Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(4): 2511–2538.
DOI: 10.3934/era.2026116
Received: 10 December 2025
Revised: 02 March 2026
Accepted: 03 March 2026
Published: 23 March 2026

Research article

Synergistic design for VRPTW: A competitive swarm optimizer guided by path diversity index and adaptive neighborhood search

Fei Liang[1,2], Fei Yu[1,2,3,4,*] Hongrun Wu[1,2,*] Songbin Lan[1,2], Yingpin Chen[2,3] and Xuewen Xia[1,2,3]

[1] School of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China
[2] Key Lab of Intelligent Optimization and Information Processing, Minnan Normal University, Zhangzhou 363000, China
[3] Key Laboratory of Light Field Manipulation and System Integration Applications in Fujian Province, Minnan Normal University, Zhangzhou 363000, China
[4] Center for China-ASEAN Regional Collaborative Development, Minnan Normal University, Zhangzhou 363000, China

* **Correspondence:** Email: yufei@whu.edu.cn, wuhongrun@whu.edu.cn.

Abstract: The vehicle routing problem with time windows (VRPTW) is a classical NP-hard combinatorial optimization problem, where NP denotes nondeterministic polynomial time, and it plays a critical role in modern logistics and transportation systems. Although competitive swarm optimization (CSO) algorithms have demonstrated strong performance in continuous optimization, their effective application to discrete combinatorial problems such as VRPTW remains challenging. In this paper, a hybrid competitive swarm optimization and tabu search algorithm (CSO-TS) is proposed to solve the VRPTW. To enhance the search capability of the CSO framework, a tabu search mechanism with an adaptive neighborhood operation strategy is integrated. Moreover, to achieve a better balance between exploration and exploitation, a path diversity index is introduced to quantitatively evaluate solution diversity based on four distinct indices. The proposed CSO-TS algorithm is tested on 56 Solomon benchmark instances and obtains 23 optimal solutions. Extensive computational experiments and comparative analyses demonstrate that CSO-TS outperforms or is competitive with nine state-of-the-art algorithms in terms of solution quality.

Keywords: VRPTW; competitive swarm optimizer; tabu search algorithm; large scale optimization

1. Introduction

The rapid expansion of e-commerce has led to a substantial increase in order volumes and parcel processing demands within the logistics industry [1]. As a consequence, achieving timely deliveries

while minimizing operational costs has become a critical research topic in modern logistics. Through an in-depth analysis of these challenges, it is evident that the vehicle routing problem (VRP) provides a feasible and promising framework for addressing such issues. An effective solution to the VRP typically considers multiple objectives, including timely customer service, and reduced distribution costs. Consequently, the development of efficient and reliable algorithms for solving VRP has emerged as a central research focus in logistics and distribution systems.

The VRP was first introduced by Dantzig and Ramser in 1954 as a classical combinatorial optimization problem, originating from their study on optimal routing strategies for gasoline delivery trucks [2]. Owing to its strong relevance to real-world logistics applications, the VRP has attracted extensive research attention over the past decades. In recent years, numerous VRP variants have been proposed to better model practical transportation scenarios, such as the stochastic capacitated multi-depot VRP with pickup and delivery (SCMVRPPD) [3] and the VRP with underground transportation (VRP-UT) [4]. Among these variants, the vehicle routing problem with time windows (VRPTW) [5] has gained particular prominence, as it incorporates customer time window constraints and vehicle capacity limitations, thereby significantly enhancing the realism of the problem formulation.

When addressing highly complex optimization problems, two primary strategies are commonly adopted: (i) simplifying the problem model to obtain exact solutions, and (ii) employing a more accurate but complex model to derive approximate solutions. Several exact algorithms have been developed to solve the VRPTW [6–8]. However, due to the NP-hard nature of the VRPTW, exact methods become computationally prohibitive when applied to large-scale instances. Consequently, increasing research efforts have been devoted to heuristic and meta-heuristic approaches for solving the VRPTW. Representative methods include tabu search (TS) [9], ant colony optimization (ACO) [10], particle swarm optimization (PSO) [11] , and genetic algorithms (GA) [12]. Nevertheless, as the problem scale and constraint complexity increase, many existing algorithms suffer from slow convergence and suboptimal solution quality, highlighting the need for more efficient and robust solution methods.

As an advanced variant of PSO, the competitive swarm optimizer (CSO) is characterized by its distinctive pairwise competition mechanism. CSO has been successfully applied in various domains, including feature selection, multiobjective optimization, and machine learning parameter optimization [13]. Notably, CSO exhibits strong global search capability and scalability when handling large-scale optimization problems, while maintaining competitive convergence speed and solution quality. Unlike traditional PSO, CSO updates only half of the population in each generation. In this process, superior individuals selected through pairwise competition not only guide the evolution of inferior individuals but are also directly retained for the next generation. Meanwhile, the mean position of the swarm is incorporated into the learning mechanism to preserve population diversity.

Despite these advantages, the CSO algorithm also exhibits certain limitations. First, the learning mechanism in the basic CSO framework is relatively simplistic, which limits its adaptability when applied to complex combinatorial optimization problems. Second, although CSO demonstrates strong exploration capability, this often comes at the expense of exploitation performance, leading to insufficient solution precision in the later stages of the search. To address these issues, numerous studies have proposed modified CSO variants aimed at enhancing population diversity and search efficiency [14–16]. In particular, several researchers have integrated efficient local search strategies into CSO frameworks to compensate for their limited exploitation ability.

Meta-heuristic algorithms have achieved notable success in solving the VRPTW; however, their

performance often degrades when dealing with large-scale instances. Owing to its pairwise competitive learning mechanism, CSO offers the advantages of low computational cost and high population diversity, making it particularly suitable for large-scale optimization problems. To the best of our knowledge, this study represents the first attempt to apply the CSO algorithm to the VRPTW.

Based on the above analysis, this paper proposes a hybrid competitive swarm optimization–tabu search (CSO-TS) algorithm for solving the VRPTW. The main motivations and methodological contributions of the proposed CSO-TS algorithm are summarized as follows:

1) An enhanced CSO-TS framework is developed by introducing novel velocity and position update mechanisms, which significantly improve optimization performance and establish a new computational paradigm for particle evolution.

2) A path difference index is proposed to resolve the dimensional inconsistency between velocity and position representations when applying CSO to discrete VRPTW solutions.

3) An improved tabu search strategy is designed to address the excessive infeasible solutions and low search efficiency of conventional TS in VRPTW, incorporating a hierarchical perturbation mechanism and dynamic tabu list length management.

The remainder of this paper is organized as follows. Section 2 reviews the VRPTW model, the CSO algorithm, and related literature. Section 3 presents the proposed CSO-TS algorithm and its associated strategies in detail. Section 4 reports and analyzes the computational results. Finally, Section 5 concludes the paper and discusses potential directions for future research.

2. Background

2.1. Mathematical definition of VRPTW

VRPTW aims to determine the lowest-cost routes for a uniform-capacity vehicle fleet to serve multiple customers within specified time windows. A crucial constraint is that the overall service demand allocated to each vehicle must not go beyond its maximum load capacity, and every customer must be served precisely once by one single vehicle within their prearranged time window. In particular, if a vehicle arrives at a customer's location prior to the commencement of their designated time window, it is required to wait until the window opens. Conversely, if a vehicle arrives after the closure of the time window, the customer cannot be served during that trip.

VRPTW can be defined as follows: A fleet composed of U vehicles is assigned to fulfill the service demands of N customers, and each vehicle is equipped with a fixed load capacity C. The depot, marked as v_0, serves as the unified starting point and terminal point for every planned vehicle route. Each customer is corresponding to a vertex denoted by v_i, where the index i ranges from 1 to N ($i \in 1, 2, ..., N$). Specifically, customer v_i is situated at the geographic coordinate x_i, y_i, with a specific demand quantity C_i and a predefined delivery time window $[e_i, f_i]$. In this time window, e_i represents the earliest allowable moment to initiate service for the customer, while f_i denotes the final time for starting the service–any service initiation beyond f_i is deemed invalid. If a vehicle arrives at the location of customer v_i ahead of the time e_i, it has to remain idle at that spot until the time window opens before providing the scheduled service. In contrast, if the vehicle's arrival time at v_i exceeds f_i, it will lose the qualification to serve this customer. The duration required to complete the service for customer v_i is recorded as s_i. As for the depot, it is located at (x_0, y_0), with a demand of $C_0 = 0$ due to it being a dispatch center rather than a service-receiving party. Its time window is set as $[0, f_0]$, where f_0 is constrained to be no smaller

than the maximum value among all e_i of the customers, ensuring the vehicle fleet has sufficient time to complete all delivery tasks. For the purpose of simplifying the problem model, the time cost generated when a vehicle travels from customer i to customer j (with $i \neq j$ and $i, j \in 1, 2, ..., N$) is quantified by the Euclidean distance $ED_{i,j}$.

Typically, VRPTW encompasses two distinct objectives, as specified in Eqs (2.1) and (2.2), respectively. The core goal is minimizing the vehicle count (VC), and the secondary goal is to minimize the total distance of all routes (DR) while keeping the number of routes unchanged. Below is the mathematical formulation corresponding to VRPTW.

$$\min VC = E \tag{2.1}$$

and

$$\min DR = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{u=1}^{U} ED_{i,j} \times z_{i,j}^{u} , \tag{2.2}$$

subject to

$$\sum_{i=0}^{N} z_{i,j}^{u} = g_j^u \quad \forall\, u = 1, \ldots, U,\ \forall\, j = 1, \ldots, N \tag{2.3}$$

$$\sum_{i=0}^{N} z_{i,j}^{u} = g_i^u \quad \forall\, u = 1, \ldots, U,\ \forall\, i = 1, \ldots, N \tag{2.4}$$

$$\sum_{u=1}^{U} g_i^u = 1 \quad \forall\, i = 1, \ldots, N \tag{2.5}$$

$$\sum_{i=0}^{N} g_i^u \times c_i \leq C \quad \forall\, u = 1, \ldots, U \tag{2.6}$$

$$\sum_{u=1}^{U} y_0^u = U \tag{2.7}$$

$$t_i + w_j + s_i + ED_{i,j} = t_j \quad \forall\, i, j = 1, \ldots, N,\ i \neq j \tag{2.8}$$

$$e_j \leq t_j \leq f_j \quad \forall\, j = 1, \ldots, N \tag{2.9}$$

$$w_j = \max\{e_j - (t_i + s_i + ED_{i,j}), 0\} \quad \forall\, i = 1, \ldots, N \tag{2.10}$$

where

$$z_{i,j}^{u} = \begin{cases} 1, & \text{if vehicle } u \text{ travels directly from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases} \tag{2.11}$$

$$g_i^u = \begin{cases} 1, & \text{if customer } i \text{ is served by vehicle } u \\ 0, & \text{otherwise,} \end{cases} \tag{2.12}$$

Constraints Eqs (2.3)–(2.5) ensure the uniqueness and completeness of customer service: Each customer is served by one vehicle, and no customer is assigned to more than one vehicle. Constraint Eq (2.6) imposes a capacity limit on each vehicle, requiring that its total load does not exceed C. Constraint Eq (2.7) mandates that all vehicle routes start from the depot. Equations (2.8)–(2.10) formalize the time window constraints, where t_i is the vehicle's arrival time at node i, w_j is the waiting time after the vehicle arrives at customer j, s_i denotes the service time at node i, and $t_{i,j}$ is the inter-node time cost between i and j.

2.2. Meta-heuristic algorithm for VRPTW

For several decades now, researchers have developed numerous algorithms targeting the VRPTW problem. Traditional deterministic algorithms can perform effectively on some VRPTW instances, yet they usually cannot produce promising results for large-scale versions of the problem within a reasonable time. Thus, the focus of recent research has been predominantly on adopting various evolutionary algorithms (EAs) for the optimization of this problem.

ACO, inspired by the foraging behavior of real world ants, is a well known probabilistic approach for solving VRPTW [17]. In view of customer service duration, Wang et al. constructed a multi-ant system embedded with local search [18]. This system merges the multi ant system algorithm and four local search operators to enhance the quality of the solutions obtained. Gupta et al. presented a modified ACO algorithm for addressing the VRPTW [19]. It utilizes innovative pheromone resetting and updating functions to strengthen the algorithm's global search performance, and optimizes the planned path through the implementation of the 2-opt method. Zhang et al. developed a solution approach based on ACO combined with three variational operators to tackle a multi-objective VRP with flexible time windows [20].

GA, a heuristic algorithm that mimics the genetic evolution of living organisms via genes and chromosomes, has been widely used in addressing VRPTW thanks to its strong search performance and good extensibility. With VRPTW as the research object, Zhang et al. developed a hybrid multi-objective evolutionary algorithm [21]. It incorporates global search based on a fast sampling strategy and local search rooted in route sequence differences, which is designated as HMOEAGL. Khoo et al. proposed a two phase distributed ruin and recreate genetic algorithm (RRGA) tailored for VRPTW [12]. This approach merges the ruin and recreate strategy with GA, leveraging the complementary strengths of the ruin and recreate strategy's search diversification and GA's search intensification to produce solutions of remarkably high quality.

PSO is a representative swarm intelligence algorithm that mimics a bird flock's foraging. Initially, for continuous problems, it has been adapted by scholars to tackle combinatorial tasks like VRPTW. Reong et al. reviewed 20 years of PSO algorithms, focusing on their use in VRP and variants such as VRPTW [22]. Saksuriya et al. proposed a hybrid heuristic framework combining novel local search–ruin and recreate and PSO for VRPTW [23]. Salehi et al. proposed tailored multiobjective PSO and variable neighborhood search algorithms for VRP with flexible time windows and recharging constraints [24]. Ding et al. formulated an electric VRP incorporating load, distance, and speed constraints under

urban/suburban driving conditions [25], and developed an adaptive PSO approach.

Local neighborhood search strategies have become a decisive factor in helping solutions get rid of local optima during the research on VRP variants. Therefore, designing efficient neighborhood search mechanisms has received great attention from the research community in recent years. Liu et al., for instance, developed an efficient hybrid algorithm combining large neighborhood search (LNS) and GA [26]. This algorithm uses a constraint relaxation scheme to expand the search space by exploring neighborhoods related to existing infeasible solutions. When the search stagnates in a local optimum, it starts the GA to explore solution spaces that have not been explored before. Yu et al. integrated a neighborhood search operator into the ACO framework and preserved the diversity of the population through tabu search mechanisms [10]. In their study, an improved adaptive tabu search algorithm was used to solve VRPTW. This algorithm comprehensively considers multiple cost dimensions such as fixed vehicle costs and driver costs, which ultimately led to excellent performance.

As observed in the literature, EAs remain prevalent for addressing VRPTW, owing to their promising performance compared to deterministic approaches particularly in terms of solution quality and computational efficiency. In EA-oriented research, a range of insertion heuristics and repair operators have been developed to generate solutions that meet constraint requirements, while diverse local search strategies are adopted to conduct in-depth optimization and further improve the quality of these feasible solutions. Despite their satisfactory performance on conventional VRPTW instances, these methods often lack efficiency when applied to large-scale problems.

To overcome this limitation, this study leverages the pairwise competition mechanism of CSO to enhance algorithmic efficiency. Furthermore, effective operators such as vehicle insertion strategies and local search mechanisms are designed to improve solution quality. Additionally, to strengthen search capability, four path diversity indices are incorporated, which have been successfully applied in various EAs [27–30].

2.3. CSO

The standard CSO algorithm defines the state of each particle p at the t^{th} generation using two core vectors: a position vector $X_p^t = \left[x_{p,1}^t, x_{p,2}^t, \ldots, x_{p,D}^t\right]$ and a velocity vector $V_p^t = \left[v_{p,1}^t, v_{p,2}^t, \ldots, v_{p,D}^t\right]$. Specifically, X_i^t serves as a potential candidate solution to the target optimization problem, while V_i^t comprehensively determines both the search direction and step size of particle i during the t^{th} generation. The parameter D corresponds to the dimension of the problem to be optimized. The swarm $S(t)$ is initialized randomly with q particles and undergoes continuous iterative updates; here, q is the total number of particles in the swarm, and t is the index distinguishing different generations.

In each generation, particles in $S(t)$ are randomly paired into $q/2$ independent couples (on the premise that q is even). For each pair, a fitness competition is conducted: The particle with better performance (the "winner") gains direct access to $S(t+1)$ without modification. The "loser" (particle with inferior fitness) updates its position and velocity by learning from the winner's characteristics, and after completing this learning-driven update, it is also incorporated into $S(t+1)$. A critical rule of this mechanism is that each particle participates in exactly one competition. To elaborate, a swarm of q particles requires $q/2$ competitions, ensuring every particle is involved in precisely one. Therefore, the position and velocity of exactly $q/2$ particles (all losers) are updated in each generation.

The position and velocity of the winner and loser in the k-th competition round of generation t are denoted as $X_{k,w}(t)$, $X_{k,l}(t)$, and $V_{k,w}(t)$, $V_{k,l}(t)$, where $k = 1, 2, \ldots, q/2$. After the k-th competition, the

loser's velocity is updated using the following learning strategy:

$$V_{k,l}(t+1) = R_1 V_{k,l}(t) + R_2(X_{k,w}(t) - X_{k,l}(t)) + \lambda R_3(\bar{X}_k(t) - X_{k,l}(t)), \tag{2.13}$$

Thus, the position of the loser can be updated based on the newly generated velocity:

$$X_{k,l}(t+1) = X_{k,l}(t) + V_{k,l}(t+1), \tag{2.14}$$

where $R_1, R_2, R_3 \in [0,1]^n$ are three randomly generated vectors, $\bar{X}_k(t)$ is the mean position value of the associated particles, and λ is a parameter that controls the impact of $\bar{X}_k(t)$. Specifically, for $\bar{X}_k(t)$, a global version and a local version can be adopted:

1) $\bar{X}_k^l(t)$ refers to the local mean position of the particles within a predefined neighborhood of the target particle;

2) $\bar{X}_k^g(t)$ stands for the global mean position of all particles in $S(t)$.

Studies have confirmed that neighborhood control improves PSO's performance on multimodal functions by maintaining high swarm diversity [31]. Similarly, integrating neighborhood control into $\bar{X}_k(t)$ aims to enhance swarm diversity, potentially boosting CSO's search performance. Unless otherwise specified, the global version $\bar{X}_k^g(t)$ is adopted as the default in the rest of this paper.

3. Method

In this section, Section 3.1 presents the framework of CSO-TS, and Section 3.2 describes the details of it. Section 3.3 details the difference indices of the path. Section 3.4 introduces the vehicle insertion strategy. Section 3.5 shows tabu search strategy. Complexity analysis of CSO-TS are described in Section 3.6.

3.1. Framework of CSO-TS

The proposed CSO-TS algorithm integrates the global search capabilities of CSO with a novel local search strategy. While CSO demonstrates excellent global optimization performance for continuous problems, its encoding scheme and operators require substantial modification to accommodate the discrete nature of VRPTW.

$x_{p,D}^t$ denotes the set $b_{a,c} = \{\langle a, b\rangle, \langle b, c\rangle\}$ of a set of arcs in the D-dimension of X_p^t, which indicates that the left and right neighbors of node b in particle p in the t^{th} generation are a and c, respectively. Meanwhile, there exists a set of probability sets $v_{p,D}^t = [\langle k, m\rangle / h(k, m) \mid \langle k, m\rangle \in A_b]$ in dimension D of the velocity vector V_p^t, where A_b denotes the set of all possible adjacent arcs to node b and $h(k, m) \in [0, 1]$ is the probability corresponding to each arc $\langle k, m\rangle$. The framework of CSO-TS is demonstrated by Figure 1.

CSO-TS builds upon the CSO framework by integrating two additional operators: a vehicle insertion strategy (see Section 3.4) and a neighborhood search strategy (see Section 3.5). After updating the velocity and position according to the CSO-TS rules (see Section 3.2), the vehicle insertion strategy is applied to reduce the vehicle count while maintaining solution feasibility. This process is repeated iteratively until the stopping criteria are satisfied.

As noted in Section 2, VRPTW has two objectives. In this study, we combine a new decision method to deal with the vehicle count VC and the total distance of all routes DR of VRPTW [11]. To better

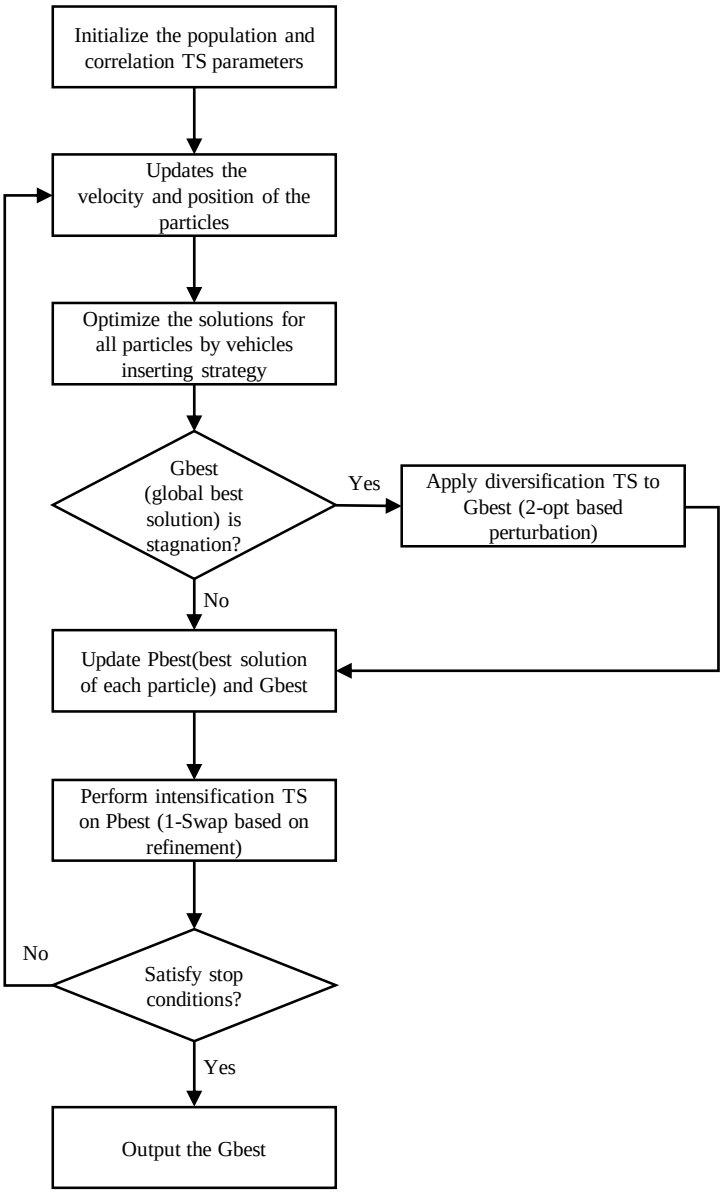

Figure 1. Framework of CSO-TS.

present the priority of the *VC* objective over the *DR* objective, we normalize *DR* in the range of [0,1] weighted with *VC*. The objective function of CSO-TS is defined as Eq (3.1), where *VC* and *DR* denote the vehicle count and the total distance of all routes corresponding to particle X_p^t, respectively.

$$\begin{aligned} \text{fitness}(X_p^t) &= VC(X_p^t) + \text{normalize}\left(DR(X_p^t)\right), \\ \text{normalize}(x) &= \frac{\arctan(x)}{\frac{\pi}{2}}, \end{aligned} \tag{3.1}$$

Considering the characteristics of the VRPTW, which is fundamentally a permutation and combination problem, the conventional CSO typically allows losing particles to learn from the winning ones while the winners remain unchanged. In this study, we propose a modified CSO algorithm in which the winners are updated using the standard PSO framework, while the losers continue to follow the standard CSO update mechanism. The update formula for the winners is given as follows:

$$V_{k,w}(t+1) = WV_{k,w}(t) + C_1R_4(PB_p^t - X_{k,w}(t)) + C_2R_5(GB^t - X_{k,w}(t)), \tag{3.2}$$

$$X_{k,w}(t+1) = X_{k,w}(t) + V_{k,w}(t+1), \tag{3.3}$$

where $R_4, R_5 \in [0,1]^n$ are three randomly generated vectors. W is an inertia weight that controls the impact of the current velocity on the updated velocity. During the population-based search process, each particle adjusts its trajectory based on both its own historical best position $PB_p^t = \left[pb_{p,1}^t, pb_{p,2}^t, \ldots, pb_{p,D}^t\right]$ and the population's historical best position $GB^t = \left[gb_1^t, gb_2^t, \ldots, gb_D^t\right]$. $C1$ and $C2$ are two constants that specify the learning weights assigned to PB_p^t and GB^t, respectively.

3.2. Basic operators in CSO-TS

In CSO-TS, the velocity is updated using Eqs (2.2) and (3.12) based on the aforementioned definitions of position and velocity. The corresponding operators in these equations are defined by Eqs (3.4)–(3.7), respectively. $c \times v_{p,D}^t$ and $v_{p,D}^t + v_{j,D}^t$ denotes the probability $p(k,m)$ variation of the arc. $x_{p,D}^t - x_{q,D}^t$ means the set solving difference set, and $c\left(x_{p,D}^t - x_{q,D}^t\right)$ stands for converting a crisp set into a set with the following probability:

$$\begin{aligned} c \times v_{p,D}^t &= \left\{\langle k,m\rangle / h'(k,m) \;\middle|\; \langle k,m\rangle \in A_b\right\}, \\ h'(k,m) &= \begin{cases} 1, & \text{if } c \times h(k,m) > 1, \\ c \times h(k,m), & \text{otherwise.} \end{cases} \end{aligned} \tag{3.4}$$

$$v_{p,D}^t + v_{q,D}^t = \left\{\langle k,m\rangle \;\middle|\; \max(h_p(k,m), h_q(k,m)) \;\middle|\; \langle k,m\rangle \in A_b\right\}, \tag{3.5}$$

$$\begin{aligned} x_{p,D}^t - x_{q,D}^t = R_D &= \left\{\langle k,m\rangle \;\middle|\; \langle k,m\rangle \in x_{p,D}^t \text{ and } \langle k,m\rangle \notin x_{q,D}^t\right\}, \\ cR_D &= \left\{\langle k,m\rangle / h'(k,m) \;\middle|\; \langle k,m\rangle \in A_b\right\}, \end{aligned} \tag{3.6}$$

$$h'(k,m) = \begin{cases} 1, & \text{if } \langle k,m\rangle \in R_D \text{ and } c > 1, \\ c, & \text{if } \langle k,m\rangle \in R_D \text{ and } 0 < c < 1, \\ 0, & \text{if } \langle k,m\rangle \notin R_D. \end{cases} \tag{3.7}$$

The winner serves as an example; we assume that $w = 0.4$, $C1 = 2.0$, $C1 = 2.0$, $R4 = 0.3$, and $R5 = 0.5$. $x_{p,1}^t = \{\langle 5,1\rangle, \langle 1,2\rangle\}$, $PB_p^t = \{\langle 1,4\rangle, \langle 5,1\rangle\}$, $GB^t = \{\langle 4,1\rangle, \langle 1,2\rangle\}$, and $v_{p,1}^t = \{\langle 1,2\rangle/0.3, \langle 1,4\rangle/0.5, \langle 4,1\rangle/0.6\}$. Then, we have $w \times v_{p,1}^t = \{\langle 1,2\rangle/0.12\}$, $PB_p^t - x_{p,1}^t = \{\langle 1,4\rangle\}$, $GB^t - x_{p,1}^t = \{\langle 4,1\rangle\}$, and $C_1R_4\left(PB_p^t - x_{p,1}^t\right) + C_2R_5\left(PB_p^t - x_{p,1}^t\right) = \{\langle 1,4\rangle/0.6, \langle 4,1\rangle/1.0\}$. Finally, the

new velocity $v^t_{p,1} = w \times v^t_{p,1} + C_1 R_4 \left(PB^t_p - x^t_{p,1}\right) + C_2 R_5 \left(PB^t_p - x^t_{p,1}\right) = \{\langle 1,2\rangle/0.12, \langle 1,4\rangle/0.6, \langle 4,1\rangle/1.0\}$ can be obtained.

The CSO-TS module introduced the standard PSO (SPSO) algorithm, a pioneering method that employs PSO to solve VRPTW. A key innovation of this approach is its novel position update process, which substitutes the original one in CSO and is outlined in Algorithm 1. The position of the particle can be obtained by three sets:

$$S_V = \left\{ m \;\middle|\; \langle k,m\rangle \in V_p \text{ and } \langle k,m\rangle \text{ satisfies } \Omega \right\}, \tag{3.8}$$

$$S_X = \left\{ m \;\middle|\; \langle k,m\rangle \in X_p \text{ and } \langle k,m\rangle \text{ satisfies } \Omega \right\}, \tag{3.9}$$

$$S_Z = \left\{ m \;\middle|\; \langle k,m\rangle \in Z \text{ and } \langle k,m\rangle \text{ satisfies } \Omega \right\}. \tag{3.10}$$

Ω refers to the feasible solution space.

Algorithm 1 Position update in CSO-TS

Input: X_p; V_p; Z;
Output: X_p^{t+1}

$X_p = \emptyset$; $k = 0$;
$S_V = \{m \mid \langle k,m\rangle \in V_p \text{ and } \langle k,m\rangle \text{ satisfies } \Omega\}$;
$S_X = \{m \mid \langle k,m\rangle \in X_p \text{ and } \langle k,m\rangle \text{ satisfies } \Omega\}$;
$S_Z = \{m \mid \langle k,m\rangle \in Z \text{ and } \langle k,m\rangle \text{ satisfies } \Omega\}$;
while All customers are unable to access **do**
    if $S_V \neq \emptyset$ **then**
        select m in S_V, and add $\langle k,m\rangle$ to X_p^{t+1}
        $k = m$;
        update S_V;
    else if $S_X \neq \emptyset$ **then**
        select m in S_X, and add $\langle k,m\rangle$ to X_p^{t+1}
        $k = m$;
        update S_X;
    else if $S_Z \neq \emptyset$ **then**
        select m in S_Z, and add $\langle k,m\rangle$ to X_p^{t+1}
        $k = m$;
        update S_Z;
    else
        $k = 0$;
        update S_V, S_X, S_Z;
    end if
end while

It can be observed that arcs $\langle k,m\rangle$ are derived from sets V_p, X_p, and Z, respectively, while satisfying relevant constraints. To prioritize the selection of arcs with higher probabilities, a random number

$r \in [0, 1]$ is set so that only arcs in V_p^t with probabilities exceeding r are added to V_p. Additionally, X_p and Z denote all possible arcs from the current position V_p^t and the search space, respectively.

As outlined in Algorithm 1, the new position X_p^{t+1} is initialized as an empty set prior to position update. In line with constraints, each vehicle departs from the depot and iteratively selects the next customer to serve. Let k denote the customer currently served by the vehicle and m represent the next customer to be visited. The selection of node m follows a deterministic sequence. Availability in set S_V is first evaluated. If an eligible node exists in S_V, it is selected. Otherwise, the search proceeds to set S_X. In the event that no available nodes are found in either S_V or S_X, the algorithm finally selects from S_Z. Subsequent to the selection of m, the arc $\langle k, m\rangle$ is added to the solution set X_p^{t+1}. This procedure is iterated until termination occurs upon visiting all customers.

In the absence of available nodes within the sets S_V, S_X, and S_Z, the constraints of VRPTW cannot be fulfilled. Consequently, it becomes necessary to establish a new route, specifically by initiating a new subtour. This involves inserting a depot node after the current customer point k, then reselecting the next customer point m to serve with the depot as the starting point. This approach ensures the feasibility of X_p^{t+1}. Finally, the updated X_p^{t+1} replaces the current position X_p^t. Additionally, the heuristic selection method nearest neighbor heuristic (NNH) [11] is adopted to accelerate the convergence of Algorithm 1.

3.3. Path diversity index

In addressing the VRPTW problem, the CSO-TS method employs a distinctive encoding technique whereby vehicle routes are encapsulated within the position X of the particles. Consequently, X must be structured as a two-dimensional array to accurately represent the sequential order of the routes. Furthermore, the velocity vector V of the particles is articulated as a probability during the subsequent path selection process, with the velocity vector being a value confined to the range of 0 to 1. Therefore, when using Eqs (2.13) and (3.2) to update the velocities of the particles, it is found that there is the addition of a two-dimensional array and a value on the right side of the equation, which presents an evident contradiction. To address this, we propose the difference indices of the path to replace the path calculation in the original formula.

The incorporation of path diversity metrics into CSO for VRPTW is driven by significant challenges faced in conventional CSO implementations, primarily arising from the discrete and constrained nature of the problem. Traditional CSO lacks explicit mechanisms to address constraints specific to VRPTW. This results in frequent generation of infeasible solutions, increasing computational overhead for constraint repair and reducing algorithm efficiency. By incorporating path diversity metrics, the proposed approach directly addresses these limitations. Metrics such as reversal distance enable the algorithm to quantify and promote structural changes in routes, encouraging particles to explore distant solution regions without relying solely on random perturbations [29]. By reinterpreting velocity as a probability distribution over discrete operations, diversity metrics help calibrate these probabilities to maintain swarm heterogeneity. In summary, path diversity metrics enhance the capability of CSO to address VRPTW by offering a systematic framework to quantify, preserve, and utilize solution diversity. This approach effectively overcomes the limitations inherent in conventional continuous optimization paradigms within discrete, constrained environments.

Due to space constraints, we provide a concise overview of the four indices employed in this study:

1) The Hamming distance is defined as the count of positions at which corresponding elements in two sequences differ [27]. Therefore, its core role in the VRPTW problem is to measure the local

differences in node positions and guide fine-tuning operations.

2) The edit distance is defined as the minimum number of elementary operations required to transform one sequence into another [28]. It can measure the overall modification cost of the path and balance the priorities between minor adjustments and major reconfigurations.

3) The reversal distance is defined as the minimum number of contiguous segment reversals necessary to align the order of elements between two sequences [29]. The metric in question can assess the overall differences in path structures and facilitate the exploration of novel customer grouping methodologies.

4) The common subpath length is defined as the length of the longest consecutive identical subsequence shared between two paths [30]. It can measure the similarity basis of the path – the longer the length, the more it indicates that the two paths share high-quality substructures.

The four indices are shown in Figure 2.

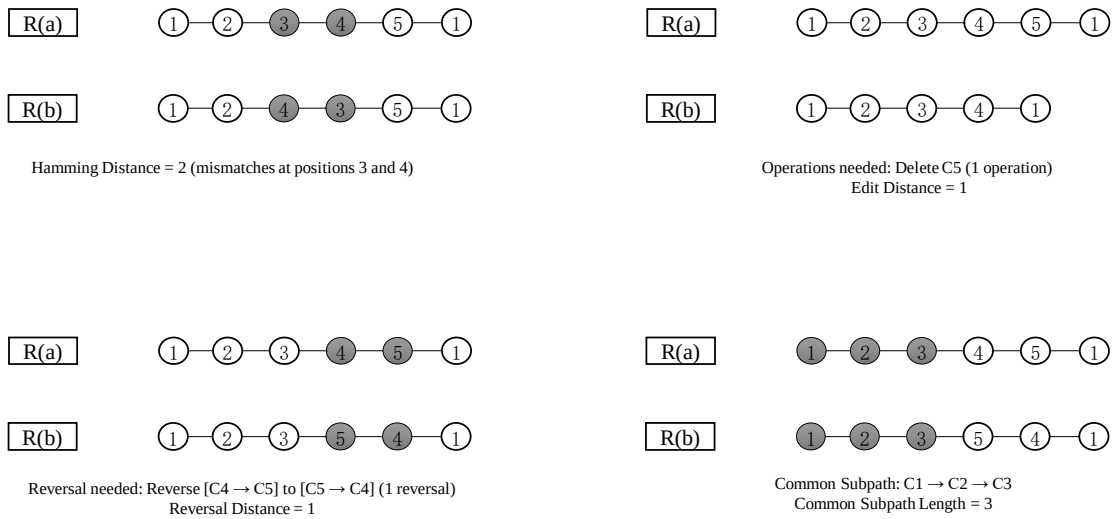

Figure 2. The difference indices of the path.

The speed update formula for the losers after the replacement is as follows:

$$\begin{aligned} V_{k,l}(t+1) = {} & R_1 V_{k,l}(t) + R_2 D_{\text{ind}}(R_{k,w}(t), R_{k,l}(t))(R_{k,w}(t) - R_{k,l}(t)) \\ & + \lambda R_3 D_{\text{ind}}(\bar{R}_{k,w}(t), R_{k,l}(t))(R_{k,w}(t) - R_{k,l}(t)), \end{aligned} \tag{3.11}$$

where $(R_{k,w}(t) - R_{k,l}(t))$ are discrete difference vectors, representing the operations required to transform route $R_{k,l}(t)$ into route $R_{k,w}(t)$. $D_{\text{ind}}(R_{k,w}(t), R_{k,l}(t))$ are based on the differences in the weights of the four indices.

The speed update formula for the winners after the replacement is as follows:

$$\begin{aligned} V_{k,w}(t+1) = {} & W V_{k,w}(t) + C_1 R_4 D_{\text{ind}}(R_{k,pb}(t), R_{k,w}(t))(R_{k,pb}(t) - R_{k,w}(t)) \\ & + C_2 R_5 D_{\text{ind}}(R_{k,gb}(t), R_{k,w}(t))(R_{k,gb}(t) - R_{k,w}(t)). \end{aligned} \tag{3.12}$$

Based on the differences in weights of the four path-specific indices $D_{\text{ind}}(R_{k,w}(t), R_{k,l}(t))$ are as follows:

$$D_{\text{ind}}(X_p^t, R_p^t) = \alpha \frac{H(X_p^t, R_p^t)}{H_{\max}} + \beta \frac{E(X_p^t, R_p^t)}{E_{\max}} + \gamma \frac{R(X_p^t, R_p^t)}{R_{\max}} + \delta \left(1 - \frac{S(X_p^t, R_p^t)}{S_{\max}}\right), \tag{3.13}$$

where $H(X_i^t, R_i^t)$ indicates the Hamming distance, $E(X_i^t, R_i^t)$ expresses the edit distance, $R(X_p^t, R_p^t)$ indicates the Reversal distance, and $(1 - \frac{S(X_p^t, R_p^t)}{S_{\max}})$ expresses the proportion of nonpublic subpaths. $H_{\max}$, $E_{\max}$, $R_{\max}$, $S_{\max}$ represent the maximum values of each index respectively, and they are used to normalize each of the indices. $[\alpha, \beta, \gamma, \delta]$ express the weight coefficient, which can be dynamically adjusted. The specific values will be discussed in the experimental section.

In applying the CSO to VRPTW, we identified the necessity for a comprehensive path diversity index. Relying on a single metric proves insufficient for fully characterizing the differences between solution paths. To address this limitation and ensure a holistic assessment that covers the full spectrum of optimization requirements from local versus global and structural versus cost-related considerations to the balance between exploration and exploitation, we introduce a composite measure based on four path diversity indices.

The four path diversity indices capture complementary aspects of solution diversity at different granularities. Their weighted combination provides a holistic measure of population divergence while allowing flexible adjustment across search stages. In the early phase, indices emphasizing structural and route-level differences contribute more to exploration by promoting broader solution variation, whereas in later stages, indices focusing on finer-grained path similarities support exploitation by refining promising solutions. As a result, the composite diversity index enables a balanced transition between exploration and exploitation that cannot be achieved by a single index or a simple average.

3.4. Vehicle insertion strategy

It is widely recognized that the number of vehicles is of paramount importance to the complexity of VRPTW, with each additional unit in practical operations incurring a substantial cost increment. Consequently, this paper proposes the implementation of a direct insertion scheme subsequent to the particle position update phase to curtail the fleet size. The particulars of this vehicle insertion strategy are elucidated in Algorithm 2.

In the CSO-TS framework, each customer must identify an appropriate insertion position. Clearly, the definition of "appropriate" is pivotal for the guided insertion strategy. In this study, consistent with the core objectives of VRPTW, when customer *vi* is reinserted at position *p* of route *r*, the time window constraints of all customers in route *r* must be given top priority. For the convenience of illustration, position *p* is termed a feasible insertion point, and the distance increase of route *r* after reinserting customer *vi* (where $vi(1 \le i \le N)$) at this position is denoted as $DIS_{i,p}$.When customer *vi* has multiple feasible insertion points, the optimal insertion point p^* for *vi* is selected such that DIS_{i,p^*} attains the minimum value. After identifying the optimal DIS_{i,p^*} for each customer *vi* across the entire population (with n representing the total number of customers in the population), the customer vi^* corresponding to the smallest DIS_{i,p^*} is reinserted at its optimal position p^* in the associated route. If there is no feasible insertion point available for customer *vi*, the route insertion operation for this customer is considered unsuccessful, and an additional vehicle needs to be deployed to serve *vi*. Following this, another customer is randomly selected from the entire population to undergo the same process of feasible insertion point detection and reinsertion. This iterative procedure continues until every customer in the population has been fully evaluated.

Algorithm 2 Vehicle insertion strategy

Input: Current routing r, collection of vehicles to insert CV

Output: Updated routing U_r

```
n = number of vehicles in CV;
i = 1;
while i ≤ n do
    vehicle_to_insert = CV(i);
    temp_route = r − CV(i);
    Insert vehicle_to_insert into temp_route using guided insertion strategy;
    if vehicle_to_insert was successfully inserted then
        r = temp_route;
        n = n − 1;
    else
        r = r;
        i = i − 1;
    end if
end while
U_r = r;
```

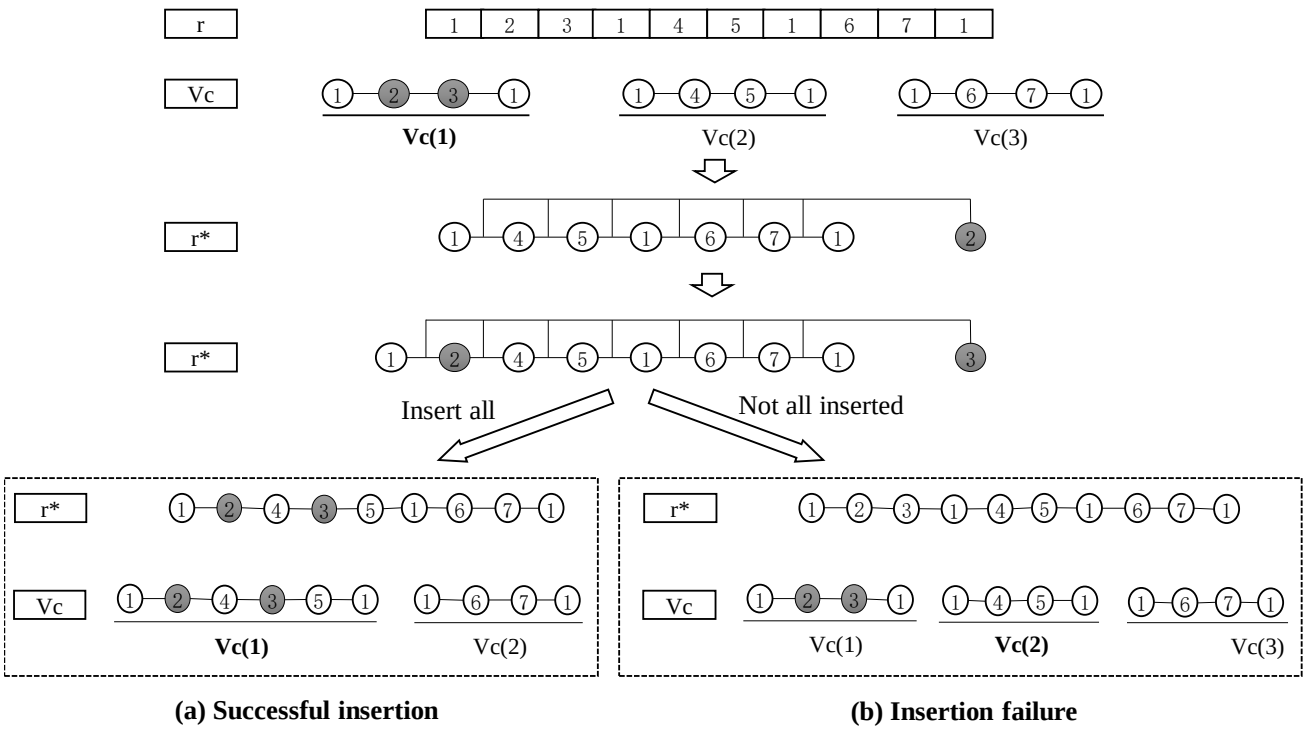

Figure 3. Framework of CSO-TS.

Using Figure 3 as an illustration, here "1" denotes the depot, while "2" to "7" represent 6 customers. The route $r = \{1, 2, 3, 1, 4, 5, 1, 6, 7, 1\}$ consists of 3 subtours, specifically $Vc(1) = \{1, 2, 3, 1\}$, $Vc(2) =$

$\{1, 4, 5, 1\}$, and $Vc(1) = \{1, 6, 7, 1\}$. First, the subtour $Vc(1)$ is removed from route r to generate an intermediate route $r*$. Subsequently, the nodes $\{2, 3\}$ from $Vc(1)$ need to be inserted into $r*$ via a guided insertion strategy. If all nodes are inserted successfully as depicted in Figure 3(a), a new route r with 2 new sub-tours $Vc(1)$ and $Vc(2)$ is obtained, and the process proceeds to remove the new $Vc(1)$ afterward. In contrast, if any node fails to be inserted into $r*$ as shown in Figure 3(b), the route retains the subtour in its preinsertion state, and the next step is to remove $Vc(2)$ instead. Such operations are repeated until every subtour has been processed. The time complexity of this vehicle insertion strategy is directly associated with the number of subtours K. Assuming the time required for a single insertion is m, the time complexity of this strategy is $O(Km)$.

3.5. Tabu search strategy

Given that local search is an essential component for EAs addressing the VRPTW, this paper proposes a novel tabu search algorithm to improve local search efficacy. The tabu search algorithm can be applied to three categories of nodes. First, if the updated pbest outperforms the original one, implementing a local search via tabu search can enhance the "actual quality" and "stability" of the pbest. Second, when the gbest becomes trapped in a local optimum, applying tabu search to the gbest can help it escape and continue improving. Third, it is advisable to perturb particles with poorer performance to promote diversity and convergence within the swarm.

The steps of our tabu search algorithm are as shown in Algorithm 3. In this study, a move refers to a specific modification applied to the current solution, including 1-swap moves and 2-opt moves. The function ApplyMove(s, X) executes the selected move on solution X to generate a new solution s. A tabu move is allowed under the aspiration criterion if it results in a solution better than the global best solution X^*. This ensures that potentially beneficial moves are not unnecessarily prohibited, enhancing convergence and helping the search escape local optima.

Initially, upon obtaining a feasible solution, a neighborhood operation is applied. In contrast to traditional tabu search algorithms, we introduce a novel hierarchical perturbation strategy. During later iterations or when performing local improvements on feasible solutions, the 1-swap strategy is employed. This approach seeks improved solutions by exchanging the positions of two nodes within a route, though its optimization scope remains relatively limited.

For more substantial refinement, the 2-opt local optimization strategy is utilized. It enhances solutions by reversing subpaths within individual routes, progressively improving solution quality through the elimination of edge crossings and subsequent path reconstruction. This method effectively balances exploration and exploitation, thereby achieving robust intermediate optimization performance. The path splitting and merging strategy plays a critical role, particularly during the initial exploration phase or when the algorithm exhibits signs of stagnation. By splitting and merging sub-paths and applying path reversal techniques, this strategy effectively perturbs the current solution, demonstrating a strong ability to escape local optima.

For the prohibition table, we document combinations of operation type and path number, such as swapping customer i with customer j or reversing customers m to n within path r, in order to prevent redundant operations from occurring in the near future. We dynamically manage the length of the taboo list and use dynamic values L:

$$L = \theta N. \tag{3.14}$$

Algorithm 3 Tabu search for VRPTW

Input: Initial feasible solution X_0, maximum iterations T, tabu list size L, neighborhood size l
Output: Best solution found X^*

$X \leftarrow X_0$, $X^* \leftarrow X_0$
Initialize TabuList = $\emptyset$
for $t = 1$ to T **do**
  $S \leftarrow$ GenerateNeighborhood(X, l) ▷ Generate l candidate moves from current solution
  $S_{\text{valid}} \leftarrow \{s \in S \mid \text{ApplyMove}(s, X) \text{ is feasible}\}$
  $S_{\text{tabu}} \leftarrow \{s \in S_{\text{valid}} \mid s \text{ is tabu}\}$
  $S_{\text{asp}} \leftarrow \{s \in S_{\text{tabu}} \mid \text{fitness}(\text{ApplyMove}(s, X)) < \text{fitness}(X^*)\}$
  if $S_{\text{asp}} \neq \emptyset$ **then**
    $s^* \leftarrow \arg\min_{s \in S_{\text{asp}}} \text{fitness}(\text{ApplyMove}(s, X))$ ▷ Aspiration allows tabu moves improving global best
  else if $S_{\text{valid}} \setminus S_{\text{tabu}} \neq \emptyset$ **then**
    $s^* \leftarrow \arg\min_{s \in S_{\text{valid}} \setminus S_{\text{tabu}}} \text{fitness}(\text{ApplyMove}(s, X))$
  else
    $s^* \leftarrow \arg\min_{s \in S_{\text{valid}}} \text{fitness}(\text{ApplyMove}(s, X))$
  end if
  $X \leftarrow \text{ApplyMove}(s^*, X)$ ▷ Apply selected move to current solution
  TabuList $\leftarrow$ UpdateTabuList(TabuList, s^*, L)
  if $\text{fitness}(X) < \text{fitness}(X^*)$ **then**
    $X^* \leftarrow X$ ▷ Update global best solution
  end if
end for
return X^*

N represents the number of clients, and θ fluctuates within the range of $[0.1, 0.3]$. In the early stage of the search, θ is set to a small value to enhance exploration, and in the later stage, it is set to a large value to strengthen exploitation. The number of candidate moves generated per iteration, denoted as l, is dynamically adjusted according to the iteration stage to balance exploration and exploitation. During the early stage of the search, a larger l is employed to explore a wider neighborhood and enhance solution diversity, while in the later stage, a smaller l is used to focus on a fine-grained local search. Specifically, a linear adjustment strategy is adopted:

$$l = l_{\max} - t\,\frac{l_{\max} - l_{\min}}{T}, \tag{3.15}$$

$$l_{\min} = \sigma_{\min} N, \quad l_{\max} = \sigma_{\max} N, \tag{3.16}$$

where t is the current iteration, T is the maximum number of iterations, and $\sigma_{\min}$ and $\sigma_{\max}$ are scaling coefficients. In our implementation, $\sigma_{\min} \in [0.05, 0.10]$ and $\sigma_{\max} \in [0.15, 0.30]$.

Since the tabu search is used as a local search component within the proposed CSO-TS framework, its maximum number of iterations should be proportional to the problem scale. Accordingly, we set

$$T = \eta N, \tag{3.17}$$

where η is a predefined coefficient, $\eta \in [1, 5]$.

From Algorithm 3, we know that the number of neighboring solutions generated by the tabu search algorithm is l and the number of iterations for the tabu search is T. Consider the time required to calculate the target value of each neighborhood solution and the time needed to select the optimal solution from the k neighborhood solutions. Thus, the time complexity of this strategy is $O(TlN)$.

3.6. Complexity analysis of CSO-TS

The time complexity of CSO-TS is discussed in this section. The key parameters are defined as follows:

Table 1. Parameters for time complexity analysis.

Parameter	Description
D	Population size of the particle
N	Number of clients
$Iter$	Maximum iteration count
T	Tabu search iterations
l	Neighborhood solutions per tabu search
K	Number of subtours in the vehicle insertion strategy
m	Insertion time

As outlined in Section 3.1, CSO-TS integrates a vehicle insertion strategy and a neighborhood search strategy into the CSO framework. From the preceding descriptions, the time complexity of its main body is $O(DN)$, the vehicle insertion strategy is $O(TlN)$, and the neighborhood search strategy is $O(Km)$. Since the strategy for escaping from the local optimal solution is only invoked when the algorithm gets stuck, the entire time complexity of CSO-TS is $O(Iter\,(DN + TlN + Km))$. We compared our algorithm with two evolutionary algorithms that have been applied to solve the VRPTW problem in recent years [32, 33]. The computational complexity of CSO-TS did not exhibit a significant increase.

4. Experimental evaluation

4.1. Setup

CSO-TS is evaluated using a well-established benchmark comprising 56 VRPTW instances introduced by Solomon [34], as this benchmark effectively represents a wide range of real-world scheduling scenarios. Based on their spatial and temporal characteristics, these instances are categorized into three main classes: customer locations clustered in a concentrated manner (Class C), randomly dispersed customer locations (Class R), and mixed patterns combining both clustering and randomness (Class RC). Furthermore, each class can be subdivided into two groups based on vehicle capacity and time window constraints: instances with tighter time windows and smaller vehicle capacities (C1, R1, and RC1), and those with more relaxed dispatch schedules and larger vehicle capacities (C2, R2, and RC2).

Furthermore, all simulation environments utilized in this study were carried out using Windows 11, 12th Gen Intel(R) Core(TM) i9-12900K 3.20 GHz, and MATLAB 2022b. In this study, a comprehensive set of experiments is conducted to assess the performance of CSO-TS.

In the experiment, the inertia weight value W in Eq (3.12) is initialized to 0.9 and decreases linearly to 0.4 throughout the optimization process. The acceleration coefficient C_1 and C_2 is set to 2.0. The

population size is set to $N = 20$. The neighborhood search and diversity maintenance strategies are triggered when pbest and gbest remain unchanged for 10 generations and 100 generations, respectively. Each experiment is independently executed five times.

4.2. Parameter tuning

This section primarily examines the impact of the four weight parameters $[\alpha, \beta, \gamma, \delta]$ associated with the path difference indices on the algorithm's performance. Table 2 presents the average optimal solutions achieved by CSO-TS for instances R101 and R201 under varying parameter conditions. In this context, "MVC" and "MDR" denote the mean vehicle count and mean total distance of all routes, respectively.

Table 2. Parameter sensitivity analysis.

Instances	Parameters				MVC	MDR
	α	β	γ	δ		
R101	0.1 ~ 0.4	0.3 ~ 0.2	0.4 ~ 0.1	0.2 ~ 0.3	19	1645.09
	0.10	0.30	0.40	0.20	19	1669.214
	0.15	0.28334	0.35	0.21666	19	1658.737
	0.20	0.26667	0.30	0.23333	19	1655.97
	0.25	0.25	0.25	0.25	19	1656.469
	0.30	0.23333	0.20	0.26667	19	1674.781
	0.25	0.21666	0.15	0.28334	19	1661.301
	0.40	0.20	0.10	0.30	19	1677.33
R201	0.1 ~ 0.4	0.3 ~ 0.2	0.4 ~ 0.1	0.2 ~ 0.3	4	1252.37
	0.10	0.30	0.40	0.20	4	1260.454
	0.15	0.28334	0.35	0.21666	4	1264.642
	0.20	0.26667	0.30	0.23333	4	1259.41
	0.25	0.25	0.25	0.25	4	1271.59
	0.30	0.23333	0.20	0.26667	4	1266.79
	0.25	0.21666	0.15	0.28334	4	1274.682
	0.40	0.20	0.10	0.30	4	1277.15

Note that the parameter α determines the sensitivity to local node position adjustments, and a larger α means the algorithm focuses more on fine-tuning adjacent node sequences. The parameter β regulates the balance between the cost of adjustments and structural changes, where a moderate β helps to avoid excessive insertion or deletion operations. The parameter γ signifies the focus on global structural exploration, with a larger γ encouraging extensive route reorganizations. The parameter δ affects the retention of high-quality shared segments, and a larger δ helps preserve proven efficient subpaths. To ensure the algorithm balances exploration and exploitation, the differences between these coefficients should be reasonable. Thus, the initial values of $[\alpha, \beta, \gamma, \delta]$ are set as follows in the adjustment framework: $\alpha = 0.1 \sim 0.4, \beta = 0.3 \sim 0.2, \gamma = 0.4 \sim 0.1$, and $\delta = 0.2 \sim 0.3$. The value of the weight is adjusted linearly with the number of iterations. In R101 and R201 with narrow and wide time windows, respectively, CSO-TS can demonstrate superior performance. Moreover, we found that when the four weights are adjusted according to the number of iterations, CSO-TS can achieve the best

results. Therefore, the strategy of setting the parameters of CSO-TS to be linearly adjusted with the number of iterations is correct.

4.3. Sensitivity analysis of components in CSO-TS

To elucidate the individual contributions of the proposed components, each was methodically integrated into the original CSO-TS framework. This incremental approach systematically evaluates the strategy's effectiveness in enhancing convergence speed without compromising population diversity.

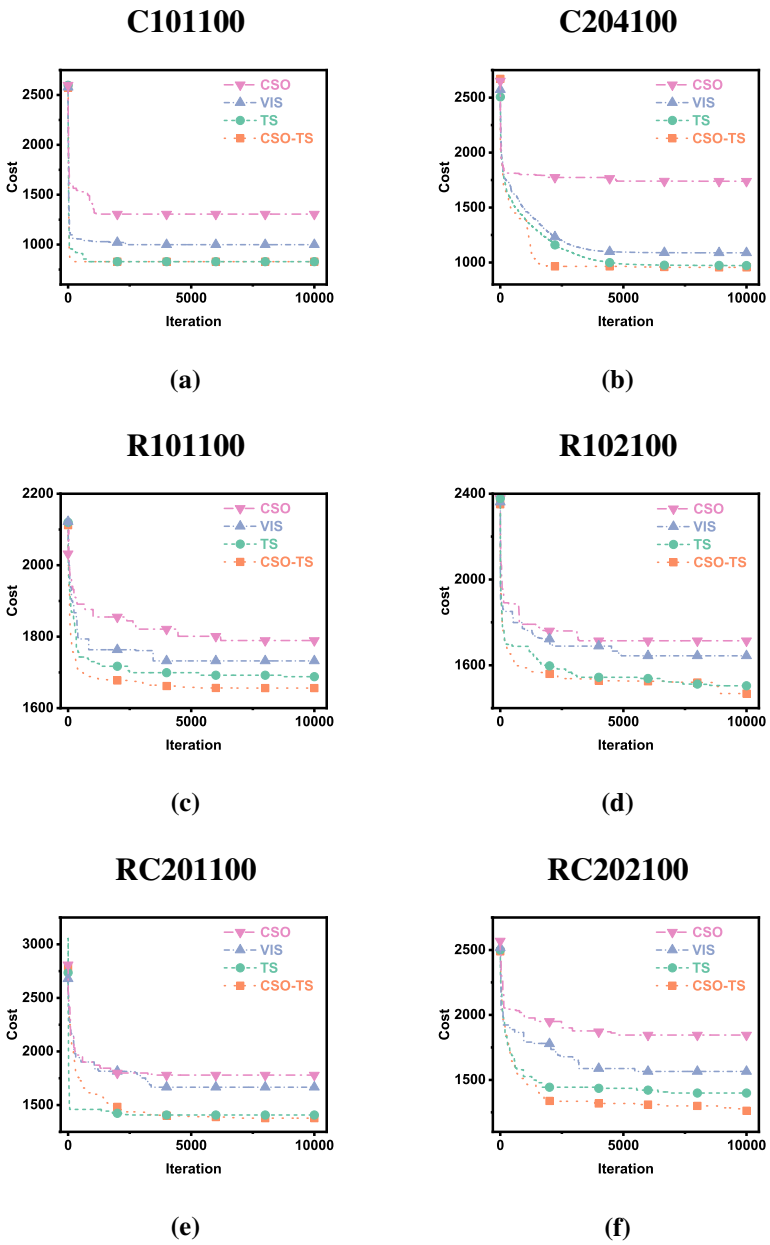

Figure 4. Contribution of vehicle insertion strategy and tabu search strategy.

In CSO-TS, the newly proposed tabu search strategy and vehicle insertion strategy are employed to accelerate the convergence of the algorithm. To evaluate the effectiveness of these strategies, three

comparative algorithms are selected as baselines: CSO, which excludes both proposed strategies; TS, which incorporates only the new Tabu search strategy; and VIS, which includes only the vehicle insertion strategy.

The comparison results presented in Figure 4 indicate that CSO-TS demonstrates a faster convergence speed and higher solution accuracy in the later stages of the experiments. The convergence speed and solution accuracy of TS are slightly inferior to those of CSO-TS, yet they remain superior to those of VIS and CSO. Although VIS performs comparably to CSO in general, it exhibits a significantly faster convergence speed in terms of solution accuracy. Furthermore, as shown in Figure 4, the convergence speed of TS is notably better than that of VIS. This improvement may be attributed to the fact that the tabu search algorithm, as a more effective local search method, possesses strong local search capabilities while being able to effectively avoid local optimal solutions. These capabilities enable particles to approach the optimal solution more closely during iterations, thereby enhancing the convergence performance of TS compared to VIS. Therefore, it can be concluded that the newly proposed tabu search strategy and vehicle insertion strategy have made a positive contribution to improving both convergence speed and solution quality. Furthermore, the CSO-TS algorithm exhibits enhanced performance over the standard TS, achieving higher solution precision at a notably more efficient computational pace. This advantage underscores the capability of the CSO's distinctive pairwise competition strategy in effectively handling large scale optimization problems.

4.4. Comparison with other algorithms

As mentioned earlier, considering the representative algorithms that have been successfully applied in various path planning scenarios, we selected 9 comparison algorithms, including an ant colony algorithm [10], a local search algorithm [35], a simulated annealing algorithm [36], two genetic algorithms [12, 37], and four other heuristic algorithms [38–41]. Finally, the best average running results for each subclass (R1, R2, C1, C2, RC1, and RC2) were presented. Additionally, the best known solutions were adopted in the experiments. Notably, the experimental parameters of each participating algorithm are consistent with those in the corresponding literature.

In Table 3, we present a comparison between the proposed algorithm and the most widely recognized benchmark results. In this context, "VC" and "DR" denote the best-known vehicle count and the relevant shortest travel distance, respectively. "BVC" refers to the best vehicle count achieved by the algorithm, while "BDR" indicates the minimum travel distance obtained under the same number of vehicles. "MVC" and "MDR" represent the average vehicle count and the average travel distance derived from the algorithm, respectively. The "Deviation" is calculated as (BDR - DR) / BDR and reflects the percentage deviation of the algorithm's results from the best-known solutions under the same vehicle count, serving as a measure of the algorithm's solution quality. "Time" shows the average computational time per generation of the algorithm. The row where "reference" appears indicates the source of the algorithm for the most well-known solution for each dataset.

Table 3. Comparison between best-known solutions, and CSO-TS results.

Dataset	Best-known			CSO-TS					
	VC	DR	references	BVC	BDR	MVC	MDR	Deviation	Time (s)
C101	10	828.94	[42]	10	828.94	10	828.94	0	1.2
C102	10	828.94	[42]	10	828.94	10	828.94	0	1.1
C103	10	828.06	[42]	10	844.35	10	846.21	1.97%	1.3
C104	10	828.78	[42]	10	833.67	10	837.08	0.59%	1.2
C105	10	828.94	[42]	10	828.94	10	828.94	0	1.1
C106	10	828.94	[42]	10	828.94	10	828.94	0	1.1
C107	10	828.94	[42]	10	828.94	10	828.94	0	1.2
C108	10	828.94	[42]	10	828.94	10	828.94	0	1.1
C109	10	828.94	[42]	10	828.94	10	828.94	0	1.2
C201	3	591.56	[42]	3	591.56	3	591.56	0	1.1
C202	3	591.56	[42]	3	591.56	3	591.56	0	1.1
C203	3	591.17	[42]	3	591.17	3	591.17	0	1.3
C204	3	590.60	[42]	3	590.60	3	590.60	0	1.2
C205	3	588.88	[42]	3	588.88	3	588.88	0	1.1
C206	3	588.49	[42]	3	588.49	3	588.49	0	1.2
C207	3	588.29	[42]	3	588.29	3	588.29	0	1.1
C208	3	588.32	[42]	3	588.32	3	588.32	0	1.1
R101	18	1613.59	[43]	19	1645.09	19	1651.10	1.95%	1.6
R102	17	1486.12	[42]	17	1486.12	17	1486.12	0	1.6
R103	13	1292.68	[44]	13	1304.27	13	1328.09	0.90%	1.6
R104	9	1007.24	[45]	10	999.51	10	1033.96	–	1.6
R105	14	1377.11	[42]	14	1377.11	14	1377.11	0	1.6
R106	12	1251.98	[45]	12	1264.64	12	1266.35	1.01%	1.6
R107	10	1104.66	[46]	11	1099.17	11	1107.66	–	1.2
R108	9	960.88	[47]	10	967.15	10	979.13	0.65%	1.2
R109	11	1194.73	[48]	11	1220.18	11	1223.11	2.13%	1.8
R110	10	1118.59	[45]	11	1139.18	11	1157.74	1.84%	1.5
R111	10	1096.72	[49]	10	1122.61	10.6	1115.75	2.36%	1.5
R112	9	982.14	[50]	10	974.95	10	982.49	–	1.6
R201	4	1252.37	[48]	4	1252.37	4	1254.05	0	1.2
R202	3	1191.70	[49]	3	1203.82	3.4	1179.85	1.02%	1.6
R203	3	939.54	[45]	3	949.54	3	955.83	1.06%	1.5
R204	2	825.52	[51]	2	861.22	2.8	804.24	4.32%	1.7
R205	3	994.42	[49]	3	1021.25	3	1067.22	2.70%	1.9
R206	3	906.14	[52]	3	944.96	3	986.44	4.28%	1.7
R207	2	837.20	[53]	3	877.91	3	899.33	4.86%	1.5
R208	2	726.75	[45]	3	761.03	3	794.19	4.72%	2.1
R209	3	909.16	[54]	3	930.33	3	935.92	2.33%	1.8
R210	3	938.58	[55]	3	973.15	3	977.59	3.68%	1.7
R211	2	892.71	[51]	3	937.01	3	951.42	4.96%	1.5
RC101	14	1696.94	[56]	14	1721.63	14	1763.73	1.45%	1.6
RC102	12	1554.75	[56]	13	1591.35	13	1607.13	2.35%	2.0
RC103	11	1261.67	[56]	11	1317.09	11	1367.33	4.39%	1.2
RC104	10	1135.48	[57]	10	1135.48	10	1135.48	0	1.6
RC105	13	1629.44	[47]	14	1586.58	14	1592.34	–	1.8
RC106	11	1424.73	[47]	11	1484.10	11	1508.85	4.17%	2.0
RC107	11	1222.10	[55]	11	1266.63	11	1310.71	3.64%	1.7
RC108	10	1139.82	[56]	11	1121.51	11	1147.76	–	1.5
RC201	4	1406.91	[45]	4	1406.91	4	1406.91	0	1.1
RC202	3	1365.65	[58]	4	1288.67	4	1287.93	–	1.5
RC203	3	1049.62	[58]	3	1078.84	3	1168.74	2.78%	1.8
RC204	3	798.41	[45]	3	809.47	3	815.57	1.39%	1.5
RC205	4	1297.19	[45]	4	1297.19	4	1297.19	0	1.9
RC206	3	1146.32	[54]	3	1146.32	3	1146.32	0	1.6
RC207	3	1061.14	[51]	3	1061.14	3	1061.14	0	1.8
RC208	3	828.14	[58]	3	840.28	3	855.58	1.47%	1.6

Table 4. Comparison of ACS-BSO, MOLNS, N-CLPSO and CSO-TS.

	ACS-BSO			MOLNS			N-CLPSO			CSO-TS		
Dataset	VC	DR	Time (s)	VC	DR	Time (s)	VC	DR	Time (s)	VC	DR	Time (s)
C101	**10**	**828.94**	1.6	**10**	**828.94**	0.9	**10**	**828.94**	1.2	**10**	**828.94**	1.1
C102	**10**	**828.94**	1.4	**10**	**828.94**	1.1	**10**	**828.94**	1.1	**10**	**828.94**	1.1
C103	**10**	**828.96**	1.5	10	828.94	1.0	10	839.35	1.3	10	844.35	1.3
C104	**10**	**828.78**	1.7	10	828.94	1.0	10	833.67	1.2	10	833.67	1.2
C105	**10**	**824.94**	1.6	**10**	**828.94**	0.9	**10**	**828.94**	1.1	**10**	**828.94**	1.1
C106	**10**	**828.94**	1.4	**10**	**828.94**	0.9	**10**	**828.94**	1.1	**10**	**828.94**	1.1
C107	**10**	**828.94**	1.5	**10**	**828.94**	0.9	**10**	**828.94**	1.2	**10**	**828.94**	1.2
C108	**10**	**828.94**	1.5	**10**	**828.94**	0.9	**10**	**828.94**	1.1	**10**	**828.94**	1.1
C109	**10**	**828.94**	1.5	**10**	**828.94**	0.9	**10**	**828.94**	1.2	**10**	**828.94**	1.2
C201	**3**	**591.56**	1.8	**3**	**591.56**	1.0	**3**	**591.56**	1.1	**3**	**591.56**	1.1
C202	**3**	**591.56**	1.8	**3**	**591.56**	1.0	**3**	**591.56**	1.1	**3**	**591.56**	1.1
C203	**3**	**591.17**	1.9	3	591.56	1.0	**3**	**591.17**	1.3	**3**	**591.17**	1.3
C204	**3**	**591.60**	1.6	**3**	**590.60**	1.0	**3**	**590.60**	1.2	**3**	**590.60**	1.2
C205	**3**	**588.88**	1.1	**3**	**588.88**	0.9	**3**	**588.88**	1.2	**3**	**588.88**	1.1
C206	**3**	**588.49**	1.5	**3**	**588.49**	0.8	**3**	**588.49**	1.2	**3**	**588.49**	1.2
C207	**3**	**588.29**	1.0	**3**	**588.29**	0.8	**3**	**588.29**	1.1	**3**	**588.29**	1.1
C208	**3**	**588.32**	1.0	**3**	**588.32**	0.8	**3**	**588.32**	1.1	**3**	**588.32**	1.1
R101	19	1671.16	2.1	19	1654.93	1.3	19	1648.08	1.7	**19**	**1645.09**	1.6
R102	17	1504.60	2.1	18	1475.33	1.0	**17**	**1486.12**	1.5	**17**	**1486.12**	1.6
R103	14	1245.86	2.9	14	1276.44	1.4	**13**	**1299.99**	1.8	13	1304.27	1.6
R104	11	1010.73	1.9	10	1010.72	1.3	**10**	**996.27**	2.1	10	999.51	1.6
R105	15	1366.05	1.9	15	1389.85	0.9	**14**	**1377.11**	1.4	**14**	**1377.11**	1.6
R106	13	1288.84	2.1	13	1269.14	1.0	**12**	**1262.8**	1.3	12	1264.64	1.6
R107	11	1101.56	2.0	11	1102.72	1.0	11	1101.17	1.2	**11**	**1099.17**	1.2
R108	10	974.17	2.3	10	991.57	1.1	10	985.76	1.2	**10**	**967.15**	1.2
R109	12	1165.71	2.0	12	1177.76	1.0	11	1223.73	1.0	**11**	**1220.18**	1.8
R110	**11**	**1090.92**	2.0	12	1129.60	0.7	11	1101.49	1.8	11	1139.18	1.5
R111	11	1148.14	2.0	12	1108.70	0.9	11	1064.67	1.5	**10**	**1122.61**	1.5
R112	10	1094.53	2.0	**10**	**964.15**	1.1	10	980.35	1.6	10	974.95	1.6
R201	4	1336.05	2.4	4	1305.25	1.1	**4**	**1252.37**	1.2	**4**	**1252.37**	1.2
R202	4	1128.05	2.4	4	1059.67	1.1	3	1225.02	1.6	**3**	**1203.82**	1.6
R203	3	1020.10	2.3	4	915.43	1.2	3	962.25	1.6	**3**	**949.54**	1.5
R204	3	834.92	2.2	3	775.99	1.2	**3**	**766.13**	1.7	3	861.22	1.7
R205	3	1105.38	2.7	3	1075.10	0.8	3	1027.79	2.0	**3**	**1021.25**	1.9
R206	3	949.11	2.4	3	979.21	1.0	3	948.46	1.8	**3**	**944.96**	1.7
R207	4	812.35	3.0	**3**	**854.89**	1.1	3	872.4	1.5	3	877.91	1.5
R208	2	940.30	2.9	**2**	**754.99**	1.2	2	774.36	2.1	2	761.03	2.1
R209	3	1046.73	2.7	4	898.23	0.9	3	943.72	2.0	**3**	**930.33**	1.8
R210	3	1069.26	2.4	4	941.58	0.9	3	989.19	2.4	**3**	**973.15**	1.7
R211	3	836.36	2.1	3	838.14	0.8	**3**	**828.9**	1.7	3	937.01	1.5
RC101	16	1643.78	2.2	15	1662.56	0.8	15	1635.11	0.8	**14**	**1721.63**	1.6
RC102	14	1464.63	1.9	14	1486.35	1.1	**13**	**1503.42**	2.1	13	1591.35	2.0
RC103	**11**	**1275.64**	1.5	12	1291.95	1.3	11	1277.99	1.2	11	1317.09	1.2
RC104	10	1156.92	1.5	10	1162.53	1.2	10	1143.56	1.2	**10**	**1135.48**	1.6
RC105	14	1609.68	1.7	15	1604.53	0.9	14	1542.55	1.8	**14**	**1540.58**	1.7
RC106	13	1378.45	1.6	13	1400.09	1.0	12	1388.70	2.2	**11**	**1484.10**	2.0
RC107	11	1318.69	1.7	12	1295.55	1.2	11	1282.81	1.7	**11**	**1266.63**	1.7
RC108	11	1134.85	1.7	11	1205.13	1.2	11	1157.12	1.7	**11**	**1121.51**	1.5
RC201	4	1514.41	2.5	4	1497.89	0.9	**4**	**1406.91**	1.1	**4**	**1406.91**	1.1
RC202	4	1326.71	2.6	4	1199.53	1.3	**4**	**1169.67**	1.5	4	1288.67	1.5
RC203	3	1166.91	2.0	4	985.54	1.7	3	1082.57	2.0	**3**	**1078.84**	1.8
RC204	3	929.94	2.6	**3**	**805.46**	1.5	3	828.61	1.5	3	809.47	1.5
RC205	4	1869.91	2.4	5	1340.38	1.0	4	1299.76	1.9	**4**	**1297.19**	1.9
RC206	3	1237.21	2.3	3	1316.42	1.0	**3**	**1146.32**	1.6	**3**	**1146.32**	1.6
RC207	4	1039.59	2.6	4	1031.62	1.3	3	1095.67	1.9	**3**	**1061.14**	1.8
RC208	3	910.59	2.3	3	859.13	1.0	3	843.28	1.6	**3**	**840.28**	1.5

From Table 3, it can be seen that the CSO-TS algorithm found 23 optimal solutions. The average deviations of R1 and C1 are 0.15% and 0.28%, respectively, both of which are less than 1%. At the same time, the deviations of R2 and RC2 are 3.51% and 1.12%, respectively, indicating that CSO-TS has good performance in "1" type problems. However, in the "2" category problems with longer time windows, the performance of CSO-TS showed a slight decline. On instances with wider time windows, the performance advantage of the proposed CSO-TS algorithm is slightly reduced. This can be attributed to the fact that relaxed time-window constraints weaken the restrictive structure of the problem, thereby reducing the effectiveness of the diversity-driven and adaptive neighborhood mechanisms designed to exploit tight temporal constraints. Nevertheless, the algorithm remains competitive on these instances, indicating that it is particularly well suited for VRPTW scenarios with tighter and more restrictive time windows.

In Table 4, two sets of recently published state-of-the-art algorithms are selected as competitors to CSO-TS. Note that for each algorithm, the data prioritizes the minimum number of vehicles, followed by the shortest path length. The results indicate that CSO-TS achieved optimal performance in 41 out of 56 datasets, while adaptive scatter search–based bee swarm optimization (ASC-BSO) [59], multiobjective large neighborhood search (MOLNS) [35], and neighborhood-based comprehensive learning particle swarm optimization (N-CLPSO) [32] performed best in 19, 18, and 27 test problems, respectively. Among the three algorithms, CSO-TS obtained both the minimum number of vehicles and the shortest path length in most datasets. Although running time is directly associated with the encoding method and experimental equipment, the average iteration time of these three algorithms is also compared as a reference index. The average running time of CSO-TS for each instance is 1.45 seconds, which is comparable to 1.96 seconds for ASC-BSO, 1.04 seconds for MOLNS, and 1.51 seconds for N-CLPSO. This result underscores the efficiency improvement achieved by the pairwise competition mechanism in CSO. The enhanced strategies introduced in CSO-TS, while incurring a reasonable computational overhead, lead to a substantial improvement in solution quality. This trade-off is justified by the significant gains in final solution performance.

5. Conclusions and future research

In this paper, a hybrid CSO-TS algorithm is proposed to address the VRPTW. Two objectives are considered in the problem formulation: The primary objective is to minimize the number of vehicles utilized, while the secondary objective is to minimize the total travel distance. The proposed algorithm builds upon enhanced operators derived from the original SPSO framework and improved learning mechanisms within the conventional CSO paradigm. Furthermore, three novel strategies are integrated to substantially improve the overall performance of CSO-TS.

An effective vehicle insertion mechanism is designed to aggressively reduce the number of vehicles, enabling solutions to approach the optimal fleet size. Furthermore, a tabu search algorithm with a dynamic neighborhood operation strategy is incorporated to strengthen local search capability and improve solution refinement. In addition, a path diversity index is introduced to address the encoding incompatibility of CSO when applied to VRPTW, while simultaneously achieving a balanced trade-off between exploration and exploitation. Extensive computational experiments demonstrate that CSO-TS outperforms competing methods and achieves superior and more robust performance across benchmark instances. The vehicle insertion strategy facilitates convergence toward solutions with minimal vehicle usage, the dynamic neighborhood-based tabu search adaptively conducts local searches to identify

high-quality solutions, and the path diversity index effectively maintains population diversity and particle quality throughout the search process.

Although the effectiveness of the proposed strategies has been validated, the interactions among these components warrant further investigation, as a deeper understanding of their interdependencies is critical for handling large-scale VRPTW instances more effectively. In addition, experimental results indicate that CSO-TS performs particularly well on Type 1 instances with narrow time windows, while its performance is slightly diminished on Type 2 instances characterized by wider time windows. This performance discrepancy may be attributed to the vehicle insertion mechanism, which jointly considers time window and distance constraints and thus provides greater benefits under tighter scheduling conditions. Accordingly, future research will focus on systematically analyzing the interactions among the proposed strategies and enhancing the adaptability and robustness of CSO-TS across diverse VRPTW scenarios.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62106092), Natural Science Foundation of Fujian Province (2024J01822, 2024J01820, 2025J01981, 2025J01980) and Natural Science Foundation of Zhangzhou City (ZZ2024J28).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. W. Li, Z. Ma, N. Liu, Design of reverse logistics system for B2C e-commerce based on management logic of internet of things, *Int. J. Ship. Transport Logist.*, **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. https://doi.org/10.1287/mnsc.6.1.80
3. R. Ojeda, H. Brenner, C. Eduardo, Metaheuristic approaches for the stochastic capacitated multi-depot vehicle routing problem with pickup and delivery, *Expert Syst. Appl.*, **290** (2025), 128258. https://doi.org/j.eswa.2025.128258
4. H. Y. Jeong, B. D. Song, Optimizing urban logistics: Vehicle routing problem with underground transportation, *IEEE Trans. Intell. Transp. Syst.*, **26** (2025), 6393–6413. https://doi.org/10.1109/TITS.2025.3533477
5. C. Hua, Solving the problem of vehicle routing problem with time window via dual adaptive genetic algorithm, *IEEE Access*, **13** (2025), 96535–96543. https://doi.org/10.1109/ACCESS.2025.3575459

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. C. Yang, X. Ning, D. Yu, H. Sun, G. Kang, H. Wang, A divide-and-conquer strategy for integer programming problems, *Electron. Res. Arch.*, **33** (2025), 3950–3967. httpss://doi.org/10.3934/era.2025175
9. M. Alinaghian, E. B. Tirkolaee, 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. J. Gong, J. Zhang, O. Liu, R. Z. Huang, H. S. H. Chung, Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach, *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, **42** (2012), 254–267. https://doi.org/10.1109/TSMCC.2011.2148712
12. T. S. Khoo, B. B. Mohammad, 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. Y. Zou, P. Xu, H. Dai, Swarm optimization with intra- and inter-hierarchical competition for large-scale berth allocation and crane assignment, *IEEE Trans. Emerging Top. Comput. Intell.*, **9** (2025), 1307–1321. https://doi.org/10.1109/TETCI.2025.3529876
14. Q. Yang, W. N. Chen, J. D. Deng, A level-based learning swarm optimizer for large-scale optimization, *IEEE Trans. Evol. Comput.*, **22** (2018), 578–594. https://doi.org/10.1109/TEVC.2017.2743016
15. Q. Yang, W. N. Chen, T. Gu, Segment-based predominant learning swarm optimizer for large-scale optimization, *IEEE Trans. Cybern.*, **47** (2017), 2896–2910. https://doi.org/10.1109/TCYB.2016.2616170
16. N. Zeng, Z. Wang, W. Liu, H. Zhang, K. Hone, X. Liu, A dynamic neighborhood-based switching particle swarm optimization algorithm, *IEEE Trans. Cybern.*, **52** (2022), 9290–9301. https://doi.org/10.1109/TCYB.2020.3029748
17. 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
18. Y. Wang, L. Wang, Z. P. Peng, 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
19. A. Gupta, S. Saini, An enhanced ant colony optimization algorithm for vehicle routing problem with time windows, in *2017 Ninth International Conference on Advanced Computing (ICoAC)*, Chennai, India, (2017), 267–274. https://doi.org/10.1109/ICoAC.2017.8441175

20. H. Z. Zhang, L. Ma, Z. Y. 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
21. W. Q. Zhang, D. J. Yang, G. H. 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
22. S. Reong, H. M. Wee, Y. L. 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
23. P. Saksuriya, C. Likasiri, Hybrid heuristic for vehicle routing problem with time windows and compatibility constraints in home healthcare system, *Appl. Sci.*, **12** (2020), 6486. https://doi.org/10.3390/app12136486
24. 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
25. 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
26. R. Liu, Z. B. 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
27. P. G. Luan, N. T. Thinh, Hybrid genetic algorithm based smooth global-path planning for a mobile robot, *Mech. Based Des. Struct. Mach.*, **51** (2023), 1758–1774. https://doi.org/10.1080/15397734.2021.1876569
28. D. B. Blumenthal, N. Boria, J. Gamper, S. Bougleux, L. Brun, Comparing heuristics for graph edit distance computation, *VLDB J.*, **29** (2020), 419–458. https://doi.org/10.1007/s00778-019-00544-1
29. P. Z. Du, N. Liu, H. F. Zhang, J. F. Lu, An improved ant colony optimization based on an adaptive heuristic factor for the traveling salesman problem, *J. Adv. Transp.*, **2021** (2021), 6642009. https://doi.org/10.1155/2021/6642009
30. Z. H. Luo, L. Li, M. X. Zhang, Diversified top-k route planning in road network, *Proc. VLDB Endow.*, **15** (2022), 3199–3212. https://doi.org/10.14778/3551793.3551863
31. J. Kennedy, Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance, in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, **3** (1999), 1931–1938. https://doi.org/10.1109/CEC.1999.785509
32. Q. C. Wu, X. W. Xia, A neighborhood comprehensive learning particle swarm optimization for the vehicle routing problem with time windows, *Swarm Evol. Comput.*, **84** (2024), 101425. https://doi.org/10.1016/j.swevo.2023.101425
33. J. N. Liu, L. Tong, X. W. Xia, A genetic algorithm for vehicle routing problems with time windows based on cluster of geographic positions and time windows, *Appl. Soft Comput.*, **169** (2025), 112593. https://doi.org/10.1016/j.asoc.2024.112593

34. 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

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

36. D. F. Zhang, S. F. Cai, F. R. Ye, Y. W. Si, T. T. Nguyen, A hybrid algorithm for a vehicle routing problem with realistic constraints, *Inf. Sci.*, **394–395** (2017), 167–182. https://doi.org/10.1016/j.ins.2017.02.028

37. T. S. Khoo, B. B. Mohammad, 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

38. 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

39. Y. L. Lan, F. Liu, W. W. Y. Ng, Decomposition based multi-objective variable neighborhood descent algorithm for logistics dispatching, *IEEE Trans. Emerging Top. Comput. Intell.*, **5** (2021), 826–839. https://doi.org/10.1109/TETCI.2020.3002228

40. Z. X. He, K. Zhou, H. Shu, X. Chen, X. Y. Lyu, Multi-objective algorithm based on tissue P system for solving tri-objective optimization problems, *Evol. Intell.*, **16** (2023), 1–16. https://doi.org/10.1007/s12065-021-00658-y

41. W. B. Dong, K. Zhou, H. Q. Qi, A tissue P system based evolutionary algorithm for multi-objective VRPTW, *Swarm Evol. Comput.*, **39** (2018), 310–322. https://doi.org/10.1016/j.swevo.2017.11.001

42. Y. Rochat, E. D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, *J. Heuristics*, **1** (1995), 147–167. https://doi.org/10.1007/BF02430370

43. 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

44. H. B. 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

45. 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.

46. P. Shaw, A new local search algorithm providing high quality solutions to vehicle routing problems, in *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, **46** (1997).

47. J. Berger, M. Barkaoui, O. Bräysy, A route-directed hybrid genetic approach for the vehicle routing problem with time windows, *INFOR: Inf. Syst. Oper. Res.*, **41** (2003), 179–194. https://doi.org/10.1080/03155986.2003.11732675

48. J. Homberger, H. Gehring, Two evolutionary metaheuristics for the vehicle routing problem with time windows, *INFOR: Inf. Syst. Oper. Res.*, **37** (1999), 297–318. https://doi.org/10.1080/03155986.1999.11732386

49. L. M. Rousseau, M. Gendreau, G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, *J. Heuristics*, **1** (2002), 43–58. https://doi.org/10.1023/A:1013661617536

50. L. M. Gambardella, É. Taillard, G Agazzi, *MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows*, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 1999.

51. R. Bent, H. V. Hentenryck, A two-stage hybrid local search for the vehicle routing problem with time windows, *Transp. Sci.*, **38** (2004), 515–530. https://doi.org/10.1287/trsc.1030.0049

52. G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, Record breaking optimization results using the ruin and recreate principle, *J. Comput. Phys.*, **159** (2000), 139–171. https://doi.org/10.1006/jcph.1999.6413

53. 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://doi.org/10.1016/j.cor.2003.11.023

54. J. Homberger, Eine verteilt-parallele Metaheuristik, in *Verteilt-parallele Metaheuristiken zur Tourenplanung: Lösungsverfahren für das Standardproblem mit Zeitfensterrestriktionen*, (2000), 139–165. https://doi.org/10.1007/978-3-322-97815-8_4

55. 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

56. 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

57. Z. J. 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*, **19** (2002), 376–383. https://doi.org/10.1109/EMPDP.2002.994313

58. 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, *Transp. Sci.*, **39** (2005), 206–232. https://doi.org/10.1287/trsc.1030.0085

59. Y. Shen, M. Liu, J. Yang, 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