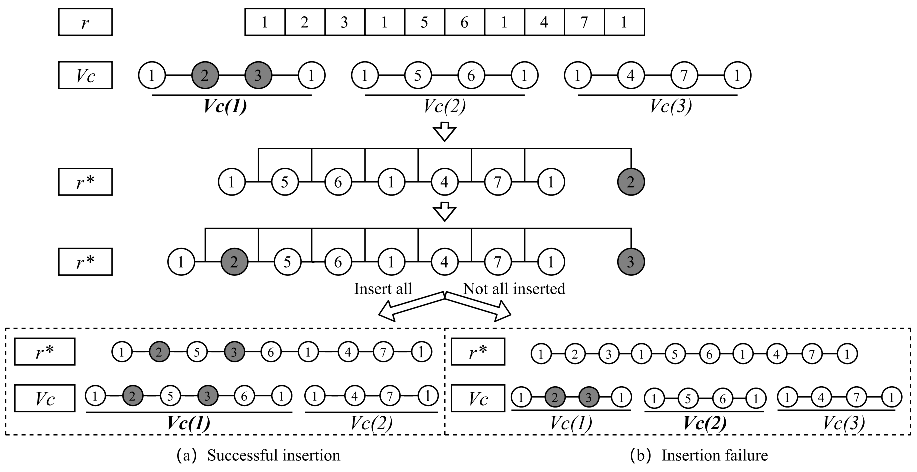

Fig. 2: Vehicle insertion diagram

4.4. *Guided reinsertion operator based on local information*

The VRPTW problem itself has a very complex time-space distribution, where the customer points are not only distributed in different locations in space (i.e., spatial distribution characteristics), but also have their distinct time windows (i.e., temporal distribution characteristics). If spatial locations of two customs are close, but the time windows of them are very different, directly connecting the two customs in a route may result in a longer waiting time for a vehicle, which makes the quality of the solution degrade. On the contrary, if the time windows of two customers are close, but the distances of them are far, infeasible solutions may be generated if the two customers are severed by a same vehicle. Therefore, it is necessary to consider both time and space factors when solving VRPTW.

In this section, a guided reinsertion operator based on local information is proposed. To create a local information matrix, the space-time distribution characteristics between customer points, elite segment information, and insertion cost are considered. Then, we go through the local information matrix to guide a customers to reinsert into a path. In this study, the local information matrix is mainly divided into two modules: one is the customer information matrix (IM), and the other is the route cost matrix (CM) that rises after customer insertion. Details of the two modules are introduced as follows.

4.4.1. Information matrix

Inspired by the study in [28], the local information among customers can be utilized to quickly calculate the probability of adjacency between two customers, and then to obtain a higher quality solution set after the crossover operation. Therefore, the IM proposed in this work is constructed based on the time-space distribution characteristics of nodes and information on elite segments of particles in N-CLPSO. Concretely, $IM^t_{i,j}$, defined as Eq.(23), denotes the probability that customers i and j can be served by the same car consecutively.

$$IM^t_{i,j} = (1-\beta_t) * \left(DST_{\max} - DST_{i,j}\right) + \beta_t * CT \tag{23}$$

From Eq.(23) we can see that $IM^t_{i,j}$ contains two components. The first component is $DST_{\max} - DST_{i,j}$, where $DST_{i,j}$, detailed as Eq. (24), represents the distance in time-space between customer i and customer j. $DT_{i,j}$ and $Dis_{i,j}$ denote the time and space distances between customer points i and j, respectively. It is obvious that the latter can be expressed in terms of the Euclidean distance between two points. The size of the time window tends to be more likely to reflect the probability that two different customer points can be served by the same vehicle. Generally, the probability of two points being served by the same vehicle will decrease as an interval between the two time windows is very small. For instance, in the condition, the vehicle is likely to exceed the latest time window constraint for the latter node while finish the serve of the former node. To avoid the problem, intuitively, we want the vehicle to have plenty of time to serve the latter customer. Therefore, we utilize the idea in [42] to select the next serve customer. Concretely, we measure the time distance based on the amount of time saved when the vehicle arrives before the end of the time window.

$$DST_{i,j} = (1-\mathrm{a}) * \frac{\left(DT_{i,j} - \min(DT)\right)}{\max(DT) - \min(DT)} + \mathrm{a} * \frac{\left(Dis_{i,j} - \min(Dis)\right)}{\max(Dis) - \min(Dis)} \tag{24}$$

Suppose there exists customer i and customer j, time windows of them are $[a,b]$ and $[c,d]$, respectively, and k_1 and k_2 denote the cost coefficients of the remaining service time and waiting time of the vehicle, respectively, then the vehicle-saving time can be found according to the time t' of the vehicle arriving from i to j, as in Eq.(25).

$$S_{i,j} = \begin{cases} k_1 * (d-c) - k_2 * (c-t'), t' < c \\ -k_1 * t' + k_1 * d, c \le t' \le d \\ -\infty, t' > d \end{cases} \tag{25}$$

When the vehicle arriving early at customer j, the vehicle needs to wait until the customer starts service time c. Therefore, the time saved $S_{i,j}$ is equal to the length of the customer's time window minus the time the vehicle is waiting. If the vehicle arrives within the time window at point j, the saving time $S_{i,j}$ is equal to the end time window j subtracting the arrival time $t^{'}$. If the vehicle arrival time exceeds the end time d of the customer, the customer cannot be served. We can see that it is easier for customer i to go to customer j if $S_{i,j}$ is larger. To be consistent with the spatial distance, we define the time distance between two customers denoted as Eq.(26).

$$DT_{i,j} = k_1 \max(S) - S_{i,j} \tag{26}$$

The second component in the right side of Eq.(23) is CT, which represents the local information (see Eq.(27)) of the elite individuals.

$$CT = \frac{ct - ct_{\min}}{ct_{\max} - ct_{\min}} \tag{27}$$

where ct denotes the total number of times that customer i and customer j are adjacent to each other from the beginning to the current generation; ct_{max} and ct_{min} denote the maximum and minimum values of the number of times that all customers are adjacent to each other, respectively.

In evolutionary algorithms, individuals with better fitness values are more likely to produce superior offspring because that the elite individuals seem more likely to be closer to the optimal solution. And common fragments of these elite individuals are more likely to be part of the optimal solution. Therefore, if two adjacent customers frequently appear in different elite particles, the probability of their being served by the same vehicle will also be high in the optimal solution.

In the iterative process of IM, the good genes of elite individuals should gradually spread to the whole population. However, too fast a propagation of good genes often leads to premature algorithm convergence. Conversely, too slow a propagation of good genes may cause a poor solution accuracy. Therefore, a linearly decreasing coefficient β_t defined in Eq.(28) is introduced to adjust the diffusion rate of the superior genes.

$$\beta_t = \frac{t}{t_{\max}} \tag{28}$$

where t is the current number of generations and t_{max} is the maximum number of generations.

It can be seen from Eq.(28) that at the early stage of algorithm optimization, a smaller β_t is conducive to preserving the population diversity and enhancing the global search ability of the population. On the contrary, at the later evolutionary stage, a greater β_t enables the spread of excellent genes to be faster, which is beneficial to the algorithm's convergence.

As mentioned above, a larger $IM^t_{i,j}$ indicates that the probability of customer i and customer j being served by the same car in the t^{th} generation is larger. It is worth noting that the size of IM is determined by the number of customers. Concretely, when the number of customers is N, the size of IM is $N * N$. Thus, to reduce the computational cost, we only update IM when the global optimal solution is updated.

4.4.2. Cost matrix

To speed up the convergence, we also propose a CM-assisted information matrix for bootstrap repair. CM finds the best insertion position in a path with the help of greedy ideas, i.e., the insertion brings the least increase in total cost afterwards. As shown in Fig.3, there exists a point i to be inserted into a path r with length $L + 1$. In the insertion process, we not only need to determine whether node i satisfies a constraint after insertion, but also need to calculate the corresponding incremental cost. Note that, if the constraint being violated after node i has been inserted into a point in the path, the corresponding cost increment is ∞. It can be seen from the value of CM in Fig.3 that node i satisfies the condition of insertion positions $L-1$ and L. Therefore, the cost increment after insertion is saved to CM.

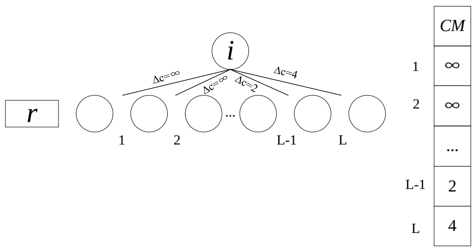

Fig. 3: CM building process

From the above definitions of IM and CM, it is clear that a larger $IM^t_{i,j}$ denotes a greater correlation between node i and node j, while a smaller CM_l means a smaller incremental cost of routing when the node is inserted into location l. Therefore, both IM and CM are considered when inserting node i into a certain path. We first perform ascending and descending operations on IM and CM, respectively. Then, the insertion position of node i is determined based on the contents of the two matrices. Specifically, we base the selection on the sum of the ranking values of the positions to be inserted in the two matrices. For example, if the position to be inserted is ranked $3rd$ in IM and $8th$ in CM, the priority value of the inserted position is equal to 11. Finally, according to the priority value of each inserted position, the inserted position with the lowest priority value is selected to insert node i. It is worth noting that if node i is in the current path and there is no location where it can be inserted, then the node will start a new route from the warehouse. Based on the above discussions, a guided reinsertion operator based on local information is detailed in Algorithm 3. From Algorithm 3, we know that the insertion operator operation is mainly focused on the process of creating l of CM and inserting one by one of the clients D to be inserted therefore, the time complexity is $O(l * D)$.

Algorithm 3 Guided reinsertion operator based on local information

```
Input:
    the current route $r$, the node set $ins$ to be inserted, the information matrix $IM$, and the
    distance matrix $Dis$.
Output: $Nr$
 1: $Nr = r$;
 2: for $i = 1$ to $|ins|$ do // $|ins|$ denotes the number of customers to be inserted
 3:     $l \leftarrow$ Calculate the path $r$ length;
 4:     $stay \leftarrow$ Record the location of the path $r$ repository;
 5:     $Pd \leftarrow ins(i);//InsertNode$
 6:     $CM \leftarrow$ Create a two-dimensional matrix with initial value $\infty$ for $l * l$;
 7:     $j \leftarrow 1$;
 8:     while $j < l$ do //Calculate $CM$
 9:         if the node $Pd$ insertion is overweight then
10:             $j = stay(\text{find}(stay == j) + 1)$; // go to the next path
11:         else
12:             if $Pd$ can be inserted in the current position then
13:                 $Pi = Nr(j)$;// Predecessor Nodes
14:                 $Pj = Nr(j + 1)$;// Post nodes
15:                 // Calculating Cost Increment
16:                 $CM(j) = Dis(Pi, Pd) + Dis(Pd, Pj) - Dis(Pi - Pj)$;
17:                 $j = j + 1$;
18:             end if
19:         end if
20:         if $Pd$ has no position where it can be inserted then
21:             $IX = []$;
22:         else
23:             $IM \leftarrow$ Sorting node $Pd$ in ascending order;
24:             $CM \leftarrow$ Sort the cost matrix of node $Pd$ in descending order;
25:             $IX \leftarrow$ Find the number of the lowest position in the sum of $IM$ and $CM$ rankings;
26:         end if
27:         if $IX$ is empty then
28:             $Nr \leftarrow$ Insert $Pd$ and repository at the end of $Nr$;
29:         else
30:             $Nr \leftarrow$ Insert $Pd$ at the location of $IX$;
31:         end if
32:     end while
33: end for
```

4.5. Removal-reinsert-based neighborhood search

When using PSO to solve VRPTW problems, an efficient neighborhood search operator plays a positive and crucial role in helping particles to jump out of local optima, improving the solution accuracy, and accelerating the convergence speed [43]. Furthermore, the study in [44] verify that the neighborhood region of elite particles is more likely to contain high-quality solutions or even global solutions. Therefore, to enhance the local search efficiency of N-CLPSO, we perform a neighborhood search operation for elite individuals, rather than all individuals, in the population. Concretely, the neighborhood search operation only exert on the individual optimal solution *Pbest*.

Motivated by the Large Neighborhood Search(LNS) [45] in we introduce a removal-reinsert-based neighborhood search method. First, we randomly choose one of the removal methods to remove D customers from the complete route and insert them into the set Nt of customers to be inserted. Then, we reinsert the route by guided reinsertion operator based on local information(see Section 4.4), i.e., the customer nodes in Nt are reinserted into the route, forming a route that traverses all nodes to generate a new solution. Finally, we compare the routes before and after the update, and then keep the better individuals for the next iteration.

We determine the number of customers D to be removed according to Eq. (29).

$$D = \min\left(\text{ceil}\left(\frac{I}{10}\right), \text{ceil}\left(\frac{N}{10}\right)\right) \tag{29}$$

where N is the number of customers and I is the number of generations in the population for which the optimal solution has not been updated. As the number of generations of optimal solution stagnation increases, the number of removed customers also increases. Obviously, in the local search, more removed customs mean a larger perturbation range of a particle. Thus, removing more customers in the local search process can help an individual to jump out of the local optimum. However, removing too many customers increases computing costs and may reduce local search efficiency. Conversely, if the number of removed customers is too little, the capability of the individual that jumps out of the local optimum as well as search for more promising regions is small. Through extensive experiments, we finally chose "N/10" as the upper bound of D.

(1) In Section 4.4.1, we define IM to measure the probability of being served by the same vehicle successively among customers. With the help of IM, we can design an information matrix-based removal strategy, as in Algorithm 4. First, we randomly select a customer point i and insert it into the customer set Nt. After that, we randomly pick a point $i^{'}$ in Nt and select j points to join Nt based on IM, where node j denotes the node that is least likely to be adjacent to node $i^{'}$, i.e. $IM^t_{i^{'},j} = min(IM^t_{i^{'}})$. Since we are borrowing the IM for evaluation, the time complexity of the operator is $O(D)$.

Algorithm 4 Removal strategy based on IM

```
Input:
    //Information Matrix and node to be deleted, respectively
    $IM$; $D$;
Output: Delete node set $Nt$
1: $Nt = \phi$;
2: for $x = 1$ to length of $D$ do
3:     Randomly select a customer $i$ from $Nt$;
4:     Find the node $j$ that is least likely to be adjacent to customer $i^{'}$ by $IM$;
5:     $Nt = Nt \cup \{j\}$;
6: end for
```

(2) We propose a removal operator based on the customer's removal cost, and its pseudo-code is shown in Algorithm 5. The algorithm performs a removal operation on a customer by calculating the difference in cost incurred by removing each customer point from the original

path, i.e., the removal cost. We select a customer i to insert into Nt based on the removal cost corresponding to each customer in a roulette wheel method. The time complexity of our removal of customer D is $O(D)$.

Algorithm 5 Removal strategy based on removal cost

```
Input:
    // Enter the route, distance matrix and node to be deleted, respectively
    $r$;$Dis$;$D$;
Output: Delete node set $Nt$
 1: $Nt = \phi$;
 2: $L = r.length$;
 3: for $x = 2 : L - 1$ do // Calculate the removal cost $\Delta c$ for each customer point removed
 4:     $F = r(x - 1)$; // Precursor node of node $x$
 5:     $C = r(x)$; // Node $x$
 6:     $R = r(x + 1)$; // Posterior node of node $x$
 7:     $\Delta c(x) = Dis(F, C) + Dis(C, R) - Dis(F, R)$;
 8: end for
 9: $k = 0$;
10: while $k < D$ do
11:     Randomly select the customer $i$ in $\Delta c$ where $Nt$ does not appear by roulette;
12:     $Nt = Nt \cup \{i\}$;
13:     $k = k + 1$;
14: end while
```

4.6. Diversity retention strategy based on elite fragments

In order to maintain the population diversity and improve the quality of the solution, a diversity retention strategy based on elite fragments is designed in this study.

In PSO, the diversity of particles disappears as the number of iterations rises, which directly leads to the premature convergence of the algorithm. In order to prevent the particles from converging prematurely, we need to perturb the particles. However, a large range of random perturb can easily make the particles degenerate and degrade the algorithm performance thought it is beneficial for improving population diversity. Generally, in evolutionary algorithms, elite individuals generally contain better genetic fragments. So when perturbing an individual, if we can also retain some gene fragments shared by other elite individual, we can increase diversity while retaining superior genetic information. Therefore, inspired by the LCS [46], we propose a diversity retention strategy based on elite fragments.

When using PSO to optimize a VRPTW problem, each particle can often be represented by a complete route. Therefore, we base on the elite fragment between the particle and the elite particle, other nodes will be inserted into the elite fragment one by one through the guided reinsertion strategy to form a new route, and finally, we reserve the better individual. It can be seen that the retention of elite fragments avoids excessive degradation of particles to some extent, and the strategy perturbs the current particles while ensuring the performance of the algorithm. Obviously, when two particles are more similar, their extracted fragments are longer and the set of nodes to be inserted is smaller. On the contrary, if two particles are more different, their common sequence is shorter and the number of nodes to be removed will be more, which is more conducive to increasing population diversity as well as helping the algorithm to jump out of the local optimum.

Fig.4 depicts process of the proposed diversity retention strategy. For example, the sample particle $r1$ is $\{1, 2, 5, 6, 1, 3, 4, 7, 8, 1\}$, and the current particle is $r2$ is $\{1, 3, 6, 1, 2, 4, 7, 1, 5, 8, 1\}$. Then, we can obtain that the LCS of $r1$ and $r2$ is $\{1, 6, 1, 4, 7, 8, 1\}$. Based on the LCS, the set of nodes to be inserted is $\{2, 3, 5\}$. After that, we insert the customer nodes in the node-set into the LCS to get the new $r3$. Nots that the finding process of the LCS depends on the length of the $r1$ and $r2$, thus the time complexity of it is O($n * m$).

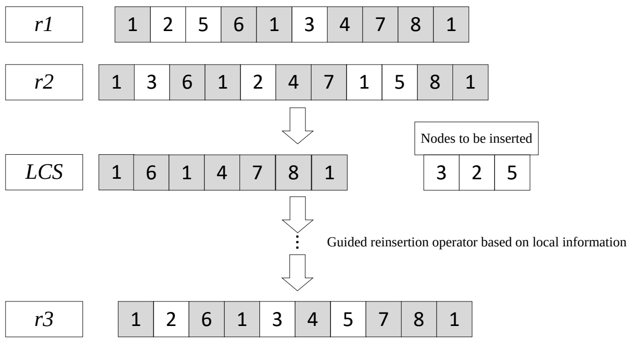

Fig. 4: Illustrative Example of Diversity Strategy

4.7. Complexity analysis of N-CLPSO

In this section, we analyze the overall time complexity of N-CLPSO. For convenience, we assume that the population size of the particle is D, the number of clients is N, the maximum stopping condition is $Iter$, the clients to be inserted are m, and the CM length is n.

According to section 4.1, N-CLPSO is the addition of a vehicle insertion strategy, diversity retention strategy, and neighborhood search strategy to the framework of CLPSO. After the description of the above sections, we can easily know that the time complexity of its main body is O(D*N); the time complexity of the vehicle insertion strategy is $O(K * m * n)$; the time complexity of the neighborhood search strategy is $O(n * m)$; the time complexity of diversity retention strategy is O(N*N). Since the policies in N-CLPSO are invoked only in special cases, the time complexity of N-CLPSO is $O(Iter * D * N) + O(D * k * m * n) + O(m * n) + O(N * N)$. Clearly, the time complexity of the algorithm is approximately equal to $O(Iter * D * N)$ when the number of iterations is sufficient.

5. Experimental evaluation

In this section, Section 5.1 describes the general setup of the experiments. Section 5.2 describes the parameter tuning of the algorithm. Section 5.3 presents a sensitivity analysis of each component of the algorithm. Sections 5.4 and 5.5 compare N-CLPSO with other 3 PSO variants and other 7 state-of-the-art algorithms and draw some conclusions.

5.1. Setup

N-CLPSO is tested on a classical benchmark of 56 VRPTW instances proposed by Solomon [47] since the benchmark can reflect various real life scheduling problems. According to properties, the instances can be classified into three categories: clients distributed in a clustered manner (Class C), clients distributed in a random manner (Class R), and test sets with a mixture of clustering and randomness (Class RC). On this basis, the test set can be further classified into to two categories, i.e., problems with smaller vehicle capacities and more compact time windows (C1, R1 and RC1), and problems with larger vehicle capacities and longer dispatch cycles (C2, R2 and RC2).

In this study, extensive experiments are conducted to investigate the performance of N-CLPSO. In the experiments, the inertia weight value w in Eq.(3) is initialized to 0.9 and decreases linearly from 0.9 to 0.4 during the optimization process, and the acceleration coefficient c_1 is set to 2.0. In this paper, the refresh gap "*rg*" of CLPSO is set according to [30]. The learning probability Pc and the exemplar selection method are used in Eq.(21) and Eq.(22). Except for the experiments of parameter tuning, the parameters of Eq.(24)-Eq.(25) are set to $a = 0.5$, $k_1 = 1$ and $k_2 = 2$, respectively. The population size is set to $N = 20$. Neighborhood search and diversity preservation policies are activated at 10 and 100 generations of $Pbest$ and $Gbest$ stagnation updates, respectively. The optimization process will be stopped when $Gbest$ has been stagnation for consecutive 10,000 generations. Each trial is performed independently 5 times.

5.2. Parameter tuning

This section mainly compares the effects of the three parameters involved in the calculation of the space-time distance (i.e., a, k_1 and k_2 in Eqs. (24)-(25)) on the N-CLPSO. Table 1 shows the average best solutions obtained by N-CLPSO with different parameters on instances R101 and R201, where ”MNV” and ”MTD” denote the results of the average number of vehicles and average distance, respectively, and ”Mean indicates the average distance with the same parameter a. Note that, the parameter a determines the time-space proportion of customer distance and time window. Thus, when a is larger, the proportion of spatial distance in customer distance information is more significant. On the contrary, if a is smaller, the proportion of temporal distance in customer distance information is more significant. k_1 and k_2 denote the cost coefficients of remaining vehicle service time and waiting time, respectively, and the difference between them should not be too significant. Otherwise, the quality of the final solution will be affected. Therefore, the values of a, k_1 and k_2 are set as follow: $a \in \{1, 2, 3\}$, $k_1 \in \{1, 2, 3\}$, $k_2 \in \{1, 2, 3\}$.

To illustrate the results more clearly, the experimental results are marked. Concretely, the darker the cell background color is, the better the result is. In R101 and R201, the average distance is shortest when $a = 0.5$, followed by $a = 0.7$ and $a = 0.3$. This may be that N-CLPSO focuses more on spatial distance when a becomes smaller, which leads to a shorter average distance. However, the most favorable results of $a = 0.5$ indicate that considering both time and spatial distance can enable N-CLPSO have a more comprehensive performance. In R101 and R201, which have a narrow time window and a wide time window, respectively, N-CLPSO can offer more promising performance. Therefore, it is reasonable to believe that the waiting time may be more important than the remaining service time of the vehicle when calculating the time distance. Interestingly, we find that N-CLPSO achieve the best results when $a = 0.5$, $k_1 = 1$, $k_2 = 2$. Therefore, the parameters of N-CLPSO are set to $a = 0.5$, $k_1 = 1$, $k_2 = 2$.

Table 1: Results for R101 and R201 with different parameters.

Intances	*a*	*k1*	*k2*	MNV	MTD	Mean
R101	0.3	1	1	19	1652.00	1657.23
			2	19	1661.48	
			3	19	1656.38	
		2	1	19	1655.74	
			2	19	1655.52	
			3	19	1653.78	
		3	1	19	1659.36	
			2	19	1661.89	
			3	19	1658.92	
	0.5	1	1	19	1654.60	**1655.73**
			2	19	1651.10	
			3	19	1655.97	
		2	1	19	1655.44	
			2	19	1653.75	
			3	19	1655.75	
		3	1	19	1658.41	
			2	19	1659.77	
			3	19	1656.81	
	0.7	1	1	19	1660.99	1656.26
			2	19	1654.05	
			3	19	1657.26	
		2	1	19	1654.51	
			2	19	1651.75	
			3	19	1658.37	
		3	1	19	1659.20	
			2	19	1656.46	
			3	19	1653.75	
R201	0.3	1	1	4	1291.91	1282.23
			2	4	1261.57	
			3	4	1274.32	
		2	1	4	1277.33	
			2	4	1287.35	
			3	4	1272.89	
		3	1	4	1289.22	
			2	4	1301.87	
			3	4	1283.58	
	0.5	1	1	4	1289.79	**1277.15**
			2	4	1258.83	
			3	4	1304.88	
		2	1	4	1278.27	
			2	4	1269.71	
			3	4	1274.02	
		3	1	4	1284.01	
			2	4	1271.03	
			3	4	1263.84	
	0.7	1	1	4	1282.50	1278.16
			2	4	1281.36	
			3	4	1260.14	
		2	1	4	1283.96	
			2	4	1286.35	
			3	4	1282.73	
		3	1	4	1258.90	
			2	4	1284.19	
			3	4	1283.29	

5.3. *Sensitivity analysis of components in N-CLPSO*

This section aims to clarify the impact of the components proposed in N-CLPSO. We attempt to verify the effectiveness of the strategy by adding components one by one to the original CLPSO. The strategys proposed in this paper focuses on speeding up the convergence and maintaining population diversity.

In N-CLPSO, we use the new velocity selection strategy and vehicle insertion strategy to velocity up the convergence of N-CLPSO. To this end, 3 algorithms are adopted as competitors to N-CLPSO, i.e., ”CLPSO” which does not contain the new proposed strategies, ”add-V” which denotes that only the new velocity update strategy is involved in CLPSO, and ”add-C” which means that only the vehicle insertion strategy is added in CLPSO.

Comparison results shown in Fig.5 demonstrate that “N-CLPSO” shows faster convergence in these experiments and higher accuracy in solving at later stages. The convergence velocity and solution accuracy of “add-C” are slightly lower than those of “N-CLPSO”, but higher than those of “add-V” and “CLPSO”. Although “add-V” displays similar performance as “CLPSO”, in terms of solution accuracy, it yields significantly higher convergence velocity than “CLPSO”. Meanwhile, we find that the convergence velocity of ”add-C” in Fig. 5 is significantly better than that of ”add-V”, which is probably because the vehicle insertion strategy reduces the number of vehicles of the particles as much as possible. So that the particles are closer to the optimal solution, the convergence velocity of ”add-C” is faster than that of ”add-V”. Therefore, we can see that the new velocity update strategy and the vehicle insertion strategy play positive performance on improving the convergence velocity.

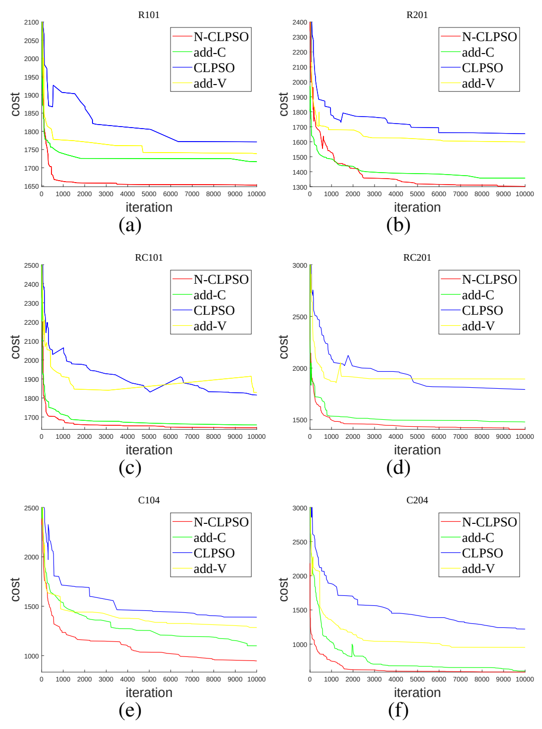

Fig. 5: Contribution of velocity selection strategy and vehicle insertion strategy

In addition, to investigate the advantages of the neighborhood search and diversity retention strategies applied in N-CLPSO, 3 variants of N-CLPSO are selected as peer algorithms. Specifically, "noLV" and "noZ" denote two algorithms in which the neighborhood search strategy and the diversity retention strategy are removed from N-CLPSO, respectively, while "noLV+noZ" means that both the two new proposed strategies are removed from N-CLPSO. The comparison results in most datasets, in terms of the Percentage Error (PE) and the Average Percentage Error (PEav) of the optimal distance between each algorithm and the best-known for the minimum number of vehicles, are shown in Fig.6. For instance, if the optimal distances corresponding to the minimum number of vehicles for the proposed algorithm and the best-known algorithm are 19, 1648 and 19, 1650, respectively, the corresponding percentage error is (19 - 19)/19 + (1648 - 1650)/1650.)

It can be observed that both the addition of the neighborhood search module and the diversity retention strategy optimized the optimal solution to be closer to the global optimal solution. However, the difference between "noLV" and "noZ" in most of the data sets is not significant, except that "noLV" is better than "noZ" in the R201 and RC201. The results manifest that diversity retention strategies may play more important and more positive performance than the neighborhood search strategy on "2" class problems. It is clear that the optimal results were obtained by adding both the neighborhood search and diversity retention strategies to N-CLPSO.

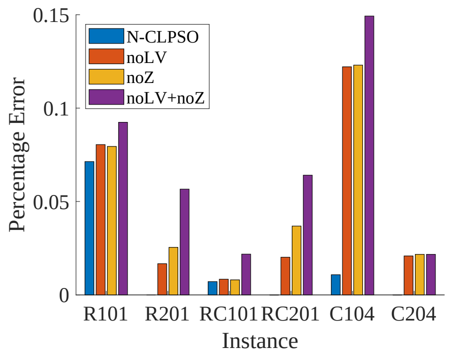

Fig. 6: Contribution of Neighborhood Search and Diversity retention Strategies

In N-CLPSO, we used a combination of the IM and the CM to guide the neighborhood reinsertion strategy. To verify the advantages of the combination matrix used in N-CLPSO, we implemented three variants of N-CLPSO, i.e., N-CLPSO with IM only, N-CLPSO with CM only, and N-CLPSO with IM and CM. Experimental results demonstrated in Fig.7 verify that using CM is better than using only IM in R101, R201, RC101 and RC201 problems, while using IM is better than using only CM in the set clustering dataset C104 and C204, it is clear that using both CM and IM has the best performance.

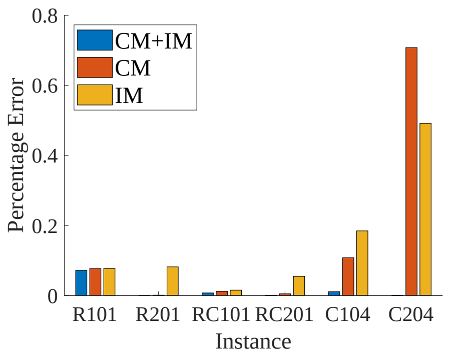

Fig. 7: Contribution of correlation matrix and cost matrix

5.4. *Comparison with other PSO-based algorithms*

Among the PSO-based algorithms, we compare the N-CLPSO algorithm with S-PSO [11], HMPSO [48], with HPSO [37]. As discussed in Section 4.2 in the revised manuscript, S-PSO and N-CLPSO share a similar velocity and position update approach. To insight the new proposed strategies in N-CLPSO, we compare S-PSO with N-CLPSO on 56 VRPTW instances. The average PEav of N-CLPSO is 0.038, smaller than 0.072 of S-PSO. Table 2 shows the simulation results for six instances, where "Time" indicates the average running time of the algorithm per generation. The population size and termination conditions of S-PSO are the same as those of N-CLPSO.

Table 2: Performance comparison between S-PSO and N-CLPSO

Instance	S-PSO(2012)		N-CLPSO	
	PE	Time(s)	PE	Time(s)
R101	0.080	1.5	**0.077**	1.7
R201	0.018	1.2	**0**	1.2
C101	**0**	1.2	**0**	1.2
C201	**0**	1.1	**0**	1.1
RC101	0.039	1.5	**0.035**	1.6
RC201	0.012	1.1	**0**	1.1

Table 2 shows that N-CLPSO achieves better results on all six datasets. The results verify that adding local search and diversity preservation strategies can significantly improve the algorithm's performance. Meanwhile, there are similarities in how the two algorithms update their positions at the time level, as described by the time complexity in Section 4.7. Although N-CLPSO incorporates various strategies, the strategies are invoked only under certain conditions. Thus, the comparison results in the table display that N-CLPSO consumes only slightly more time than S-PSO on the R101 and RC101, while the time consumed in the other instances is the same.

N-CLPSO was compared with HMPSO and HPSO on 56 VRPTW instances. Due to space limitation, only the comparison results on six typical VRPTW instances are shown in Fig. 8. The average PEav of N-CLPSO is 0.038, much better than 0.165 for HMPSO and 0.352 for HPSO. It means that N-CLPSO is significantly better than HMPSO and HPSO.

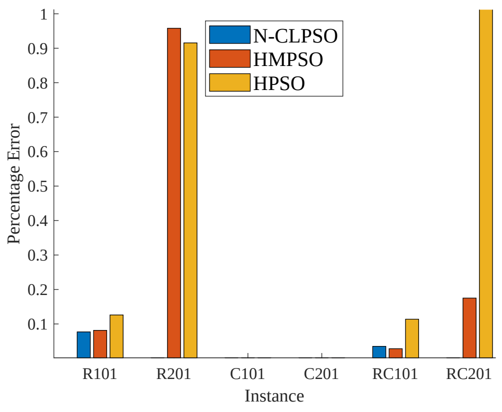

Fig. 8: Performance comparison between HMPSO, HPSO ,and N-CLPSO

Furthermore, to verify whether there is a significant difference between N-CLPSO and the 3 competitors, in terms of the PEav, a set of Wilcoxon signed rank tests are conducted. The comparison results are presented in Table 3. The table shows that N-CLPSO is significantly better than all the 3 peer algorithms since the three R^{+} values are greater than corresponding R^{-} values. Although N-CLPSO and S-PSO share some basic operators, N-CLPSO dominates S-PSO in all the test instances. Thus, we can draw a preliminary conclusion that the new proposed strategies enable N-CLPSO to have a favorable performance.

Table 3: The Wilcoxon non-parametric test of N-CLPSO and the three competitors.

N-CLPSO vs	S-PSO	HMPSO	HPSO
R^{+}	45	27	37
R^{-}	0	21	11
p-value	**5.1793e-9**	**0.014**	**8.5137e-8**

5.5. Comparison with other algorithms

To verify the comprehensive performance of N-CLPSO, we compared it with 9 state-of-the-art algorithms, including two GAs [12, 49], one LNS [50], one ACO [10], one TS [51], four other heuristics [52, 53, 54, 55], and finally giving the results of the best average operation for each subclass (C1, C2, R1, R2, RC1, and RC2). Moreover, the best-known solutions also are adopted in experiments. Note that the experimental parameters of each peer algorithm are consistent with the corresponding literature.

In Table 4, we also give a comparison of the proposed algorithm with the best-known results, where "NV" and "TD" denote the best-known results for the best number of vehicles and the corresponding minimum distance, respectively. "BNV" denotes the best number of vehicles, while "BTD" denotes the minimum distance corresponding to the best number of vehicles. "MNV" and "MTD" denote the average number of vehicles and the average distance obtained by the proposed algorithm respectively. "Deviation" = (BTD- TD)/BTD is the percentage deviation of the algorithm from the best-known result for the same number of vehicles, which is used to measure the algorithm's quality. "Std-N" and "Std-T" denote the standard deviation of the number of vehicles and the distance obtained by the algorithm, respectively, and are used to measure the stability of the solution. Time denotes the average running time of the algorithm per generation.

Table 4: Comparison with the best-known results.

		Best-known				N-CLPSO					
Dataset	NV	TD	References	BNV	BTD	MNV	MTD	Deviation	Std-N	Std-T	Time(s)
R101	18	1613.59	[56]	19	1648.08	19	1651.1	-	0	3.42	1.7
R102	17	1486.12	[57]	**17**	**1486.12**	17	1493.12	0	0	7.05	1.5
R103	13	1292.68	[58]	13	1299.99	13	1312.13	0.56%	0	17.66	1.8
R104	9	1007.24	[59]	10	996.27	10	1021.06	-	0	15.54	2.1
R105	14	1377.11	[57]	**14**	**1377.11**	14	1379.35	0	0	3.17	1.4
R106	12	1251.98	[59]	12	1262.80	12	1264.5	0.86%	0	2.40	1.3
R107	10	1104.66	[60]	11	1081.17	11	1107.85	-	0	2.40	1.2
R108	9	960.88	[61]	10	985.76	10	988.59	-	0	2.45	1.2
R109	11	1194.73	[62]	11	1210.73	11.7	1196.21	1.32%	0.58	8.85	1.5
R110	10	1118.59	[59]	11	1101.49	11	1114.61	-	0	10.26	1.8
R111	10	1096.72	[63]	11	1064.67	11	1072.80	-	0	7.78	1.5
R112	9	982.14	[64]	10	974.95	10	982.27	-	0	8.85	1.6
R201	4	1252.37	[62]	**4**	**1252.37**	4	1258.83	0	0	7.52	1.2
R202	3	1191.70	[63]	3	1225.02	3.5	1178.49	2.72%	0.58	5.56	1.6
R203	3	939.54	[59]	3	962.25	3	976.32	2.36%	0	9.06	1.5
R204	2	825.52	[65]	3	766.13	3	777.83	-	0	10.92	1.7
R205	3	994.42	[63]	3	1027.79	3	1051.16	3.25%	0	14.61	1.9
R206	3	906.14	[66]	3	939.46	3	947.38	3.55%	0	6.53	1.7
R207	2	837.20	[67]	3	872.40	3	877.56	-	0	5.90	1.5
R208	2	726.75	[59]	2	740.36	2	772.04	1.83%	0	28.30	2.1
R209	3	909.16	[68]	3	943.72	3	957.46	3.66%	0	10.13	1.8
R210	3	938.58	[69]	3	965.88	3	986.43	2.74%	0	17.00	1.7
R211	2	892.71	[65]	3	828.90	3	842.35	-	0	11.88	1.5
C101	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.2
C102	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.1
C103	10	828.06	[57]	10	839.35	10	840.20	1.35%	0	1.08	1.3
C104	10	824.78	[57]	10	833.67	10	837.08	1.07%	0	4.82	1.2
C105	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.1
C106	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.1
C107	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.2
C108	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.1
C109	10	828.94	[57]	**10**	**828.94**	10	828.94	0	0	0	1.2
C201	3	591.56	[57]	**3**	**591.56**	3	591.56	0	0	0	1.1
C202	3	591.56	[57]	**3**	**591.56**	3	591.56	0	0	0	1.1
C203	3	591.17	[57]	**3**	**591.17**	3	591.17	0	0	0	1.3
C204	3	590.60	[57]	**3**	**590.60**	3	591.08	0	0	0	1.2
C205	3	588.88	[57]	**3**	**588.88**	3	588.88	0	0	0	1.1
C206	3	588.49	[57]	**3**	**588.49**	3	588.49	0	0	0	1.2
C207	3	588.29	[57]	**3**	**588.29**	3	588.29	0	0	0	1.1
C208	3	588.32	[57]	**3**	**588.32**	3	588.32	0	0	0	1.1
RC101	14	1696.94	[70]	15	1635.11	15	1644.17	-	0	6.98	1.6
RC102	12	1554.75	[70]	13	1503.42	13	1516.54	-	0	11.37	2.0
RC103	11	1261.67	[70]	11	1277.99	11	1278.11	1.28%	0	0.20	1.2
RC104	10	1135.48	[71]	**10**	**1135.48**	10	1140.4	0	0	6.96	1.6
RC105	13	1629.44	[61]	14	1542.55	14	1542.91	-	0	6.96	1.8
RC106	11	1424.73	[61]	12	1388.70	12	1396.14	-	0	10.06	2.0
RC107	11	1222.1	[69]	11	1230.48	11	1235.71	0.68%	0	5.78	1.7
RC108	10	1139.82	[70]	11	1157.12	11	1163.34	-	0	6.72	1.5
RC201	4	1406.91	[59]	**4**	**1406.91**	4	1410.48	0	0	6.16	1.1
RC202	3	1365.65	[72]	4	1169.67	4	1176.73	-	0	9.98	1.5
RC203	3	1049.62	[72]	3	1082.57	3	1107.58	3.04%	0	21.34	1.8
RC204	3	798.41	[59]	3	828.61	3	834.68	3.64%	0	6.93	1.5
RC205	4	1297.19	[59]	**4**	**1297.19**	4	1314.81	0	0	14.06	1.9
RC206	3	1146.32	[68]	**3**	**1146.32**	3	1156.29	0	0	8.64	1.6
RC207	3	1061.14	[65]	3	1095.67	3	1120.65	3.15%	0	19.85	1.8
RC208	3	828.14	[73]	3	843.28	3	872.78	1.80%	0	17.33	1.6

It can be seen from Table 4 that the N-CLPSO reaches 22 optimal solutions. The average deviations of R1, C1 and RC1 are 0.55%, 0.27% and 0.65%, respectively, which are less than 1%. Meanwhile the deviations of R2 and RC2 are 2.51% and 1.94%, respectively, which shows that N-CLPSO has favorable performance in the “1” class problems. Regarding the “Std-T” standard deviation, all of class C2 is 0, and only class C1 has 1.08 and 4.82 for C103 and C104. The "Std-T" of R1, R2, RC1, RC2, and the total data set is 7.38, 11.58, 6.88, and 13.04, respectively, and most of the standard deviations are below 10. And under "Std-N" standard deviation, only R109 and R202 possess 0.58, and all other data sets are 0. The results show that N-CLPSO is stable and has good robustness. Furthermore, it can be seen that N-CLPSO yields more favorable performance in the "1" class problems who have narrow time windows, while there is a small decrease in the performance of N-CLPSO in the "2" class problems who have longer time windows.

In Table 5, we have selected two recently published state-of-art algorithms as competitors for N-CLPSO. Note that, the data for each algorithm prioritizes the minimum vehicle, followed by consideration of the shortest path length. The results show that N-CLPSO obtains the best results on 47 out of the 56 data sets, while ASC-BSO [52] and MOLNS [50] yield the best results on 21 and 17 test problems, respectively. In all three algorithms, N-CLPSO is able to find the minimum number of vehicles and has the shortest path length in most of the data sets. Although the running time is directly related to the encoding method and experimental equipment, we also compare the average iteration time of the three algorithms as one of the reference metrics. N-CLPSO has an average running time of 1.48s per arithmetic case, which is competitive with ASC-BSO's 1.96 and MOLNS's 1.04.

Table 5: Comparison with recently published representative algorithms

	ACS-BSO			MOLNS			N-CLPSO		
Dataset	NV	TD	Time(s)	NV	TD	Time(s)	NV	TD	Time(s)
R101	19	1671.16	2.1	19	1654.93	1.3	**19**	**1648.08**	1.7
R102	17	1504.60	2.1	18	1475.33	1.0	**17**	**1486.12**	1.5
R103	14	1245.86	1.9	14	1240.44	1.4	**13**	**1299.99**	1.8
R104	11	1010.73	1.9	10	1010.72	1.3	**10**	**996.27**	2.1
R105	15	1366.05	1.9	15	1389.85	0.9	**14**	**1377.11**	1.4
R106	13	1288.84	2.1	13	1269.14	1.0	**12**	**1262.8**	1.3
R107	11	1101.56	2.0	11	1102.72	1.0	**11**	**1081.17**	1.2
R108	**10**	**974.17**	2.3	10	991.57	1.1	10	985.76	1.2
R109	12	1165.71	2.0	12	1177.76	1.0	**11**	1210.73	1.5
R110	**11**	**1090.92**	2.0	12	1129.60	0.7	11	1101.49	1.8
R111	11	1148.14	2.0	12	1108.70	0.9	**11**	1064.67	1.5
R112	10	1004.53	2.0	**10**	**964.15**	1.1	10	974.95	1.6
R201	4	1336.05	2.1	4	1305.25	1.1	**4**	**1252.37**	1.2
R202	4	1128.05	2.4	4	1093.67	1.1	**3**	**1225.02**	1.6
R203	3	1020.10	2.3	4	915.43	1.2	**3**	**962.25**	1.5
R204	3	834.92	2.2	3	775.99	1.2	**3**	**766.13**	1.7
R205	3	1105.38	2.7	3	1075.10	0.8	**3**	**1027.79**	1.9
R206	3	949.11	2.4	3	979.21	1.0	**3**	**939.46**	1.7
R207	4	812.35	3.0	**3**	**851.89**	1.1	3	872.4	1.5
R208	2	940.30	2.9	2	754.99	1.2	**2**	**740.36**	2.1
R209	3	1046.73	2.7	4	898.23	0.9	**3**	**943.72**	1.8
R210	3	1069.26	2.4	4	941.58	0.9	**3**	**965.88**	1.7
R211	3	836.36	2.1	3	838.14	0.8	**3**	**828.9**	1.5
C101	**10**	**828.94**	1.6	**10**	**828.94**	0.9	**10**	**828.94**	1.2
C102	**10**	**828.94**	1.4	**10**	**828.94**	1.1	**10**	**828.94**	1.1
C103	**10**	**828.06**	1.5	10	828.94	1.0	10	839.35	1.3
C104	**10**	**828.78**	1.7	10	828.94	1.0	10	833.67	1.2
C105	**10**	**824.94**	1.6	**10**	**828.94**	0.9	**10**	**828.94**	1.1
C106	**10**	**828.94**	1.4	**10**	**828.94**	0.9	**10**	**828.94**	1.1
C107	**10**	**828.94**	1.4	**10**	**828.94**	0.7	**10**	**828.94**	1.2
C108	**10**	**828.94**	1.5	**10**	**828.94**	0.9	**10**	**828.94**	1.1
C109	**10**	**828.94**	1.5	**10**	**828.94**	0.9	**10**	**828.94**	1.2
C201	**3**	**591.56**	1.8	**3**	**591.56**	1.0	**3**	**591.56**	1.1
C202	**3**	**591.56**	1.2	**3**	**591.56**	1.0	**3**	**591.56**	1.1
C203	**3**	**591.17**	1.9	3	591.56	1.0	**3**	**591.17**	1.3
C204	**3**	**591.60**	1.6	**3**	**590.60**	1.0	**3**	**590.60**	1.2
C205	**3**	**588.88**	1.1	**3**	**588.88**	0.9	**3**	**588.88**	1.1
C206	**3**	**588.49**	1.5	**3**	**588.49**	0.8	**3**	**588.49**	1.2
C207	**3**	**588.29**	1.5	**3**	**588.29**	0.8	**3**	**588.29**	1.1
C208	**3**	**588.32**	1.0	**3**	**588.32**	0.8	**3**	**588.32**	1.1
RC101	16	1643.78	2.2	15	1662.56	0.8	**15**	**1635.11**	1.6
RC102	14	1464.63	1.9	14	1486.35	1.1	**13**	**1503.42**	2.0
RC103	**11**	**1275.64**	1.5	12	1291.95	1.3	11	1277.99	1.2
RC104	10	1156.92	1.5	10	1162.53	1.2	**10**	**1135.48**	1.6
RC105	14	1609.68	1.7	15	1604.53	0.9	**14**	**1542.55**	1.8
RC106	13	1378.45	1.6	13	1400.09	1.1	**12**	**1388.70**	2.0
RC107	11	1318.69	1.7	12	1259.55	1.2	**11**	**1230.48**	1.7
RC108	**11**	**1134.85**	1.7	11	1205.13	1.2	11	1157.12	1.5
RC201	4	1514.41	2.5	4	1497.89	0.9	**4**	**1406.91**	1.1
RC202	4	1326.71	2.6	4	1199.53	1.3	**4**	**1169.67**	1.5
RC203	3	1166.91	2.0	4	985.54	1.7	**3**	**1082.57**	1.8
RC204	3	929.94	2.6	**3**	**805.46**	1.5	3	828.61	1.5
RC205	4	1360.91	2.4	5	1340.38	1.0	**4**	**1297.19**	1.9
RC206	3	1237.21	2.3	3	1316.42	1.0	**3**	**1146.32**	1.6
RC207	4	1039.59	2.6	4	1031.62	1.3	**3**	**1095.67**	1.8
RC208	3	910.59	2.3	3	859.13	1.0	**3**	**843.28**	1.6

Meanwhile, to verify the performance of N-CLPSO in terms of statistics results, we used the Wilcoxon signed ranks test to compare it with ACS-BSO and MOLNS, and the test results are demonstrated in Table 6. From the statistics test results, we see that, for N-CLPSO and ACS-BSO, the computed R^+, R^-, and p-value are 35, 7, and 2.6715e-7 respectively. For N-CLPSO and ACS-BSO, the computed R^+, R^-, and p-value are 35, 5, and 4.8356e-7 respectively. The results verify that N-CLPSO dominates the 2 competitors in the Wilcoxon test. Since a large number of instances have various properties, we can regard that N-CLPSO has a more reliable and comprehensive performance than the other algorithms.

Table 6: The Wilcoxon non-parametric test of N-CLPSO with ACS-BSO and MOLNS.

N-CLPSO vs	ACS-BSO	MOLNS
R^+	35	35
R^-	7	5
p-value	**2.6715e-7**	**4.8356e-7**

In Table 7, the average best results of the given algorithms are given for each subclass (C1, C2, R1, R2, RC1 and RC2). The results are given in the form of NV_TD, where NV and TD are the averages of the best minimum number of vehicles (NV) and the best minimum total travel distance (TD) found in each subclass using the corresponding method, respectively. For instance, the data "13.10_1213.16" in the second row of the table indicates that the average number of vehicles and the average distance cost obtained by ACO-N on independently runs are 13.10 and 1213.16, respectively. The best results for each category are highlighted in bold.

It can be seen from Table 7 that for the class C2, the N-CLPSO is the same as the best-known result. In class C1, the N-CLPSO algorithm is slightly lower than the best-known result regarding the shortest distance. In R2 and RC2, N-CLPSO performance is lower than D-VND. In R1 and RC1, N-CLPSO has slightly lower performance than HRRGA in terms of NV but has more advantages in terms of TD and is closest to the best-known results compared to the other six algorithms. Therefore, N-CLPSO performs better in solving the "1" class problems than the "2" ones.

Table 7: Comparison of average levels.

	References	R1	R2	C1	C2	RC1	RC2
Best-known		11.83_1207.20	2.73_946.74	10_828.38	3_589.86	11.50_1383.12	3.25_1119.17
Tabu-ABC(2017)	[51]	13.75_1187.90	4.64_891.24	**10_828.38**	**3_589.86**	13.13_1361.08	5.50_1017.47
ACO-N(2011)	[10]	13.10_1213.16	4.60_952.30	10_841.92	3.3_612.75	12.70_1415.62	5.60_1120.37
D-VND(2021)	[53]	12.42_1214.02	**3.09_944.71**	**10_828.38**	**3_589.86**	12.13_1369.00	**3.38_1069.53**
RRGA(2020)	[12]	13.25_1180.66	5.54_878.64	**10_828.38**	**3_589.86**	12.75_1341.60	6.25_1004.35
MO TSP(2021)	[54]	12.75_1195.77	3.09_953.34	10_846.91	3_598.10	12.88_1373.06	4.00_1106.15
PDVA(2018)	[55]	12.92_1228.60	3.45_1033.53	10_828.38	3_590.60	12.73_1350.67	4.00_1081.87
HRRGA(2021)	[49]	12.25_1211.89	3.09_966.24	10_828.38	3_590.60	11.88_1360.19	4.00_1055.53
N-CLPSO		**12.42_1207.47**	3.00_956.75	10_830.62	**3_589.86**	**12.13_1358.86**	3.38_1108.78

Table 8 demonstrates that Wilcoxon signed ranks test is used to compare the PEav of the N-CLPSO algorithm and the other seven algorithms. In C1 and C2, N-CLPSO is not significantly different from the other seven algorithms. In R2 and RC2, N-CLPSO outperformed the other six algorithms except for D-VND et al. Finally, N-CLPSO outperformed all six algorithms in R1 and RC1, but there was no significant difference compared to HRRGA.

Table 8: The Wilcoxon non-parametric test of N-CLPSO and the other seven algorithms.

N-CLPSO vs	Tabu-ABC			ACO-N			D-VND			RRGA			MOTSP			PDVA			HRRGA		
Intances	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value	R^+	R^-	p-value
R1	11	1	**0.003**	9	3	**0.019**	11	1	**0.028**	9	3	**0.015**	8	2	**0.037**	10	1	**0.033**	6	6	0.583
R2	11	0	**0.003**	9	2	**0.008**	4	7	0.929	11	0	**0.003**	8	2	**0.047**	10	1	**0.004**	8	3	**0.041**
C1	0	2	0.180	2	2	0.500	0	2	0.180	0	2	0.180	4	2	0.366	0	2	0.180	0	2	0.180
C2	1	0	0.317	3	0	0.109	0	0	1.000	0	0	1.000	5	2	0.397	3	0	0.109	0	0	1.000
RC1	7	1	**0.017**	8	0	**0.012**	5	3	**0.048**	6	1	**0.042**	7	1	**0.017**	6	0	**0.028**	2	6	0.069
RC2	8	0	**0.012**	8	0	**0.012**	1	7	**0.036**	8	0	**0.012**	8	0	**0.013**	6	1	**0.043**	6	2	**0.036**

6. Conclusions and future research

In this paper, we propose the N-CLPSO algorithm for solving VRPTW. We discuss two objectives of VRPTW. The primary target is the number of vehicles, while the secondary target is the total distance traveled by the vehicles. The algorithm is based on improving the learning probability and learning exemplars of the original CLPSO. Furthermore, three novel strategies are introduced to improve the performance of N-CLPSO. The first one is using an insertion strategy to reduce the number of vehicles as much as possible. The second one is utilizing the guided reinsertion operator based on local information proposed in this paper to guide the neighborhood search. The last one is a diversity preservation strategy, the core concept of which is retaining the LCS elite fragment by comparing elite particles with current particles. Extensive experiments verify that N-CLPSO yield more promising and comprehensive performance other other 12 competitors. Concretely, the vehicle insertion allows the particles to reach almost the optimal number of vehicles. Moreover, the introduced reinsertion operator, which not only considers the insertion cost but also focuses on the time window information of the client, allows the algorithm to find better quality solutions in the local search. Last, the diversity retention strategy based on the LCS of elite fragments can guarantee particle quality as well as increase population diversity.

Although characteristics of the new proposed strategies in N-CLPSO have been discuss and analyze, coupling relationships between different strategy still need to be further figured out, because they are crucial for the algorithm to handles larger-scale problems. Moreover, N-CLPSO performs better in the “1” class problems who have narrow time windows, while there is a slight decrease in the overall performance of the “2” class problems who have longer time windows. This may because the insertion operator takes complete account of the time distance, which gives N-CLPSO better performance on problems with narrow time windows. Therefore, we will further explore the interrelationship between different strategies in the follow-up study. And enhance its performance on various VRPTW instances.

Acknowledgment

The study was funded by the Natural Science Foundation of Xinjiang Uygur Autonomous Regionthe(Grant No.: 2021D01A67), the National Natural Science Foundation of China (Grant Nos.: 61663009, 62106092), the Natural Science Foundation of Fujian Province (Grant Nos.: 2021J011 008, 2021J011007, 2022J01916), the Science and Technology Plan Projects of Zhangzhou (Grant Nos: ZZ2020J06, ZZ2020J24), and the Natural Science Foundation of Education Department of Jiangxi Province (Grant No.: GJJ200629).

Reference

[1] L. Wei, Z. Ma, N. Liu, Design of reverse logistics system for b2c e-commerce based on management logic of internet of things, Int. J. Ship. Trans. Log. 13 (2021) 484–497, https://doi.org/10.1504/IJSTL.2021.117274.

[2] G. B. Dantzig, J. H. Ramser, The truck dispatching problem, Manage. Sci. 6 (1959) 80–91, http://doi.org/10.1287/mnsc.6.1.80.

[3] B. Rabbouch, F. Saâdaoui, R. Mraihi, Empirical-type simulated annealing for solving the capacitated vehicle routing problem, J. Exp. Theor. Artif. Intell. 32 (2020) 437–452, https://doi.org/10.1080/0952813X.2019.1652356.

[4] D. S. Lai, O. C. Demirag, J. M. Leung, A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph, Transport. Res. E-log. 86 (2016) 32–52, https://doi.org/10.1016/j.tre.2015.12.001.

[5] L. Cruz-Reyes, et al., Ant colony system with characterization-based heuristics for a bottled-products distribution logistics system, J. Comput. Appl. Math. 259 (2014) 965–977, https://doi.org/10.1016/j.cam.2013.10.035.

[6] R. Baldacci, A. Mingozzi, R. Roberti, Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints, Eur. J. Oper. Res. 218 (2012) 1–6, https://doi.org/10.1016/j.ejor.2011.07.037.

[7] B. Kallehauge, Formulations and exact algorithms for the vehicle routing problem with time windows, Comput. Oper. Res. 35 (2008) 2307–2330, https://doi.org/10.1016/j.cor.2006.11.006.

[8] Y. Zhong, X. Pan, A hybrid optimization solution to vrptw based on simulated annealing, in: IEEE Int. Conf. Autom. Logist., 2007, pp. 3113 – 3117, http://dx.doi.org/10.1109/ICAL.2007.4339117.

[9] M. Alinaghian, E. B. Tirkolaee, Z. K. Dezaki, S. R. Hejazi, W. Ding, An augmented tabu search algorithm for the green inventory-routing problem with time windows, Swarm. Evol. Comput. 60 (2021) 100802, https://doi.org/10.1016/j.swevo.2020.100802.

[10] B. Yu, Z. Z. Yang, B. Z. Yao, A hybrid algorithm for vehicle routing problem with time windows, Expert Syst. Appl. 38 (2011) 435–441, https://doi.org/10.1016/j.eswa.2010.06.082.

[11] Y. Gong, J. Zhang, O. Liu, R. Huang, H. S. Chung, Y. Shi, Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach, IEEE Trans. Syst. Man Cybern. Part C 42 (2012) 254–267, https://doi.org/10.1109/TSMCC.2011.2148712.

[12] T. Khoo, B. B. Mohammad, V.-H. Wong, Y.-H. Tay, M. Nair, A two-phase distributed ruin-and-recreate genetic algorithm for solving the vehicle routing problem with time windows, IEEE Access 8 (2020) 169851–169871, https://doi.org/10.1109/ACCESS.2020.3023741.

[13] E. H. Houssein, A. G. Gad, K. Hussain, P. N. Suganthan, Major advances in particle swarm optimization: theory, analysis, and application, Swarm. Evol. Comput. 63 (2021) 100868, https://doi.org/10.1016/j.swevo.2021.100868.

[14] M. Chih, Stochastic stability analysis of particle swarm optimization with pseudo random number assignment strategy, Eur. J. Oper. Res. 305 (2023) 562–593, https://doi.org/10.1016/j.ejor.2022.06.009.

[15] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y. Zhang, Z. Zhan, Triple archives particle swarm optimization, IEEE Trans. Cybern. 50 (2020) 4862–4875, https://doi.org/10.1109/TCYB.2019.2943928.

[16] X. Xia, L. Gui, G. He, B. Wei, Y. Zhang, F. Yu, H. Wu, Z. Zhan, An expanded particle swarm optimization based on multi-exemplar and forgetting ability, Inf. Sci. 508 (2020) 105–120, https://doi.org/10.1016/j.ins.2019.08.065.

[17] X. Xia, Y. Tang, B. Wei, Y. Zhang, L. Gui, X. Li, Dynamic multi-swarm global particle swarm optimization, Computing 102 (2020) 1587–1626, https://doi.org/10.1007/s00607-019-00782-9.

[18] X. Xia, H. Song, Y. Zhang, L. Gui, X. Xu, K. Li, Y. Li, A particle swarm optimization with adaptive learning weights tuned by a multiple-input multiple-output fuzzy logic controller, IEEE T. Fuzzy. Syst. 11 (2022) 1–15, https://doi.org/10.1109/TFUZZ.2022.3227464.

[19] D. Yousri, D. Allam, M. Eteiba, P. N. Suganthan, Static and dynamic photovoltaic models parameters identification using chaotic heterogeneous comprehensive learning particle swarm optimizer variants, Energ Convers Manage 182 (2019) 546–563, https://doi.org/10.1016/j.enconman.2018.12.022.

[20] R. V. Kulkarni, G. K. Venayagamoorthy, Bio-inspired algorithms for autonomous deployment and localization of sensor nodes, IEEE Trans. Syst. Man Cybern. Part C 40 (2010) 663–675, https://doi.org/10.1109/TSMCC.2010.2049649.

[21] P. Kanakasabapathy, K. S. Swarup, Evolutionary tristate PSO for strategic bidding of pumped-storage hydroelectric plant, IEEE Trans. Syst. Man Cybern. Part C 40 (2010) 460–471, https://doi.org/10.1109/TSMCC.2010.2041229.

[22] R. V. Kulkarni, G. K. Venayagamoorthy, Particle swarm optimization in wireless-sensor networks: A brief survey, IEEE Trans. Syst. Man Cybern. Part C 41 (2011) 262–267, https://doi.org/10.1109/TSMCC.2010.2054080.

[23] S. S. M. Ajibade, M. O. Ogunbolu, R. Chweya, S. Fadipe, Improvement of population diversity of meta-heuristics algorithm using chaotic map, in: Lecture. Notes. Data Eng. Commun. Tech., 2022, pp. 95–104, https://doi.org/10.1007/978-3-030-98741-1_9.

[24] S. Cheng, Y. Shi, Q. Qin, Promoting diversity in particle swarm optimization to solve multimodal problems, in: Lect. Notes Comput. Sci., 2011, pp. 228–237, https://doi.org/10.1007/978-3-642-24958-7_27.

[25] S. Chourasia, H. Sharma, M. Singh, J. C. Bansal, Global and local neighborhood based particle swarm optimization, in: Adv. Intell. Sys. Comput., 2019, pp. 449 – 460, http://dx.doi.org/10.1007/978-981-13-0761-4_44.

[26] Z. Liu, Z. Qin, P. Zhu, H. Li, An adaptive switchover hybrid particle swarm optimization algorithm with local search strategy for constrained optimization problems, Eng. Appl. Artif. Intell. 95 (2020) 103771, https://doi.org/10.1016/j.engappai.2020.103771.

[27] Z. Wang, Z. Zhan, J. Zhang, An improved method for comprehensive learning particle swarm optimization, in:

IEEE Symp. Ser. Comput. Intell., 2015, pp. 218–225, https://doi.org/10.1109/SSCI.2015.41.

[28] H. Jiang, M. Lu, Y. Tian, J. Qiu, X. Zhang, An evolutionary algorithm for solving capacitated vehicle routing problems by using local information, Appl. Soft Comput. 117 (2022) 108431, https://doi.org/10.1016/j.asoc.2022.108431.

[29] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE T. Evolut. Comput. 20 (2016) 773–791, https://doi.org/10.1109/TEVC.2016.2519378.

[30] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006) 281–295, https://doi.org/10.1109/TEVC.2005.857610.

[31] W. Chen, J. Zhang, H. S. Chung, W. Zhong, W. Wu, Y. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evol. Comput. 14 (2010) 278–300, https://doi.org/10.1109/TEVC.2009.2030331.

[32] Y. Wang, L. Wang, Z. Peng, G. Chen, Z. Cai, L. Xing, A multi ant system based hybrid heuristic algorithm for vehicle routing problem with service time customization, Swarm Evol. Comput. 50 (2019) 100563, https://doi.org/10.1016/j.swevo.2019.100563.

[33] A. Gupta, S. Saini, An enhanced ant colony optimization algorithm for vehicle routing problem with time windows, in: Int. Conf. Adv. Comput., 2017, pp. 267 – 274, http://dx.doi.org/10.1109/ICoAC.2017.8441175.

[34] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, Y. Liu, A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows, Inf. Sci. 490 (2019) 166–190, https://doi.org/10.1016/j.ins.2019.03.070.

[35] W. Zhang, D. Yang, G. Zhang, M. Gen, Hybrid multiobjective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search for VRPTW, Expert Syst. Appl. 145 (2020) 113151, https://doi.org/10.1016/j.eswa.2019.113151.

[36] S. Reong, H. Wee, Y. Hsiao, 20 years of particle swarm optimization strategies for the vehicle routing problem: A bibliometric analysis, Mathematics 10 (2022) 3669, https://doi.org/10.3390/math10193669.

[37] P. Saksuriya, C. Likasiri, Hybrid heuristic for vehicle routing problem with time windows and compatibility constraints in home healthcare system, Appl. Sci. 12 (2022) 6486, https://doi.org/10.3390/app12136486.

[38] M. S. Sarbijan, J. Behnamian, Real-time collaborative feeder vehicle routing problem with flexible time windows, Swarm Evol. Comput. 75 (2022) 101201, https://doi.org/10.1016/j.swevo.2022.101201.

[39] N. Ding, J. Yang, Z. Han, J. Hao, Electric-vehicle routing planning based on the law of electric energy consumption, Mathematics 10 (2022) 3099, https://doi.org/10.3390/math10173099.

[40] R. Liu, Z. Jiang, A hybrid large-neighborhood search algorithm for the cumulative capacitated vehicle routing problem with time-window constraints, Appl. Soft Comput. 80 (2019) 18–30, https://doi.org/10.1016/j.asoc.2019.03.008.

[41] M. Alinaghian, M. Jamshidian, E. B. Tirkolaee, The time-dependent multi-depot fleet size and mix green vehicle routing problem: improved adaptive large neighbourhood search, Optimization 71 (2022) 3165–3193, https://doi.org/10.1080/02331934.2021.2010078.

[42] M. Qi, W. Lin, N. Li, L. Miao, A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows, Transport. Res. E-Log. 48 (2012) 248–257, https://doi.org/10.1016/j.tre.2011.07.001.

[43] M. Chih, Three pseudo-utility ratio-inspired particle swarm optimization with local search for multidimensional knapsack problem, Swarm Evol. Comput. 39 (2018) 279–296, https://doi.org/10.1016/j.swevo.2017.10.008.

[44] J. Shi, Q. Zhang, E. P. K. Tsang, EB-GLS: an improved guided local search based on the big valley structure, Memetic Comput. 10 (2018) 333–350, https://doi.org/10.1007/s12293-017-0242-5.

[45] L. Hong, An improved LNS algorithm for real-time vehicle routing problem with time windows, Comput. Oper. Res. 39 (2012) 151–163, https://doi.org/10.1016/j.cor.2011.03.006.

[46] X. Xia, H. Qiu, X. Xu, Y. Zhang, Multi-objective workflow scheduling based on genetic algorithm in cloud environment, Inf. Sci. 606 (2022) 38–59, https://doi.org/10.1016/j.ins.2022.05.053.

[47] M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (1987) 254–265, https://doi.org/10.1287/opre.35.2.254.

[48] D. Wu, M. Dong, H. Li, F. Li, Vehicle routing problem with time windows using multi-objective co-evolutionary approach, Int. J. Simul. Model. 15 (2016) 742–753, https://doi.org/10.2507/IJSIMM15(4)CO19.

[49] T. Khoo, M. B. Bonab, The parallelization of a two-phase distributed hybrid ruin-and-recreate genetic algorithm for solving multi-objective vehicle routing problem with time windows, Expert Syst. Appl. 168 (2021) 114408, https://doi.org/10.1016/j.eswa.2020.114408.

[50] G. D. Konstantakopoulos, S. P. Gayialis, E. P. Kechagias, G. A. Papadopoulos, I. P. Tatsiopoulos, A multiobjective large neighborhood search metaheuristic for the vehicle routing problem with time windows, Algorithms 13 (2020) 243, https://doi.org/10.3390/a13100243.

[51] D. Zhang, S. Cai, F. Ye, Y. Si, T. T. Nguyen, A hybrid algorithm for a vehicle routing problem with realistic

constraints, Inf. Sci. 394 (2017) 167–182, https://doi.org/10.1016/j.ins.2017.02.028.
[52] Y. Shen, M. Liu, J. Yang, Y. Shi, M. Middendorf, A hybrid swarm intelligence algorithm for vehicle routing problem with time windows, IEEE Access 8 (2020) 93882–93893, https://doi.org/10.1109/ACCESS.2020.2984660.
[53] Y. Lan, F. Liu, W. W. Ng, J. Zhang, M. Gui, Decomposition based multi-objective variable neighborhood descent algorithm for logistics dispatching, IEEE Trans. Emerging Topics Comp. Intell 5 (2020) 826–839, https://doi.org/10.1109/TETCI.2020.3002228.
[54] Z. He, K. Zhou, H. Shu, X. Chen, X. Lyu, Multi-objective algorithm based on tissue p system for solving tri-objective optimization problems, Evol. Intell. (2021) 1–16https://doi.org/10.1007/s12065-021-00658-y.
[55] W. Dong, K. Zhou, H. Qi, C. He, J. Zhang, A tissue p system based evolutionary algorithm for multi-objective vrptw, Swarm Evol. Comput. 39 (2018) 310–322, https://www.sciencedirect.com/science/article/pii/S2210650216305867.
[56] K. C. Tan, Y. H. Chew, L. H. Lee, A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows, Comput. Optim. Appl. 34 (2006) 115–151, https://doi.org/10.1007/s10589-005-3070-3.
[57] Y. Rochat, É. D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, J. Heuristics 1 (1995) 147–167, https://doi.org/10.1007/BF02430370.
[58] H. Li, A. Lim, Local search with annealing-like restarts to solve the VRPTW, Eur. J. Oper. Res. 150 (2003) 115–127, https://doi.org/10.1016/S0377-2217(02)00486-1.
[59] D. Mester, An evolutionary strategies algorithm for large scale vehicle routing problem with capacitate and time windows restrictions, in: proceedings of the conference on mathematical and population genetics, University of Haifa, Israel, 2002.
[60] P. Shaw, A new local search algorithm providing high quality solutions to vehicle routing problems, in: Technical Report, Department of Computer Science, University of Strathclyde, Scotland, 1997.
[61] J. Berger, M. Barkaoui, O. Bräysy, A route-directed hybrid genetic approach for the vehicle routing problem with time windows, INFOR: Information Systems and Operational Research 41 (2003) 179–194, https://doi.org/10.1080/03155986.2003.11732675.
[62] J. Homberger, H. Gehring, Two evolutionary metaheuristics for the vehicle routing problem with time windows, INFOR: Information Systems and Operational Research 37 (1999) 297–318, https://doi.org/10.1080/03155986.1999.11732386.
[63] L. Rousseau, M. Gendreau, G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, J. Heuristics 8 (2002) 43–58, https://doi.org/10.1023/A:1013661617536.
[64] L. M. Gambardella, É. Taillard, G. Agazzi, Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows, in: New Ideas in Optimization, McGraw Hill, London, UK, 1999, pp. 63–76.
[65] R. Bent, P. Van Hentenryck, A two-stage hybrid local search for the vehicle routing problem with time windows, Transport. Sci. 38 (2004) 515–530, https://doi.org/10.1287/trsc.1030.0049.
[66] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, Record breaking optimization results using the ruin and recreate principle, J. Comput. Phys. 159 (2000) 139–171, https://www.sciencedirect.com/science/article/pii/S0021999199964136.
[67] A. L. Bouthillier, T. G. Crainic, A cooperative parallel meta-heuristic for the vehicle routing problem with time windows, Comput. Oper. Res. 32 (2005) 1685–1708, https://www.sciencedirect.com/science/article/pii/S0305054803003654.
[68] J. Homberger, Verteilt-parallele metaheuristiken zur tourenplanung, Wiesbaden: Deutscher Universitatsverlag (2000).
[69] K. Ghoseiri, S. F. Ghannadpour, Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm, Appl. Soft Comput. 10 (2010) 1096–1107, https://doi.org/10.1016/j.asoc.2010.04.001.
[70] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, A tabu search heuristic for the vehicle routing problem with soft time windows, Transport. Sci. 31 (1997) 170–186, https://doi.org/10.1287/trsc.31.2.170.
[71] Z. Fu, R. Eglese, L. Y. O. Li, A unified tabu search algorithm for vehicle routing problems with soft time windows, J. Oper. Res. Soc. 59 (2008) 663–673, https://doi.org/10.1057/palgrave.jors.2602371.
[72] Z. Czech, P. Czarnas, Parallel simulated annealing for the vehicle routing problem with time windows, in: Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002, pp. 376–383, https://doi.org/10.1109/EMPDP.2002.994313.
[73] T. Ibaraki, S. Imahori, M. Kubo, T. Masuda, T. Uno, M. Yagiura, Effective local search algorithms for routing and scheduling problems with general time-window constraints, Transport. Sci. 39 (2005) 206–232, https://doi.org/10.1287/trsc.1030.0085.