Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

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

Qichao Wu (872793924@qq.com)
Minnan Normal University https://orcid.org/0000-0002-7399-8524
Xuewen Xia
Minnan Normal University https://orcid.org/0000-0002-4938-1479
Haojie Song
Minnan Normal University
Xing Xu
Minnan Normal University
Yinglong Zhang
Minnan Normal University
Fei Yu
Minnan Normal University
Hongrun Wu
Minnan Normal University

Research Article

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

Posted Date: January 9th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2359566/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

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

Qichao Wu[1,2], Xuewen Xia[1,2*], Haojie Song[1,2†], Xing Xu[1,2†], Yinglong Zhang[1,2†], Fei Yu[1,2†] and Hongrun Wu[1,2†]

[1]College of Physics and Information Engineering, Minnan Normal University, Zhangzhou, 363000, China.
[2]Key Lab of Intelligent Optimization and Information, Minnan Normal University, Zhangzhou, 363000, China.

*Corresponding author(s). E-mail(s): xwxia@whu.edu.cn;
Contributing authors: 872793924@qq.com;
†These authors contributed equally to this work.

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. We calculate an information matrix ($\boldsymbol{IM}$) recording the probability of adjacency between two clients based on the information about the clients themselves and the local information about the elite individuals to guide the removal operation of the neighborhood search. At the same time, we combine the cost matrix ($\boldsymbol{CM}$) that records the cost of customer removal with $\boldsymbol{IM}$ to create a guided reinsertion operator based on local information to guide the routing. Moreover, to enhance the exploration of N-CLPSO, a semi-random disturbance strategy is proposed. To prevent degradation of the population, the longest common sequences of elites are saved when performing the disturbance. To illustrate the effectiveness of N-CLPSO, this paper conducts extensive experiments

on Solomon's benchmark sets. The numerical results show that the proposed algorithm outperforms or can compete with many other state-of-the-art algorithms. Furthermore, sensitivity analysis of the newly introduced strategies is also conducted based on extensive experiments.

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

1 Introduction

The vehicle routing problem (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 (Dantzig and Ramser, 1959). Since there is a high correlation between VRP and the reality of logistics, the VRP attracts much attention of researches in the logistics field. During the last few years, various variants of VRP, such as the Capacitated VRP (CVRP) (Rabbouch et al, 2020) and the Heterogeneous Fleet VRP (HFVRP) (Lai et al, 2016), have derived from different real-world problems. One of the most significant variations is the VRP with time windows (VRPTW) (Yu et al, 2011a), in which users' access time and vehicles' capacity limits are considered in VRP aiming to make the problem more realistic.

Initially, some deterministic algorithms are adopted to solve VRPTW (Baldacci et al, 2012; Kallehauge, 2008), but due to VRPTW's NP-hardness, solving large-scale VRPTW with the deterministic 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) (Zhong and Pan, 2007), Taboo Search (TS) (Ho and Haugland, 2004), Ant Colony Optimization (ACO) (Cruz-Reyes et al, 2014), Particle Swarm Optimization (PSO) (Gong et al, 2012), and Genetic Algorithm (GA) Nalepa and Blocho (2015) 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 (Poli et al, 2007), as an excellent heuristic proposed by Kennedy and Eberhart in 1995, was initially widely used for continuous function optimization and achieved many promising achievements in both theoretical studies (Wei et al, 2020; Xia et al, 2020b,a,c) and engineering applications (Veeramachaneni et al, 2005; Lin et al, 2009; Kulkarni and Venayagamoorthy, 2010; Kanakasabapathy and Swarup, 2010; Kulkarni and Venayagamoorthy, 2011). In recent years, some researchers have tried to improve the basic PSO algorithm to adapt it to combinatorial optimization problems, including VRPTW. 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 (Ajibade et al, 2022; Cheng et al, 2011). To overcome the latter drawback of the PSO algorithm, various excellent and efficient local search strategies have been applied to some PSO variants (Chourasia et al, 2019; Liu et al, 2020). 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, how to improve these strategies in PSO and adapt them to VRPTW still needs to be further studied.

Based on the analysis above mentioned, this paper proposes a neighborhood comprehensive learning PSO (N-CLPSO) to solve VRPTW. In N-CLPSO, update operators of the velocity and position applied in traditional PSO are improved to satisfy characteristics of VRPTW. Based on the update operators, we design a novel neighborhood search operator to enhance the local search capability of N-CLPSO. Moreover, intending to maintain population diversity, a diversity retention strategy based on elite fragments is introduced in N-CLPSO. Properties of introduced strategies in N-CLPSO is analyzed by systematic experiments, while the overall performance of N-CLPSO is testified by comparison results between it and other state-of-the-art metaheuristics on a large number of benchmark VRPTW instances.

The main contributions of this study are summarized below.

(1) PSO has been widely used in the field of VRPTW, but its lack of local search capability leads to the solution quality being often unsatisfactory. To remedy this deficiency, we propose a novel reinsert operator and two remove operators with the help of the remove-reinsert idea of Large Neighborhood Search (Hong, 2012). We combine the novel remove-reinsert-based neighborhood search with CLPSO and propose N-CLPSO.

(2) In previous reinsert operators, most of them only consider the incremental repair cost after node insertion, which tends to ignore other useful information. Therefore, we propose an efficient repair operator, in which not only considers the local information of the location between clients, time windows and elite segments, but also evaluates the incremental cost caused by the insertion of client points into the current location.

(3) To overcome the premature convergence, this study proposes a diversity retention strategy based on semi-random disturbance of elite fragments. In order to ensure the diversity of particles, a certain scale of reorder operations on the original routes should be performed. However, a large-scale random reorder operations may lead to particle degradation. Therefore, we design a new reorder strategy, in which the longest common sequences of elite particles are saved. Based on the strategy, the promising gene blocks, i.e., the longest common sequences, of the elites, can be saved when executing the reorder operator.

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 current state of VRPTW research. N-CLPSO and the main strategies are detailed in Section 4. Next, the experimental results and analysis 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 tth 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 tth 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 speed on the latest speed; 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 (2006) proposed a CLPSO in which a new speed 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 jth 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 (Chen et al, 2010) 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 consumers in a given geographic area 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. 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 customer point v_i is located at (x_i, y_i), its the demand for goods is q_i and the delivery time window $[b_i, l_i]$, 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 $[0, l_0 \geq \max(e_i)]$. For simplicity, the time cost that a vehicle traveling from customer i to customer is represented by the Euclidean distance between nodes $(d_{i,j} = d_{j,i})$, where $i \neq j$, $i, j \in \{1, 2, \ldots, M\}$.

VRPTW has two objectives 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. Define variable:

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

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

The goal of the VRPTW is to minimize

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

and

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

s.t.

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

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

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

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

$$\sum_{k=1}^{K} y_{0}^{k} = K \tag{11}$$

$$t_{i} + w_{i} + s_{i} + d_{i,j} = t_{j} \quad \forall i, j = 1, \ldots M, i \neq j \tag{12}$$

$$e_{i} \leq t_{j} \leq l_{j} \quad \forall j = 1, \ldots M \tag{13}$$

$$w_{i} = \max\{e_{i} - t_{i}, 0\} \quad \forall i = 1, \ldots M \tag{14}$$

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. Eqs.(12)-(14) define the time window constraint, where t_i is the vehicle arrival time at node i; w_i is the vehicle waiting time at the customer location to start the service time; S_i is the service time; and $d_{i,j}$ 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 et al (2019) 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

and Saini (2017) 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 et al (2019) 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 problems. Gocken et al (2017) proposed a hybrid version of the GA that performs a scan calculation for the initial population, allowing the algorithm to start searching directly for high-quality solutions to produce more has been solutions. Moon et al (2012) proposed a model of the VRPTW with overtime and outsourcing vehicles (VRPTWOV) by extending the conventional VRPTW model, which allows drivers to appear to work overtime and use outsourcing vehicles. Thus, an integer programming model, a GA and a hybrid algorithm based on simulated annealing are proposed to solve the VRPTWOV problem. Zhang et al (2020) 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).

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 have started to try to improve the standard PSO to make it applicable to the optimization of combinatorial problems, such as VRPTW. Marinakis et al (2019) proposed a Multi-Adaptive PSO (MAPSO), which improves the traditional PSO in three aspects: initialization, position update and hyperparameters. Moreover, two memories i.e., global best memory (GBM) and personal best memory (PBM), are proposed in MAPSO, to update the particles position information. Zhang et al (2018) proposed an evolutionary scattering search PSO (ESS-PSO) to solve VRPTW, which introduces a GA and a new "route +/-" evolutionary operator. The algorithm redefines the velocity and position update rules based on the concept of "destroy and rebuild".

Moreover, some studies on VRPTW have indicated that an efficient local search strategy has become a useful method to help individuals to jump out of local optima. Therefore, designing efficient local search strategies becomes one of research hot spots on VRPTW, and attract more attentions of researchers in recent years. For instance, Liu and Jiang (2019) 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 et al (2011b) introduced a neighborhood search operator on top of the ACO and borrow the population diversity protected by forbidden search and explore new solutions.

4 Proposed method

Since VRPTW is an NP-hard problem with high computational complexity and many conditional constraints, this study proposes N-CLPSO to solve VRPTW. 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 tth 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 tth 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) $d \in (0, 1, 2, \ldots, n)$, n represents the city number; (2) $k, l \in \{0, 1, \ldots, d-1, d+1, \ldots, n\}$ and (3) $k \neq l$. Constraint (1) ensures that each element is a valid city, constraint (2) makes each city will not be adjacent to itself, and constraint (3) enables that each city's neighbouring points are not duplicated. Taking a VRPTW problem 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: $(0_{1,2}, 1_{3,0}, 2_{0,4}, 3_{4,1}, 4_{2,3})$. 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. N-CLPSO is the framework of CLPSO with the addition of 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 the feasible solution. Then, we insert two start conditions after the steps of vehicle insertion strategy and individual optimal solution *Pbest* update to determine whether the diversity and neighborhood search strategies are started or not, respectively. Finally, after updating the global optimal solution *Gbest*, N-CLPSO continues with the next iteration until the stopping condition is reached.

It has been pointed out in Section 2 that VRPTW has two objectives. In this study, we combine a new decision method (Gong et al, 2012) to deal with the number of vehicles NV and the total distance TD of VRPTW. To better

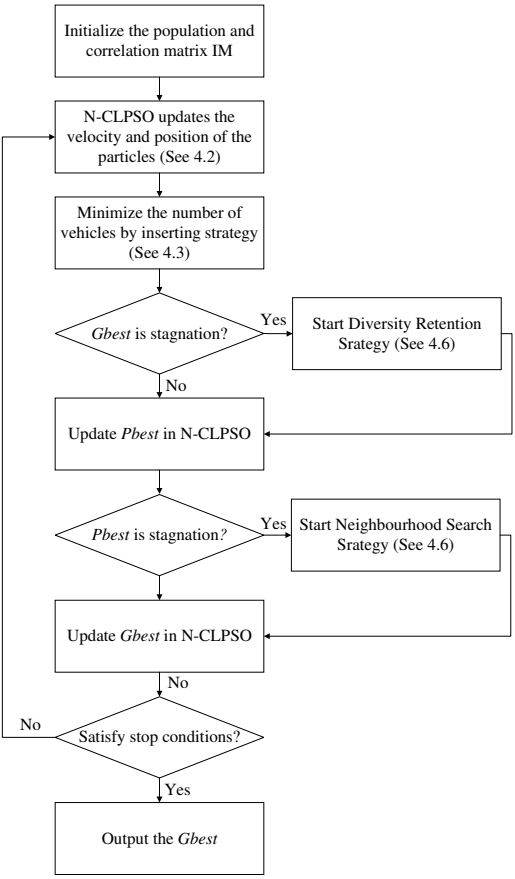

Fig. 1 Framework of N-CLPSO

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, inspired by the literature (Gong et al, 2012),we define new operators to update the position and velocity of each particle on the set and

probability. The proposed 4 operators 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:

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

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

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

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

Taking the velocity update Eq.(3) as an example, 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 convergence of the algorithm, this study modifies the traditional CLPSO with certain improvements, shown as Eq.(21) and Eq.(22).

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

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

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 sample particles.

Unlike the traditional CLPSO, in which a particle selects its learning exemplar based on its index, a particle in N-CLPSO adjusts it learning rate and the number of learning samples based on the its fitness values. Concretely, the higher the particle fitness is, the smaller its selection probability Pc and the number of learning examples n will be, thus ensuring the fitness of its particles. On the contrary, if the particles fitness value is lower, both Pc and n will become larger to make the particles have a greater probability to learn from

other particles. In other words, it will have a greater probability to learn from other outstanding particles, which is beneficial for speeding up the convergence.

Based on the velocity generated by the above equation, the position of the particle can be obtained by three set: $S_V = \{m \mid < k, m > \in V_i$, and $< k, m >$ satisfies $\Omega\}$, $S_x = \{m \mid < k, m > \in X_i^t$, 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^t and A respectively, while satisfying the constraints. First, 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. Second, the set X_i^t and A in the set of all possible arcs in the search space since the position of the previous generation of particles and the set of all possible arcs in the search space, respectively.

As in Algorithm 1, assume that the new location X_i^{t+1} is set to the empty set, and according to the constraint, each vehicle departs from the depot and iteratively selects the next customer node adjacent to the current customer. 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} and the search continues with m as the current client until all client visits are complete. When there are no available nodes in all S_v, S_x, and S_a, it indicates that the constraints of VRPTW cannot be satisfied, so we need to create a new path. 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} goes to replace the current position X_i^{t+1}. In addition, we also use the heuristic selection method NNH (Gong et al, 2012) to speed up the convergence of the algorithm.

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 subpaths, i.e., $Vc(1) = \{1, 2, 3, 1\}$, $Vc(2) = \{1, 5, 6, 1\}$, and $Vc(3) = \{1, 4, 7, 1\}$. First, we remove subpath $Vc(1)$ 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-paths $Vc(1)$ 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-path in the

Algorithm 1 Pseudocode for the position update process in N-CLPSO

Input: X_i^t; V_i; A;
Output: X_i^{t+1}

$X_i^t = \phi; k = 0; t = 0;$
$S_V = \{m \mid < k, m > \in V_i, \text{ and } < k, m > \text{ satisfies } \Omega\};$
$S_x = \{m \mid < k, m > \in X_i^t, \text{ and } < k, m > \text{ satisfies } \Omega\};$
$S_a = \{m \mid < k, m > \in A, \text{ and } < k, m > \text{ satisfies } \Omega\};$
while Customers do not have complete access **do**:
    if $\boldsymbol{S_V \neq \phi}$ **then**
        select m in S_V, and add $< t, m >$ to X_i^{t+1}
        $k = m; t = m;$
        update S_V,S_x,S_a;
    else if $\boldsymbol{S_x \neq \phi}$ **then**
        select m in S_x, and add $< t, m >$ to X_i^{t+1}
        $k = m; t = m;$
        update S_V,S_x,S_a;
    else if $\boldsymbol{S_a \neq \phi}$ **then**
        select m in S_a, and add $< t, m >$ to X_i^{t+1}
        $k = m; t = m;$
        update S_V,S_x,S_a;
    else
        $k = 0;$
        update S_V,S_x,S_a;
    end if
end while

Algorithm 2 Vehicle insertion strategy

Input: Routing r, Vehicle Collection Vc
Output: New Routing Nr

$n = number(Vc)$; //Record total number of vehicles
$i = 1;$
while $i <= n$ **do**:
    $ins = Vc(i)$; //Add the node with vehicle i to the set to be inserted
    $r^* = r - Vc(i)$; //Remove the vehicle i from the path r
    Insert ins into r^* using Algorithm 3;//See section 4.4
    if ins all inserted successfully **then**
        $r = r^*;$
        $n = n - 1;$
    else
        $r = r;$
        $i = i - 1;$
    end if
end while
$Nr = r;$

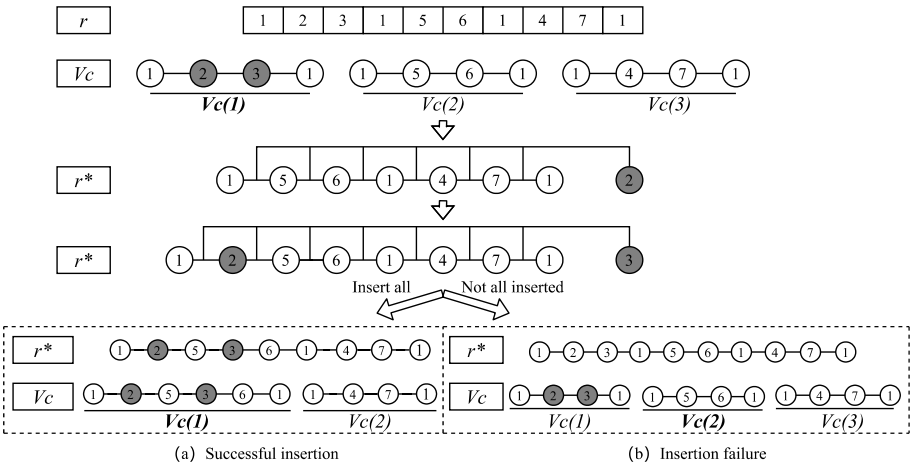

Fig. 2 Vehicle insertion diagram

state before insertion, and removes path $Vc(2)$. Such operations are repeated until all sub-paths being visited.

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 (Jiang et al, 2022), 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, $IM_{i,j}^{t}$, defined as Eq.(23), denotes the probability that customers i and j can be served by the same car consecutively.

$$IM_{i,j}^{t} = (1 - a_t) * (DST_{\max} - DST_{i,j}) + a_t * CT \tag{23}$$

From Eq.(23) we can see that $IM_{i,j}^{t}$ 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 (Qi et al, 2012) 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.

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

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).

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

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 promising offspring. Since these individuals usually have better fitness values, it seems that they may be closer to the optimal solution and their common fragment is more likely to be part of the optimal solution. Thus, if two adjacent customers appear frequently in elite individuals, the probability of them being served by the same vehicle in the optimal solution will also be high.

During the search process, the excellent genes of elite individuals will gradually spread to the whole population. If the diffusion rate of the favorable genes is fast, it is beneficial to improve the convergence speed of the algorithm, but it is may cause the population fall into local optimum easily. Conversely, if the diffusion rate is slow, it is beneficial to maintain the population diversity, but it will reduce the convergence speed of the algorithm. Therefore, a linearly decreasing coefficient a_t, defined as Eq.(28), is introduced in this study to adjust the diffusion rate of good.

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

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 the optimization process, the value of a_t is small. In this case, the diffusion rate of excellent genes of elite individuals is slow, which is conducive to preserving population diversity and enhancing the global search ability of the population. On the contrary, a larger a_t at the later optimization process can bring a higher diffusion rate of excellent genes, which is beneficial for the speed 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 tth 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 inserting node i 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.

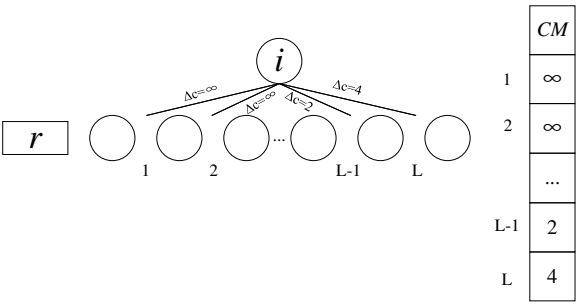

Fig. 3 CM building process

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. See Algorithm 3 for a guided reinsertion operator based on local information.

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

$Nr = r$;
for $i = 1$ to ins **do** // ins denotes the number of customers to be inserted
  $l \leftarrow$ Calculate the path r length;
  $stay \leftarrow$ Record the location of the path r repository;
  $Pd \leftarrow ins(i)$;//$InsertNode$
  $CM \leftarrow$ Create a two-dimensional matrix with initial value ∞ for $l * l$;
  $j \leftarrow 1$;
  while $j < l$ **do** //Calculate CM
    if the node Pd insertion is overweight **then**
      $j = stay(\text{find}(stay == j) + 1)$; // go to the next path
    else
      if Pd can be inserted in the current position **then**
        $Pi = Nr(j)$;// Predecessor Nodes
        $Pj = Nr(j + 1)$;// Post nodes
        // Calculating Cost Increment
        $CM(j) = Dis(Pi, Pd) + Dis(Pd, Pj) - Dis(Pi - Pj)$;
        $j = j + 1$;
      end if
    end if
    if Pd has no position where it can be inserted **then**
      $IX = []$;
    else
      $IM \leftarrow$ Sorting node Pd in ascending order
      $CM \leftarrow$ Sort the cost matrix of node Pd in descending order;
      $IX \leftarrow$ Find the number of the lowest position in the sum of IM and CM rankings;
    end if
    if IX is empty **then**
      $Nr \leftarrow$ Insert Pd and repository at the end of Nr;
    else
      $Nr \leftarrow$ Insert Pd at the location of IX;
    end if
  end while
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 (Chih, 2018). Furthermore, the study in (Shi et al, 2018) 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*.

Subject to the ideas of LNS (Hong, 2012) removal-reinsert, this section introduces 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 from the original route 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. It can be seen that when the optimal solution has more generations-stagnancy, there are more customers should be removed. Thus, the search range of the neighborhood can be increased. It is worth noting that the number of removed customers D set in this paper must not exceed $N/10$ because a too large D not only increases the computational effort, but also causes the inability to find the best insertion position for nodes. Two removal strategies proposed in this study are described below.

(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).

(2) We propose a removal operator on removal cost 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 4 Removal strategy based on IM

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

Algorithm 5 Removal strategy based on removal cost

H
Input:
// Enter the route, distance matrix and node to be deleted, respectively
r;Dis;D;
Output: Delete node set Nt
$Nt = \phi$;
$L = r.length$;
for $x = 2 : L - 1$ **do** // Calculate the removal cost Δc for each customer point removed
    $F = r(x - 1)$; // Precursor node of node x
    $C = r(x)$; // Node x
    $R = r(x + 1)$; // Posterior node of node x
    $\Delta c(x) = Dis(F, C) + Dis(C, R) - Dis(F, R)$;
end for
$k = 0$;
while $k < D$ **do**
    Randomly select the customer i in Δc where Nt does not appear by roulette;
    $Nt = Nt \cup \{i\}$;
    $k = k + 1$;
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 longest common subsequence (LCS) (Xia et al, 2022), 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.

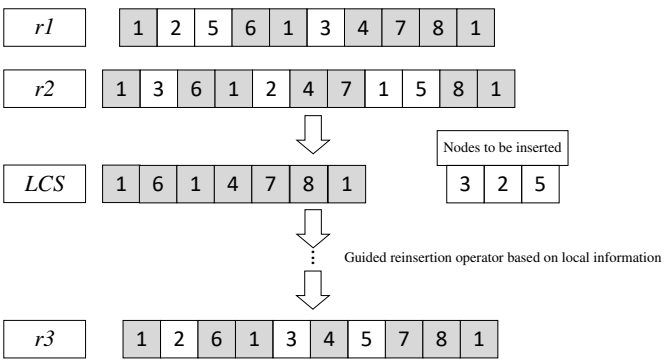

Fig. 4 Illustrative Example of Diversity Strategy

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 $O(n * m)$.

5 Experimental evaluation

In this section, Section 5.1 describes the general setup of experiments while Section 5.2 presents an introduction to the dataset used in the experiments. In Section 5.3, a sensitivity analysis of each component in N-CLPSO is presented. Section 5.4 gives the experimental results and corresponding discussions.

5.1 Setup

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 is set to 2.0. In this paper, the refresh gap "rg" of CLPSO is set according to (Liang et al, 2006). The learning probability Pc and the sample selection method are used in Eq.(21) and Eq.(22). 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 Datasets

N-CLPSO is tested on a classical benchmark of 56 VRPTW problem instances proposed by Solomon (1987) 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).

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 speed selection strategy and vehicle insertion strategy to speed 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 speed selection 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 speed 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 speed than "CLPSO". Therefore, we can see that the new speed selection strategy and the vehicle insertion strategy play positive performance on improving the convergence speed.

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 of the datasets, in terms of the percentage error between the best stroke length and the best-known stroke length for each algorithm, are shown in Fig.6. 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, and it is clear that the optimal results were obtained by adding both the neighborhood search and diversity retention strategies to N-CLPSO.

In N-CLPSO, we used a combination of the information matrix IM and the cost matrix 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 CB. Experimental results demonstrated in Fig.7 verify that using CB is better than using only IM in R101, R201, RC101 and RC201 problems, while using IM is better than using only CB in the set clustering dataset C104 and C204, it is clear that using both CM and IM has the best performance.

5.4 Comparison with other algorithms

To verify the comprehensive performance of N-CLPSO, we compare it with several state-of-the-art methods for solving VRPTW. Table 1 summarizes the basic information of the selected peer algorithms. We demonstrate the effectiveness of N-CLPSO by comparing it with the best known results and the

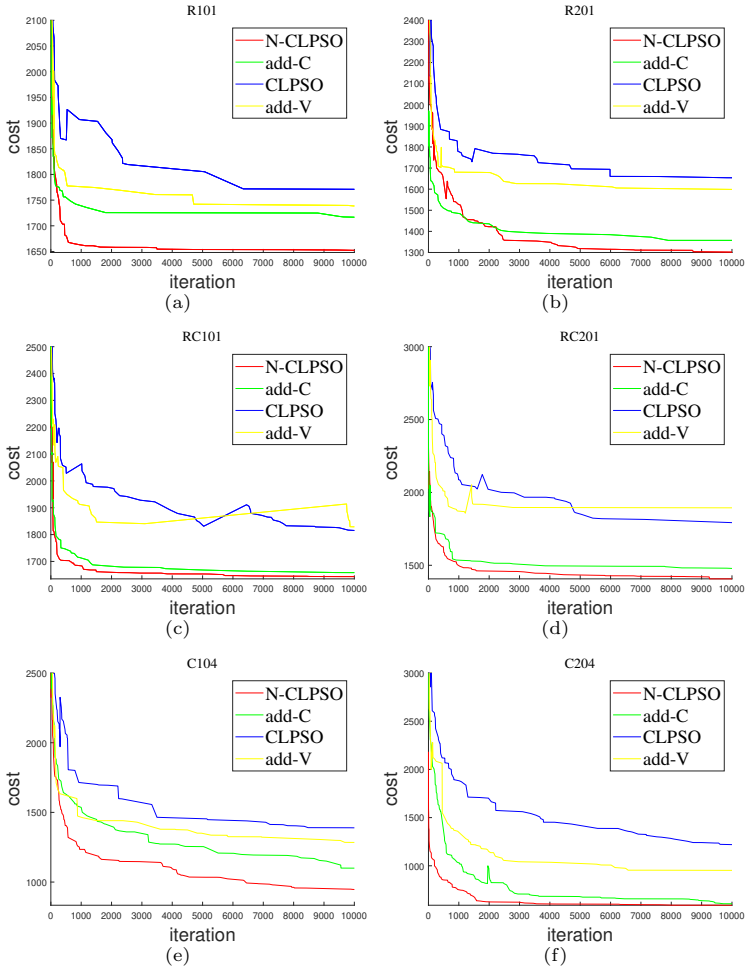

Fig. 5 Contribution of speed selection strategy and vehicle insertion strategy

other 7 peer algorithms, and finally giving the results of the best average operation for each subclass (C1, C2, R1, R2, RC1 and RC2).

In Table 2, 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 denoting the percentage deviation of the algorithm from the best-known result is used to measure the solutions quality of the algorithm. "Std" is the standard deviation of multiple solutions achieved by an algorithm, which is used to measure the stability of the algorithm.

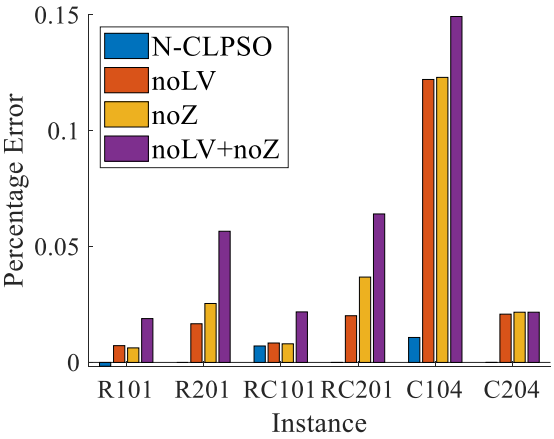

Fig. 6 Contribution of Neighborhood Search and Diversity retention Strategies

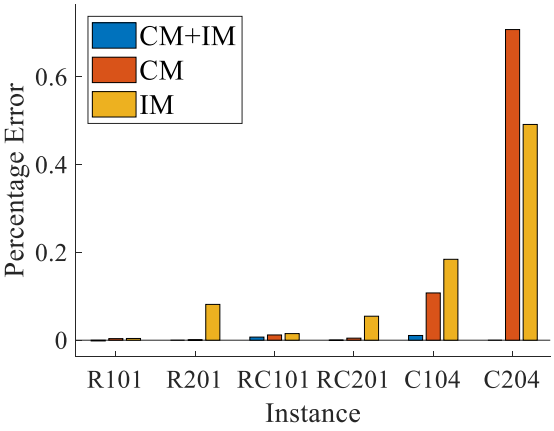

Fig. 7 Contribution of correlation matrix and cost matrix

It can be seen from Table 2 that the N-CLPSO algorithm obtains a new more optimal solution on R101 and reaches 23 optimal solutions on other tested problems. The average deviations of R1, C1 and RC1 are 0.45%, 0.27% and 0.43%, respectively, which are less than 0.5%. 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. There is no standard deviation for class C2, and only C103 and C104 have 1.08 and 4.82 for class C1. the average standard deviations of R1, R2, RC1, RC2 and the total data set are 8.85, 12.19, 6.88 and 12.75, respectively, and most of the standard deviations are below 10. 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

Table 1 Information about the comparison algorithm

Symbol	Method	References
LNS	Large neighborhood search	Hong (2012)
CPSO	Hybrid chaos-particle swarm optimization	Hu et al (2013)
AC-CH	Ant colony with characterization heuristics	Cruz-Reyes et al (2014)
S-PSO	set-based particle swarm optimization	Gong et al (2012)
BSO	hybrid swarm intelligence algorithm by hybridizing Ant Colony System and Brain Storm Optimization algorithm	Shen et al (2020)
Tabu-ABC	hybrid approach by combining Tabu search and the artificial bee colony algorithm.	Zhang et al (2017)
MOLNS	Multiobjective Large Neighborhood Search (MOLNS) algorithm	Konstantakopoulos et al (2020)

there is a small decrease in the performance of N-CLPSO in the "2" class problems who have longer time windows.

Table 2: Comparison with the best-known results.

Best-known			N-CLPSO					
Dataset	NV	TD	BNV	BTD	MNV	MTD	Deviation	Std
R101	19	1650.8	**19**	**1648.08**	19	1650.5	-0.16%	3.42
R102	17	1486.12	**17**	**1486.12**	17	1493.12	0	7.05
R103	13	1292.68	13	1299.99	13	1312.13	0.56%	17.66
R104	9	1007.31	10	996.27	10	1021.06	-	15.54
R105	14	1377.11	**14**	**1377.11**	14	1379.35	0	3.17
R106	12	1252.03	12	1262.80	12	1264.5	0.86%	2.40
R107	10	1104.66	11	1081.17	11	1107.85	-	2.40
R108	9	960.88	10	985.76	10	988.59	-	2.45
R109	11	1194.73	11	1210.73	11.7	1196.21	1.32%	-
R110	10	1118.84	11	1101.49	11	1114.61	-	10.26
R111	10	1096.72	11	1064.67	11	1072.80	-	7.78
R112	9	982.14	10	974.95	10	982.27	-	8.85
R201	4	1252.37	**4**	**1252.37**	4	1258.83	0	7.52
R202	3	1191.70	3	1225.02	3.5	1178.49	2.72%	-
R203	3	939.50	3	962.25	3	976.32	2.36%	9.06
R204	2	825.52	3	766.13	3	777.83	-	10.92
R205	3	994.43	3	1027.79	3	1051.16	3.25%	14.61
R206	3	906.14	3	939.46	3	947.38	3.55%	6.53
R207	2	890.61	3	872.40	3	877.56	-	5.90
R208	2	726.82	2	740.36	2	772.04	1.83%	28.30
R209	3	909.16	3	943.72	3	957.46	3.66%	10.13
R210	3	939.37	3	965.88	3	986.43	2.74%	17.00
R211	2	885.71	3	828.90	3	842.35	-	11.88
C101	10	828.94	**10**	**828.94**	10	828.94	0	0
C102	10	828.94	**10**	**828.94**	10	830.15	0	0
C103	10	828.06	10	839.35	10	840.20	1.35%	1.08
C104	10	824.78	10	833.67	10	837.08	1.07%	4.82
C105	10	828.94	**10**	**828.94**	10	828.94	0	0

Table 2: (Continued.)Comparison with the best-known results.

Best-known			N-CLPSO					
Dataset	NV	TD	BNV	BTD	MNV	MTD	Deviation	Std
C106	10	828.94	**10**	**828.94**	10	828.94	0	0
C107	10	828.94	**10**	**828.94**	10	828.94	0	0
C108	10	828.94	**10**	**828.94**	10	828.94	0	0
C109	10	828.94	**10**	**828.94**	10	828.94	0	0
C201	3	591.56	**3**	**591.56**	3	591.56	0	0
C202	3	591.56	**3**	**591.56**	3	591.56	0	0
C203	3	591.17	**3**	**591.17**	3	591.17	0	0
C204	3	590.60	**3**	**590.60**	3	590.60	0	0
C205	3	588.88	**3**	**588.88**	3	588.88	0	0
C206	3	588.49	**3**	**588.49**	3	588.49	0	0
C207	3	588.29	**3**	**588.29**	3	588.29	0	0
C208	3	588.32	**3**	**588.32**	3	588.32	0	0
RC101	14	1696.95	15	1635.11	15	1644.17	-	6.98
RC102	12	1554.75	13	1503.42	13	1516.54	-	11.37
RC103	11	1261.67	11	1277.99	11	1278.11	1.28%	0.20
RC104	10	1135.48	**10**	**1135.48**	10	1140.4	0	6.96
RC105	13	1629.44	14	1542.55	14	1542.91	-	6.96
RC106	11	1424.73	12	1388.70	12	1396.14	-	10.06
RC107	11	1230.48	**11**	**1230.48**	11	1235.71	0	5.78
RC108	10	1139.82	11	1157.12	11	1163.34	-	6.72
RC201	4	1406.94	**4**	**1406.91**	4	1410.48	0	6.16
RC202	3	1365.65	4	1169.67	4	1176.73	-	9.98
RC203	3	1049.62	3	1082.57	3	1107.58	3.04%	21.34
RC204	3	798.46	3	828.61	3	834.68	3.64%	6.93
RC205	4	1297.65	**4**	**1297.19**	4	1314.81	0	14.06
RC206	3	1146.32	**3**	**1146.32**	3	1156.29	0	8.64
RC207	3	1061.14	3	1095.67	3	1120.65	3.15%	19.85
RC208	3	828.14	3	843.28	3	872.78	1.80%	17.33

In Table 3, we have selected two recently published state-of-art algorithms as competitors for N-CLPS. 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 BSO and MOLNS 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. It can be seen that N-CLPSO has the strongest synthesis capability in the prioritization of vehicles problem.

Table 3: Comparison with recently published representative algorithms

	BSO		MOLNS		N-CLPSO	
Dataset	NV	TD	NV	TD	NV	TD
R101	19	1671.16	19	1654.93	**19**	**1648.08**
R102	17	1504.60	18	1475.33	**17**	**1486.12**
R103	14	1245.86	14	1240.44	**13**	**1299.99**
R104	11	1010.73	10	1010.72	**10**	**996.27**
R105	15	1366.05	15	1389.85	**14**	**1377.11**
R106	13	1288.84	13	1269.14	**12**	**1262.8**
R107	11	1101.56	11	1102.72	**11**	**1081.17**
R108	**10**	**974.17**	10	991.57	10	985.76
R109	12	1165.71	12	1177.76	**11**	**1210.73**
R110	**11**	**1090.92**	12	1129.60	11	1101.49
R111	11	1148.14	12	1108.70	**11**	**1064.67**
R112	10	1004.53	**10**	**964.15**	10	974.95
R201	4	1336.05	4	1305.25	**4**	**1252.37**
R202	4	1128.05	4	1093.67	**3**	**1225.02**
R203	3	1020.10	4	915.43	**3**	**962.25**
R204	3	834.92	3	775.99	**3**	**766.13**
R205	3	1105.38	3	1075.10	**3**	**1027.79**
R206	3	949.11	3	979.21	**3**	**939.46**
R207	4	812.35	**3**	**851.89**	3	872.4
R208	2	940.30	2	754.99	**2**	**740.36**
R209	3	1046.73	4	898.23	**3**	**943.72**
R210	3	1069.26	4	941.58	**3**	**965.88**
R211	3	836.36	3	838.14	**3**	**828.90**
C101	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C102	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C103	**10**	**828.06**	10	828.94	10	839.35
C104	**10**	**828.78**	10	828.94	10	833.67
C105	**10**	**824.94**	**10**	**828.94**	**10**	**828.94**
C106	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C107	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C108	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C109	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
C201	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**
C202	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**
C203	**3**	**591.17**	3	591.56	**3**	**591.17**
C204	**3**	**590.60**	**3**	**590.60**	**3**	**590.60**
C205	**3**	**588.88**	**3**	**588.88**	**3**	**588.88**
C206	**3**	**588.49**	**3**	**588.49**	**3**	**588.49**
C207	**3**	**588.29**	**3**	**588.29**	**3**	**588.29**

Table 3: (Continued.)Comparison with recently published representative algorithms

	BSO		MOLNS		N-CLPSO	
Dataset	NV	TD	NV	TD	NV	TD
C208	**3**	**588.32**	**3**	**588.32**	**3**	**588.32**
RC101	16	1643.78	15	1662.56	**15**	**1635.11**
RC102	14	1464.63	14	1486.35	**13**	**1503.42**
RC103	**11**	**1275.64**	12	1291.95	11	1277.99
RC104	10	1156.92	10	1162.53	**10**	**1135.48**
RC105	14	1609.68	15	1604.53	**14**	**1542.55**
RC106	13	1378.45	13	1400.09	**12**	**1388.70**
RC107	11	1318.69	12	1259.55	**11**	**1230.48**
RC108	**11**	**1134.85**	11	1205.13	11	1157.12
RC201	4	1514.41	4	1497.89	**4**	**1406.91**
RC202	4	1326.71	4	1199.53	**4**	**1169.67**
RC203	3	1166.91	4	985.54	**3**	**1082.57**
RC204	3	929.94	**3**	**805.46**	3	828.61
RC205	4	1360.91	5	1340.38	**4**	**1297.19**
RC206	3	1237.21	3	1316.42	**3**	**1146.32**
RC207	4	1039.59	4	1031.62	**3**	**1095.67**
RC208	3	910.59	3	859.13	**3**	**843.28**

Table 4 Comparison of average levels.

	R1	R2	C1	C2	RC1	RC2
Best-known	11.92_1210.34	2.73_951.03	10.00_828.38	3.00_589.86	11.50_1384.17	3.25_1119.24
LNS	12.25_1218.28	3.27_964.11	10.00_833.10	3.00_590.31	12.13_1369.57	3.75_1131.18
CPSO	11.92_1215.78	**2.73_952.98**	**10.00_828.38**	**3.00_589.86**	11.50_1414.24	3.25_1136
AC-CH	12.50_1234.88	2.82_1057.42	10.00_829.59	3.00_593.88	11.88_1441.89	3.38_1146.5
S-PSO	12.58_1232.28	3.00_1016.66	10.00_835.92	3.00_593.42	12.13_1385.47	3.38_1169.07
BSO	12.83_1214.35	3.18_1007.15	**10.00_828.38**	**3.00_589.86**	12.50_1372.83	3.50_1185.78
Tabu-ABC	13.50_1187.90	4.73_862.90	**10.00_828.38**	3.00_590.40	13.25_1361.08	5.50_1017.47
MOLNS	13.00_1209.52	3.36_948.13	10.00_828.94	3.00_589.91	12.75_1384.09	3.75_1129.50
N-CLPSO	**12.42_1207.47**	3.00_956.75	10.00_830.62	**3.00_589.86**	**12.13_1358.86**	**3.38_1108.78**

In Table 4, 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 "12.25_1218.28" in the second row of the table indicates that the average number of vehicles and the average distance cost obtained by LNS on independently runs are 12.25 and 1218.28, respectively. The best results for each category are highlighted in bold.

It can be seen from Table 4 that for the C2 class problems, N-CLPSO obtains the same results as the most well-known ones and obtains smaller travel

distances at the expense of certain vehicles in the R1, RC1, and RC2 problems. However, in R2 and C1 class problems, the performance of N-CLPSO is slightly worse than the CPSO and MOLNS. Similar to the previous comparison, it proved that although the algorithm is better at solving the category "1" problems, it also performs very well in the category "2" problems.

6 Conclusions and future research

In this paper, we propose a PSO-based algorithm, names as N-CLPSO, to solve VRPTW. In the study, two objectives of VRPTW are considered. The primary target is the number of vehicles, while the secondary target is the total distance travelled all the vehicles.

In N-CLPSO, we use CLPSO as the main framework and redefine the speed and position update operators of the CLPSO intending to make it more suitable for VRPTW. Moreover, we propose an elite fragment diversity retention strategy and removal-reinsert-based neighborhood search strategy aiming to address the premature convergence and speed up the convergence of PSO. Meanwhile, we apply a novel sample selection strategy and vehicle insertion strategy to improve the convergence speed of N-CLPSO. The experimental results show that the new proposed strategies enable N-CLPSO to outperform other selected state-of-the-art algorithms for VRPTW at present. Moreover, properties and characteristics of the new introduced strategies are also testified by extensive experiments.

Note that, although N-CLPSO yields very promising performance on different VRPTW problems, we regard that there are still some works need to be further investigated. For example, as well as information of customer and elite segment, other potential useful information about VRPTW can be extracted and utilized to guide the evolution of the population, such as information on customer clusters. Moreover, the location update of particles in N-CLPSO is determined by the three new introduced sets.Thus, how to extract more helpful information from a VRPTW and design more efficient evolution operators for PSO is our next work. Moreover, we also hope to apply the algorithm to other combinatorial optimization problems, especially various variants of the VRP.

Compliance with ethical standards

Acknowledgments The study was funded by the National Natural Science Foundation of China (Grant No.: 61663009, 62106092), the Natural Science Foundation of Fujian Province (Grant Nos.: 2021J011008, 2021J011007, 2022J01916), and the Natural Science Foundation of Education Department of Jiangxi Province (Grant No.: GJJ200629).

Conflict of interest The authors claim that none of the material in the paper has been published or is under consideration for publication elsewhere. And all authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ajibade SSM, Ogunbolu MO, Chweya R, et al (2022) Improvement of population diversity of meta-heuristics algorithm using chaotic map. In: Lecture. Notes. Data Eng. Commun. Tech., Springer, pp 95–104, https://doi.org/10.1007/978-3-030-98741-1_9

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

Chen W, Zhang J, Chung HS, et al (2010) A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans Evol Comput 14:278–300. https://doi.org/10.1109/TEVC.2009.2030331

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

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

Chourasia S, Sharma H, Singh M, et al (2019) Global and local neighborhood based particle swarm optimization. In: Adv. Intell. Sys. Comput., pp 449 – 460, http://dx.doi.org/10.1007/978-981-13-0761-4_44

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

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Manage Sci 6:80–91. http://doi.org/10.1287/mnsc.6.1.80

Gocken T, Yaktubay M, Kilic F (2017) Improvement of a genetic algorithm approach for the solution of vehicle routing problem with time windows. In: IDAP - Int. Artif. Intell. Data Process. Symp., pp 1–8, http://dx.doi.org/10.1109/IDAP.2017.8090185

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

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

Ho SC, Haugland D (2004) A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Comput Oper Res 31:1947–1964. https://doi.org/10.1016/S0305-05480300155-2

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

Hu W, Liang H, Peng C, et al (2013) A hybrid chaos-particle swarm optimization algorithm for the vehicle routing problem with time window. Entropy 15:1247–1270. https://doi.org/10.3390/e15041247

Jiang H, Lu M, Tian Y, et al (2022) An evolutionary algorithm for solving capacitated vehicle routing problems by using local information. Appl Soft Comput 117:108,431. https://doi.org/10.1016/j.asoc.2022.108431

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

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

Konstantakopoulos GD, Gayialis SP, Kechagias EP, et al (2020) A multiobjective large neighborhood search metaheuristic for the vehicle routing problem with time windows. Algorithms 13:243. https://doi.org/10.3390/a13100243

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

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

Lai DS, Demirag OC, Leung JM (2016) A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transport Res E-log 86:32–52. https://doi.org/10.1016/j.tre.2015.12.001

Liang JJ, Qin AK, Suganthan PN, et al (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions.

IEEE Trans Evol Comput 10:281–295. https://doi.org/10.1109/TEVC.2005.857610

Lin C, Chen C, Lin C (2009) A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. IEEE Trans Syst Man Cybern Part C 39:55–68. https://doi.org/10.1109/TSMCC.2008.2002333

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

Liu Z, Qin Z, Zhu P, et al (2020) An adaptive switchover hybrid particle swarm optimization algorithm with local search strategy for constrained optimization problems. Eng Appl Artif Intell 95:103,771. https://doi.org/10.1016/j.engappai.2020.103771

Marinakis Y, Marinaki M, Migdalas A (2019) A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows. Inf Sci 481:311–329. https://doi.org/10.1016/j.ins.2018.12.086

Moon I, Lee J, Seong J (2012) Vehicle routing problem with time windows considering overtime and outsourcing vehicles. Expert Syst Appl 39:13,202–13,213. https://doi.org/10.1016/j.eswa.2012.05.081

Nalepa J, Blocho M (2015) Co-operation in the parallel memetic algorithm. Int J Parallel Program 43:812–839. https://doi.org/10.1007/s10766-014-0343-4

Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell-Us 1:33–57. https://doi.org/10.1007/s11721-007-0002-0

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

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

Shen Y, Liu M, Yang J, et al (2020) A hybrid swarm intelligence algorithm for vehicle routing problem with time windows. IEEE Access 8:93,882–93,893. https://doi.org/10.1109/ACCESS.2020.2984660

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

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

Veeramachaneni K, Osadciw LA, Varshney PK (2005) An adaptive multimodal biometric management algorithm. IEEE Trans Syst Man Cybern Part C 35:344–356. https://doi.org/10.1109/TSMCC.2005.848191

Wang Y, Wang L, Peng Z, et al (2019) A multi ant system based hybrid heuristic algorithm for vehicle routing problem with service time customization. Swarm Evol Comput 50:100,563. https://doi.org/10.1016/j.swevo.2019.100563

Wei B, Xia X, Yu F, et al (2020) Multiple adaptive strategies based particle swarm optimization algorithm. Swarm Evol Comput 57:100,731. https://doi.org/10.1016/j.swevo.2020.100731

Xia X, Gui L, He G, et al (2020a) An expanded particle swarm optimization based on multi-exemplar and forgetting ability. Inf Sci 508:105–120. https://doi.org/10.1016/j.ins.2019.08.065

Xia X, Gui L, Yu F, et al (2020b) Triple archives particle swarm optimization. IEEE Trans Cybern 50:4862–4875. https://doi.org/10.1109/TCYB.2019.2943928

Xia X, Tang Y, Wei B, et al (2020c) Dynamic multi-swarm global particle swarm optimization. Computing 102:1587–1626. https://doi.org/10.1007/s00607-019-00782-9

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

Yu B, Yang Z, Yao B (2011a) A hybrid algorithm for vehicle routing problem with time windows. Expert Syst Appl 38:435–441

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

Zhang D, Cai S, Ye F, et al (2017) A hybrid algorithm for a vehicle routing problem with realistic constraints. Inf Sci 394:167–182. https://doi.org/10.1016/j.ins.2017.02.028

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

Zhang J, Yang F, Weng X (2018) An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows. IEEE Access 6:63,468–63,485. https://doi.org/10.1109/ACCESS.2018.2877767

Zhang W, Yang D, Zhang G, et al (2020) Hybrid multiobjective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search for VRPTW. Expert Syst Appl 145:113,151. https://doi.org/10.1016/j.eswa.2019.113151

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

Statements and Declarations

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Qichao Wu. The first draft of the manuscript was written by Qichao Wu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability

The datasets generated during and analysed during the current study are not publicly available due to follow-up studies but are available from the corresponding author on reasonable request.