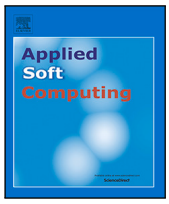
Applied Soft Computing Journal 169 (2025) 112593

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



A genetic algorithm for vehicle routing problems with time windows based on cluster of geographic positions and time windows

Jiani Liu [a], Lei Tong [b], Xuewen Xia [c,d,*]

[a] School of Mechanical and Electrical Engineering, Wuhan Business University, Wuhan 430056, China
[b] Hubei SME Mathematical Intellectualization Innovation Development Research Center, Wuhan Business University, Wuhan 430056, China
[c] College of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China
[d] Key Lab of Intelligent Optimization and Information Processing, Minnan Normal University, Zhangzhou 363000, China

ARTICLE INFO

Keywords:
VRPTW
Genetic algorithm
Cluster
LCS
Local search

ABSTRACT

The vehicle routing problem with time windows (VRPTW) has attracted many scholars' attention because it plays an important role in distribution and logistics. Many studies show that (meta-)heuristics are practical approaches for VRPTW. However, how to efficiently utilizing characteristics of customers' time windows and geographic distributions is neglected in last few years. Thus, this paper proposes an improved genetic algorithm (GA) for VRPTW based on a ***c***lustering method and the ***l***ongest ***c***ommon ***s***ubstring (LCS) of elite and inferior individuals. In the proposed algorithm, called CLCS-GA, cluster information of customers' geographic distributions and time windows is utilized to initialize three subpopulations with distinct properties. Moreover, during the evolutionary process, the LCS of elite and inferior individuals is utilized in the crossover and mutation operators to speed up the convergence and help individuals jump out of local optima. When performing the local search, which is a crucial operator for optimizing VRPTW, relatedness measured by customers' geographic distribution and time windows are considered, aiming to overcome the blindness of the common local search. Comprehensive properties of CLCS-GA are extensively testified by 56 VRPTW instances, in which seven state-of-art algorithms are adopted as peer algorithms. Moreover, distinct characteristics of the proposed strategies are also analyzed based on a set of experiments.

1. Introduction

The vehicle routing problem (VRP) is a popular combinatorial optimization problem. The objective of VRP is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. In recent years, VRP has attracted attentions of many scholars since similar optimization problems can be found in many application areas, including supply chain management [1,2], transportation [3], and telecommunication [4], et al.

Nowadays, different variants have extended the traditional VRP to satisfy the requirements of real applications. The VRP with time windows (VRPTW) is a popular one. In VRPTW, each customer can be serviced within a time interval, i.e., a time window. Regarding the VRP, each VRPTW is characterized by the following additional restrictions: (1) a solution becomes infeasible if a customer is supplied after the upper bound of its time window, (2) a vehicle arriving before the lower limit of the time window causes additional waiting time on the route, and (3) each route must start and end within the time window associated with the depot.

As VRPTW is NP-hard, deterministic algorithms generally cannot solve large-scale VRPTW. Hence, various evolutionary algorithms (EAs) have been proposed to solve it in the last few years [5]. As a famous EA, genetic algorithm (GA) has been applied to solve VRPTW [6–9]. Unlike unconstrained continuous problems, where GA has displayed very favorable performance, VRPTW has its own characteristics, such as multiple constraints, ambiguous concept of neighborhood, etc. Thus, some evolution operators in the traditional GA need to be modified to satisfy specific requirements of VRPTW. For example, traditional genetic operators, such as crossover and mutate operators, may generate infeasible individuals. Hence, various repair operators are proposed to address the issue [10,11]. Furthermore, many studies verify that it is also indispensable to enhance GA's performance on VRPTW by designing an efficient local search strategy according to the characteristics of VRPTW [12–14].

Obviously, if some helpful knowledge extracted from a VRPTW instance is utilized to improve the operators in GA, we can obtain a more efficient GA to optimize the VRPTW. Thus, in this study, some practical

* Corresponding author.
E-mail address: xwxia@whu.edu.cn (X. Xia).

https://doi.org/10.1016/j.asoc.2024.112593
Received 30 March 2024; Received in revised form 4 November 2024; Accepted 5 December 2024
Available online 14 December 2024

knowledge based on geographic distribution and time windows of customers, which are two essential features of customers, are used to facilitate initialization strategy, crossover operators, and mutation operators. Moreover, how to obtain some promising gene blocks and then protect them from damage during the evolution process are two crucial issues to improve the convergence speed. Intuitively, if some elites share the same gene block, there is an excellent probability that the real optimal solution contains the gene block. Thus, in this paper, the longest common substring (LCS) of different elite individuals can be regarded as a promising gene block. To speed up the convergence, the promising gene block is protected from damage with a higher probability when conducting evolutionary operators. Similarly, if different inferior individuals share the LCS, we can say that the real optimal solution does not contain the LCS with a higher probability. Thus, timely destroying the genes (i.e., the LCS) in the inferior individuals can help them jump out of local optima, and then search for more promising regions. Furthermore, considering there is blindness in traditional local search, we introduce a relativeness-based local search method, in which the relativeness between different customers is considered. In this case, a more purposeful local search can be obtained.

The remainder of the paper is organized as follows. Related works of VRPTW is introduced in Section 2, while the mathematical model of VRPTW is presented in Section 3. Motivations and details of our proposed algorithm, named as CLCS-GA, are explained in Section 4. Comparison experiments between CLCS-GA and other selected state-of-art algorithms are presented in Section 5. In addition, to figure out the properties of newly introduced strategies in CLCS-GA, sensitivity analyses of them are provided in this section based on the results of a set of experiments. Finally, some conclusions of the study are briefly summarized in Section 6.

2. Literatures review

During the last decades, various algorithms have been proposed to address VRPTW. Although traditional deterministic algorithms can offer favorable performance on some VRPTW, they do not yield promising results on large scale VRPTW [15–18] in a reasonable time. Thus, the majority of researches focus on utilizing various evolutionary algorithms (EAs) to optimize the problem in recent years [19–23].

2.1. Initialization methods

When utilizing EAs to deal with VRPTW, an initial population containing multiple solutions must be generated. Generally, randomly generating solutions is easily conducted. However, unlike solving many unconstrained optimization problems, the randomly generated solutions may violate some constraints in VRPTW. Thus, the some studies emphasize generating feasible solutions based on various strategies.

In 1987, Solomon introduced an efficient method, named as push-forward insertion heuristic (PFIH) [24], to generate a feasible solution for any VRP. The PFIH starts a new route by choosing an initial customer who is usually the farthest from the depot, and then inserts another unassigned customer into the current route until the capacity of the current vehicle or the time windows for the inserted customers are exceeded. Motivated by the study, other variants were proposed. For example, Caseau applied a heuristics insertion to initialize feasible individuals rather than random initialize individuals, in which time windows and incremental distance are considered [25]. In [26], Potvin introduced a greedy insertion heuristic to obtain an insertion order for customers, and then insert them one by one into a route. Although starting from feasible solutions, better ones can be obtained through a serial of operators, the time consuming of the initialization is very high.

2.2. Crossover methods

Generally, crossover is an essential operator in a GA. During the last few years, various crossover operators have been proposed [27] for combinatorial optimization problems. The one type of crossover operator is based on positions. For example, the position-based crossover (PBX) has been applied in many studies [28] since it is easily implemented. In PBX, first selects a subset of positions in the first parent, copies the components at these positions to the offspring at the same positions, and then fills the other positions with the remaining components in the same order as in the second parent. Moreover, other crossover operators, such as cycle crossover (CX) [29], order crossover (OX) [30,31], and position-based crossover (POS) [31], also consider the order of cities in parents after exchanging two selected subtours in the parents though different exchanging strategies are adopted in these crossover operators. Obviously, reserving the order of some cities can save some information of parents to offspring.

The other type of crossover operator is based on edges [32]. For instance, in the heuristic crossover (HC), a city is randomly selected as the start city of a tour. Then, a probability distribution is defined based on four edges incident to the city. Based on the distribution, an edge is selected and added into the tour. In Genetic edge recombination crossover (ER), which is suitable for the symmetrical TSP, it assumes that only the distances of edges are important, not their direction. Although ER has been verified by Larranaga et al. [33] as an outstanding crossover operator, it cannot be directly applied in VRPTW since time windows cause the directions of edges to be very important. The ER operator attempts to preserve the edges of the parents in order to pass on a maximum amount of information to the offspring. The breaking of edges is seen as an unwanted mutation. The ER operator does not consider the common sequences of the parent tours.

2.3. Mutation methods

In TSP or VRPTW, there are three popular mutation operators. The first one is called exchange mutation (EM) [34], in which two randomly selected two cities in a tour exchange their positions. At the beginning of the algorithm, a high number of exchanges was carried out. In this case, an individual has higher exploration ability. Via the algorithm, the number of exchanges was lowered to 1, which is beneficial for convergence. The second one is insertion mutation (ISM) [35], in which a city randomly removed from a tour is inserted in a randomly selected place of the tour. Although it is easily implemented, SIM may lose some promising knowledge in an elite solution. The third widespread mutation is scramble mutation (SM) [36], which selects a random subtour and scrambles the cities in it.

2.4. Local search methods

Since there are many constraints in VRPTW, operators in traditional EAs inevitably generate unfeasible and unfavorable offspring. Thus, various local search strategies and corresponding repair operators have proposed in the last decades. In fact, studies of local search strategies and repair operators have become the main research content in VRPTW [37,38]. In [25], for instance, the interplay between an insertion and the local optimization routines is considered when generating new solutions. In [39], the authors introduced an efficient local search with a large neighborhood based on pruning and propagation techniques of constraint programming. Based on the large neighborhood, the local search operator can find a better solution at each iteration with a higher probability. Furthermore, it can also reduce the requirements of specially-designed methods to avoid local minima. Although the local search can offer good results on Solomon class 1 problems where the time windows and capacities are tight and the solution consists of many small routes, it displays unfavorable performance on Solomon's class 2 problems. Moreover, the large neighborhood makes the search process time-consuming.

To overcome the shortcomings of the local search operator in [39], Potvin [26] proposed a heuristic method in which an incremental local search scheme called a look-ahead insertion algorithm and a constraint

solver are combined. Instead of visiting some large neighborhoods after an initial solution has been found, a limited number of insertions is examined, in which additional constraints are used to guide the insertion process. Meanwhile, the partial solution, rather than the entire solution, is examined after each insertion. In this case, the insertion operator is more time-efficient.

In recent years, the tabu search has offered auspicious performance in studies of VRPTW, and it has been adopted as an efficient local search operator to enhance the solutions' quality. For example, Lee [40] focused on designing a local search algorithm enhanced by a tabu structure to optimize VRPTW. Many studies verify that the tabu search strategy can offer promising and feasible solutions compared to other competing heuristics [41].

Considering that various local search strategies have their own merits, many researchers began to focus on integrating different local search strategies, aiming to take distinct advantages of them. For example, in [19], a local search repeatedly applies 2-opt, exchange, cross, and relocation operators until no better solution can be found. Similar to the study in [19], Wang [12] used six popular local search operators to exploit the best solution within each searching basin. Extensive experiments in these studies reveal that a hybrid local search mechanism can attain favorable solutions with less computational time for many testing instances. To bring more efficient and reliable performance of a local search strategy, Zhang [14] proposed an adaptive local search chain, in which different local search operators are utilized to search multiple directions. During the evolution, various helpful knowledge of obtained promising solutions can be adaptively selected for the subsequent local search operators. Extensive studies verify that problem-specific local search procedure can bring more robustness and reliability to different VRPTW problems [42].

From the above review, we notice that EAs are still prevalent in dealing with VRPTW since they can exhibit more promising performance than deterministic algorithms, measured by solution's accuracy and time-consuming. In these EAs-based studies, various insertion heuristics and repair operators are introduced to find feasible solutions, while distinct local search strategies are utilized to improve the solutions' quality. Although these methods offer positive performance on VRPTW, they cannot full use some kernel and helpful information of VRPTW et al. customers' geographical position and time windows. Thus, in this study, the information is utilized to depict relativeness between different customers, and then design efficient operators, such as initialization and local search strategy. Moreover, to improve the search capability, two types of longest common substrings (LCS), which has been successfully applied in some EAs [43,44], are introduced in this study.

3. VRPTW

VRPTW is a problem in which a fleet of K vehicles serves N customers. Each vehicle has a constant capacity Q. Generally, the ith ($i \in \{1, 2, \ldots, N\}$) customer denoted as a vertex c_i can be described by 4 attributions: (1) the geographic location, (2) the demand for goods q_i, (3) the delivery time window $[e_i, l_i]$ where e_i and l_i refer to the earliest time and the latest time when the customer starts service, and (4) the service time s_i. If a vehicle arrives at the customer c_i earlier than e_i, it must wait until the start of the time window to serve the customer. Note that $t_i = t_{i'} + w_{i'} + s_{i'} + d_{i'i}$ is a vehicle arrival time at c_i after servicing $c_{i'}$; The time cost between each pair of customers is represented by their Euclidean distance $d_{i'i}$ is the time cost between customers $c_{i'}$ and c_i, where $i' \neq i$ and $(d_{i'i} = d_{ii'})$, $i', i \in \{1, 2, \ldots, N\}$. Conversely, if the vehicle does not arrive before l_i, it cannot serve the customer c_i. For the convenience of description, the depot is denoted as c_0 whose demand q_0 is 0 and the time window is $[0, l_0 \geq \max(e_i)]$.

Generally, each VRPTW has two objectives defined as Eqs. (1) and (2), respectively. Considering that using as few vehicles as possible to satisfy customers' demand is a main target in real applications, the primary goal of this study 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.

Based on the aforementioned discussions, VRPTW can be mathematically formulated as follows.

Define variable

$$x_{ijk} = \begin{cases} 1, \text{if vehicle } k \text{ travels directly from } c_i \text{ to } c_j \\ 0, \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1, \text{if customer } c_i \text{ is served by vehicle } k \\ 0, \text{otherwise} \end{cases}$$

The goal of the VRPTW is to minimize

$$\min NV = K \tag{1}$$

and

$$\min TD = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} d_{ij} * x_{ijk} \tag{2}$$

$$s.t. \begin{cases} \sum_{i=0}^{N} x_{ijk} = y_{jk} \quad \forall k = 1, \ldots, K, \quad \forall j = 1, \ldots, N & \text{(a)} \\ \sum_{k=1}^{K} y_{ik} = 1 \quad \forall i = 1, \ldots, N & \text{(b)} \\ \sum_{i=0}^{N} y_{ik} * q_i \leq Q \quad \forall k = 1, \ldots, K & \text{(c)} \\ \sum_{k=1}^{K} y_{0k} = K & \text{(d)} \\ e_i \leq t_i \leq l_i \quad \forall i = 1, \ldots, N & \text{(e)} \end{cases} \tag{3}$$

where K and N are the numbers of vehicles and customers, respectively; t_i is a vehicle arrival time at c_i; $w_i = \max\{e_i - t_i, 0\}$ is the vehicle waiting time at the customer location to start the service time; s_i is the service time of c_i; e_i and l_i are the earliest arrival time and the latest arrival time at customer c_i, respectively.

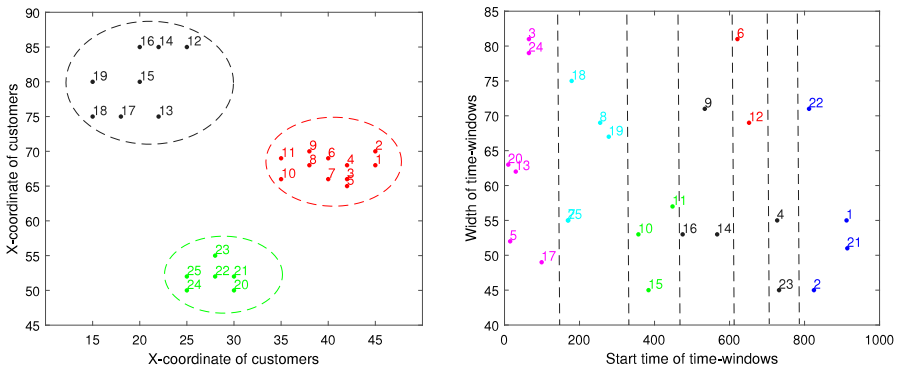
In Eq. (3), constraints (3a) and (3b) are used to ensure that each customer must be served by only one vehicle; constraint (3c) is used to limit that total number of requests carried by vehicle k must not exceed its capacity Q; constraint (3d) ensures that the number of all vehicles severing customers is K; constraint (3e) ensures that a vehicle will arrive at each customer within the limit time windows.

4. CLCS-GA

4.1. Initialization

Generally speaking, a random initialization is widely used in EAs not only because there is little prior information of a "black-box" problem, but also for it can provide a diversified population. Although a diversified population can cover a wider search space, it may be harmful to convergence speed. Thus, if much helpful information about a problem can be obtained in advance, it can be utilized to guide the initialization process and then enable an initial population near the optimal solution (or suboptimal solutions). As a result, the convergence process can be speeded up and solutions' accuracy can be improved.

Based on the characteristics of VRPTW, we regard that two customers can be served by a same vehicle with a high probability if the two customers are very close, in terms of the Euclidean distance. In addition, if the time windows of two customers are very close, using the same vehicle to serve the two customers may easily violate the constraints of the time window of the latter served customer. In other words, it is more probable that the two customers must be served by different vehicles. From the above preliminary discussions, we regard



(a) C101100 (b) C201100

Fig. 1. Clustering results of customers' positions and time windows in instance C10125.

that information of customers' geographical position and time windows can prevent invalid searches in infeasible space or unfavorable spaces.

To illustrate the hypothesis, we use a popular VRPTW instance, i.e., C10125, to analyze the characteristics of customers' positions and time windows. In Fig. 1, the clustering results of customers' positions and time windows are demonstrated. From Fig. 1(a), we can see that 25 customers in the instance can be clustered into three groups according to their geographical position. Intuitively, it is a promising method that using a vehicle to serve customers in the same group because the method can obtain routes with shorter distances. However, due to the constraint of time windows, the customers in the same group may not be served by a same vehicle. For instance, from Fig. 1(b) we notice that the customers are clustered into seven groups. In the figure, customers 5, 13, and 20 belong to the same group meaning their start times of time-windows are very close. In this case, utilizing the same vehicle to serve them is tough. Thus, to reduce the probability of violating constraints, utilizing a vehicle to serve customers in different groups is a promising choice. As a result, we think that extracting potential information about customers' geographical positions and time windows can bring more favorable results.

In CLCS-GA, according to the above discussions, we first apply the K-means algorithm to cluster customers according to their positions and time windows, respectively. Based on the two cluster results, two initial sub-populations, recorded as $SubP_p$ and $SubP_t$, can be obtained by Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Init_SubPp()

Input: Sub-population $SubP_p = \emptyset$; Sub-population size PS_p; Number of customers N; Cluster results of customers' positions C_p.
1: **for** $i = 1 : PS_p$ **do** //
2: **for** $j = 1 : |C_p|$ **do** // $|C_p|$ is the number of clusters
3: Generate the j^{th} route based on random permutated customers in the j^{th} cluster in C_p;
4: **end for**
5: Construct the i^{th} individual Ind_i based on the above generated routes;
6: $SubP_p = SubP_p \cup Ind_i$
7: **end for**
Output: Sub-population $SubP_p$.

Although the two sub-populations may be beneficial for convergence speed, a higher population diversity is also indispensable for optimizing a complicated VRPTW. Thus, a random initializing sub-population, denoted as $SubP_r$, is also applied in this study. Due to that, the random initialization method has been introduced in other studies, it is not described in this study.

According to the introduced above, three sub-populations, i.e., $SubP_p$, $SubP_t$, and $SubP_r$, with different characteristics are generated in the initial stage.

Algorithm 2 Init_SubPt()

Input: Sub-population $SubP_t = \emptyset$; Sub-population size PS_t; Number of customers N; Cluster results of customers' time windows C_t.
1: **for** $i = 1 : PS_t$ **do**
2: Initial an set $\gamma = \emptyset$;
3: **for** $j = 1 : |C_t|$ **do** // $|C_t|$ is the number of clusters
4: Randomly select a customer c_k $(1 \le k \le |C_t|)$ in the j^{th} cluster in C_t;
5: $\gamma = \gamma \cup c_k$; $C_t = C_t \backslash c_k$;
6: **end for**
7: Reorder customers in γ in ascending order of their time windows' start time;
8: Construct the i^{th} individual Ind_i based on γ;
9: $SubP_t = SubP_t \cup Ind_i$
10: **end for**
Output: Sub-population $SubP_t$.

4.2. The LCS of elite and inferior individuals

In this study, as introduced in Section 1, the LCS of elites can be regarded as a promising gene block that needs to be protected, while the LCS of inferiors can be seen as an unfavorable gene block that needs to be destroyed. Thus, the LCSs of elite and inferior individuals can be applied to enhance the convergence speed and help a population jump out of local optima, respectively.

To obtain the two different types of LCSs, individuals in the population must be sorted according to their fitness values. According to the sort results, n individuals who have the highest fitness values are used to generate the elites LCS, denoted as LCS_{elite}. Similarly, n individuals who have the lowest fitness values can be utilized to generate the inferiors LCS, named LCS_{infer}. In this study, n is set as 3 based on a trial-and-error method. Note that LCS_{elite} may be identical to LCS_{infer}. In this case, we can say that the characteristics of the LCS are unknown. In other words, we cannot draw a definitive conclusion that the LCS is a promising gene block or a harmful gene block. As a result, the same LCS (i.e., LCS_{infer} and LCS_{elite}) will be discarded.

Note that the dynamic programming is used to find the two types of LCSs efficiently. Considering that one can easily obtain details of the dynamic programming, we do not provide the detailed generation process of the LCS_{infer} and LCS_{elite} in this paper.

Obviously, different LCSs may be obtained in different evolution generations because different fitness landscapes have distinct properties. Thus, to take advantages of various LCSs in different generations, it is a comparatively feasible strategy that saving multiple LCSs into an archive. Concretely, in each a few generations, an obtained LCS is saved into an archive. While the archive is full, a first-in-first-out (FIFO) mechanism is used to delete some LCSs. In CLCS-GA, Arc_{elite} and Arc_{infer} are two archives used to save obtained LCS_{infer} and LCS_{elite}, respectively. The size of the two archives is set as N_{Arc}. The generation of LCS_{infer} and LCS_{elite} can be described as Algorithm 3.

Algorithm 3 LCS_Generation()

Input: Population P; population size PS; fitness F, size of archive N_{Arc}, elites archive Arc_e, inferiors archive Arc_e.

Sort P as $X_{i1}, X_{i2}, ..., X_{iPS}$ based on individuals' fitness F;
Obtain the LCS, named as LCS_{elite}, of elites (i.e., X_{i1}, X_{i2}, X_{i3});
Obtain the LCS, named as LCS_{infer}, of inferiors (i.e., $X_{iPS-2}, X_{iPS-1}, X_{iPS}$);
if LCS_{elite} is not equal to LCS_{infer} **then**
  if $|Arc_e| < N_{Arc}$ and $LCS_{elite} \notin Arc_e$ **then**
    $Arc_e = Arc_e \cup LCS_{elite}$;
  end if
  if $|Arc_i| < N_{Arc}$ and $LCS_{infer} \notin Arc_i$ **then**
    $Arc_i = Arc_i \cup LCS_{infer}$;
  end if
end if

Output: The generated archives Arc_e and Arc_i.

4.3. Crossover

When optimizing VRPTW, a simple and popular crossover operator is executed on two selected two chromosomes. Generally, there are different selections for chromosomes, such as random selection, roulette wheel selection, and tournament selection. In this study, considering that the tournament selection can give an inferior chromosome more chance to generate offspring and bring more population diversity, we utilize the tournament method to select chromosomes when performing the crossover operator. The details of the crossover operator are described as follows.

First, two parents in generation t, denoted as X_1^t and X_2^t, are selected based on the tournament selection strategy. Then, a random length of gene block, len ($1 \le len < LEN$) is determined, where LEN is the length of a chromosome. After that, the two parents, X_1^t and X_2^t, randomly select their gene blocks denoted as b_1 and b_2 with the same length len, respectively. Based on the two selected gene blocks, the two parents can perform the crossover operator. Concretely, the gene block b_1 is added to the left-most of X_2^t, while the gene block b_2 is added to the left-most of X_1^t. For convenience of description, the two parents with added blocks are denoted as $X_1'^t$ and $X_2'^t$, respectively. Obviously, $X_1'^t$ and $X_2'^t$ violate the constraint in VRPTW, since they have duplicated customers. Thus, a repair operator needs to be conducted. A simple repair operator is deleting some genes from a parent that are similar to that in the added gene block. After the repair operator, two new individuals, i.e., X_1^{t+1} and X_2^{t+1}, are generated. The detail of the simple repair operator can be described in Fig. 2.

Although the simple repair operator is easily performed, it may destroy some promising gene blocks in a parent. For example, when the parent has a gene block [c_i, c_j, c_k], which is (or is part of) a LCS_{elite}, the promising gene block will be destroyed if the added gene block has the customer c_j. In this case, a better alternative is deleting the duplicate customer c_j from the added gene block rather than deleting it from the parent. Thus, in CLCS-GA, two different conditions are considered when performing the repair operator. The one case is that a gene block composed of the duplicated customer c_j and its neighbors in a parent is (or is a part of) a LCS_{elite}. In this case, to save the gene block in the parent, the duplicated customer c_j in the added gene block is removed. The other case is that a gene block composed of the duplicated customer c_j and its neighbors is not in (or is not a part of) a LCS_{elite}. In this condition, to improve the exploration of the parent, the duplicated customer in it is removed. The new LCS-based repair process of X_1' in Fig. 2 can be illustrated by Fig. 3.

In Fig. 3, a parent denoted as [8, 9, 4, 7, 1, 2, 3, 5, 6] means the order of the customers serviced by a vehicle is: $depot$ →8→9→4→7→1→2→3→5→6, and the gene blocks selected from another parent is [3, 7, 2]. A LCS_{elite} is [8, 3, 5], which means that the sequence has appeared in many elite individuals. Before the repair process, we notice that three customers in X_1', i.e., 3, 7, and 2, are duplicated. Thus, we need to perform the repair process by removing the duplicated genes. From Fig. 3 we can see that, in the parent, the gene block [3, 5] is a part of the LCS_{elite}, which means the gene block should be saved. Thus, the gene 3 (i.e., the customer c_3) in the block [3, 7, 2] needs to be removed. On the contrary, in the parent, the genes 7 and 2 with their neighbors do not appear in the LCS_{elite}. As a result, the two duplicated genes in the parent need to be removed. Thus, after the repair process, an offspring can be obtained as [7, 2, 8, 9, 4, 1, 3, 5, 6].

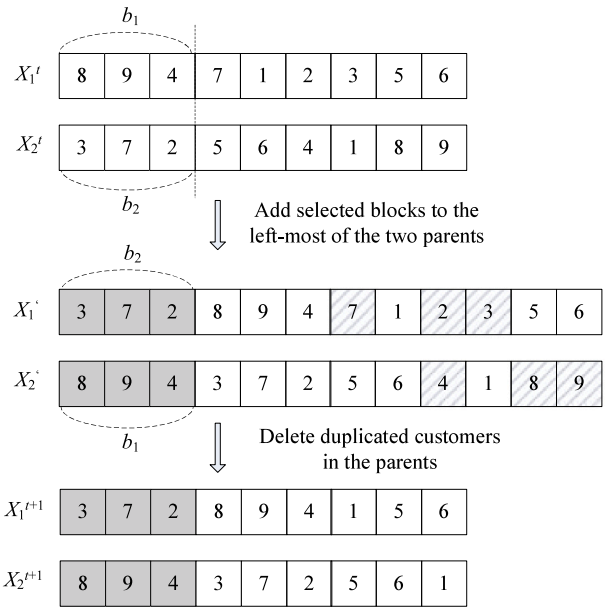


Fig. 2. The crossover process based on the simple repair method. In the example, the length of gene block is $len = 3$.

Based on the repair operator introduced above, the crossover operator in CLCS-GA can be described as Algorithm 4.

Algorithm 4 Crossover()

Input: Population P^t; fitness F, rate gap R, and archive A_{LCS}.

Select candidate parents CP based on fitness F, where the size of CP, $|CP| = |P^t| * R$;
Random divide CP into two subpopulations: $SubP_1$ and $SubP_2$ with the same size $|P^t| * R/2$, where $SubP_i$=[$Xi_1^t, Xi_2^t, ..., Xi_{|P|*R/2}^t$] ($i$=1, 2);
for $i = 1 : |P| * R/2$ **do**
  Select $X1_i^t$ from $SubP_1$, where $X1_i^t = [c_{1,1}, c_{1,2}, ..., c_{1,D}]$;
  Select $X2_i^t$ from $SubP_2$, where $X2_i^t = [c_{2,1}, c_{2,2}, ..., c_{2,D}]$;
  Randomly generate len, where $1 \le len < D$;
  Obtain a gene block b_1 from $X1_i^t$, where b_1=[$c_{1,1}, c_{1,2}, ..., c_{1,l}$];
  Obtain a gene block b_2 from $X2_i^t$, where b_2=[$c_{2,1}, c_{2,2}, ..., c_{2,l}$];
  Generate two temporary children: $X1_i'$=[$b_2, X1_i$], $X2_i'$=[$b_1, X2_i$];
  Perform the repair process on $X1_i^{t+1}$ and $X2_i^{t+1}$ according to Fig. 3;
end for
Randomly generate $|P| - |CP|$ individuals;
Gather $X1_i^{t+1}$, $X2_i^{t+1}$ ($1 \le i \le |P| * R/2$), and the randomly generated individuals as a new population P^{t+1};

Output: The new population P.

4.4. Mutation

When performing a mutation operator, a popular strategy is exchanging two randomly selected genes. However, it should be considered that a population must perform different search behaviors. In the early evolution stages, the population needs to pay more attentions to exploration. Meanwhile, an individual may violate some customers' time windows constraints in the evolution stage. Thus, we regard exchanging two customers randomly selected from two different routes is beneficial for exploration capability and searching for feasible solutions in the early evolution stage. On the contrary, the population should pay more attention to exploitation and shorten the distance of a feasible route in the later evolution stage. Hence, in the later evolution stage,

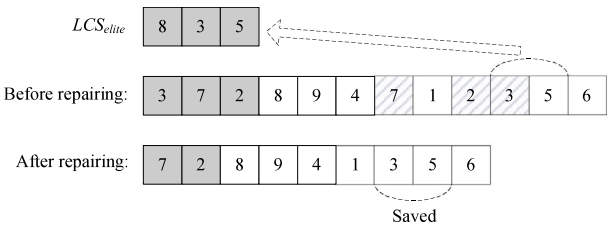


Fig. 3. The repair processes based on a LCS_{elite} selected from Arc_e.

it is a promising choice to randomly select two customers in the same route to exchange their values.

Similar to saving some promising gene blocks in the crossover operator introduced in Section 4.3, destroying some unpromising gene blocks by the mutation operator can help an individual look for better solutions. But how to define an *unpromising* gene block? Intuitionally, if a gene block appears in many inferior individuals, the gene block can be regarded as an *unpromising* gene block. Thus, to obtain the unpromising gene block, the LCS of many inferior individuals, denoted as LCS_{infer}, should be obtained in each generation. And then, destroying LCS_{infer} in an individual who has the gene block is beneficial for the individual to jump out of local optima and search for more promising regions.

Based on the discussions, the mutation operator in CLCS-GA can be illustrated by Algorithm 5. The step 2 to step 5 in the algorithm mean that two genes are randomly selected from different routes in the early evolution stage or the individual violates constrains. The aim of the operator is to enhance the exploration capability and search for feasible solutions in the early stage. Meanwhile, step 8 to step 10 indicate that two customers are randomly selected from the same route in the later evolution stage to perform the mutation operator. Based on the operator, the individual can pay more attention to exploitation and shorten the distance of it. After exchanging the two randomly selected genes, some unpromising gene blocks in the individual needed to be destroyed with probability p_d (see step 13 to step 19). Hence, the individual with unpromising gene blocks can jump out of local optima and search for more favorable regions.

Algorithm 5 Mutation()

Input: The population P^t; current generation t; the maximum generation T, destroy ratio p_d, and an archive LCS_{infer}.
1: **for** Each individual X in population P^t **do**
2: **if** $t/T < 0.5$ or X violates constrains **then**
3: Random select two different routes $r1$ and $r2$ in X;
4: Random select two customers c_i and c_j from $r1$ and $r2$, respectively;
5: Exchange c_i and c_j in X;
6: **else**
7: **for** Each route r in X **do**
8: Random select a routes r in X;
9: Random select two different customers c_i and c_j from r;
10: Exchange c_i and c_j;
11: **end for**
12: **end if**
13: **if** $rand < p_d$ **then**
14: Random select a LCS_{infer} from the archive Arc_{infer} ;
15: **if** There is a block $[c_i, c_{i+1}]$ $(1 \le i < D)$ in X is a part of LCS_{infer} **then**
16: Random select a customer c_j $(c_j \neq c_i)$;
17: Exchange c_i and c_j;
18: **end if**
19: **end if**
20: **end for**
Output: The population P.

4.5. Local search

Local search is a popular and essential strategy for EAs to deal with VRPTW. In CLCS-GA, the local search operator consists of two steps, i.e., removing a few selected customers from routes and reinserting the customers into the routes.

4.5.1. Removing customers

When removing a few customers from a route, two issues must be dealt with. The one is how many customers should be removed, and another one is how to select them from the route.

For the first issue, obviously, removing more customers from a route and reinserting them into the route can bring a larger local search range, which is beneficial for improving the solution's accuracy though it is more time-consuming. On the contrary, removing fewer customers can save some computational resources, but it can only yield a smaller local search range. Generally, a VRPTW with more customers has a larger search space, which is more difficult to be optimized. In this case, we regard that the local search operator with more selected customers can bring more improvements with a greater local search range. In other words, the number of removed customers should be correlated with the total number of customers in a VRPTW. Thus, for a VRPTW with N customers, $N/10$ customers are selected to be removed when performing the local search operator.

For the second issue, i.e., how to select the customers from a current route, a simple method is randomly selecting them. However, this method may cause the local search blindly. Hence, in CLCS-GA, customers with high relatedness are removed from the route.

In this study, the relatedness value of customers c_i and c_j denoted $\mathfrak{R}_{ij}$ is determined by three factors. The first factor is the Euclidean distance between the two customers, which is the most intuitive measurement. The second factor is whether the two customers are on the same route. The last factor is whether the two customers belong to the same cluster, measured by the time windows.

Based on the above discussions, we define $\mathfrak{R}_{ij}$ as Eq. (4).

$$\mathfrak{R}_{ij} = \frac{1}{rd_{ij} + sr_{ij} + tc_{ij}} \tag{4}$$

where rd_{ij}, sr_{ij}, and tc_{ij} are defined as Eqs. (5), (6), and (7), respectively.

$$rd_{ij} = \frac{d_{ij}}{max\{d_{ik} | 1 \le k \le N\}} \tag{5}$$

where d_{ij} means the distance between c_i and c_j.

$$sr_{ij} = \begin{cases} 1, & \text{if customers } c_i \text{ and } c_j \text{ belong to a same cluster} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$tc_{ij} = \begin{cases} 1, & \text{if time windows of } c_i \text{ and } c_j \text{ belong to a same cluster} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

From Eq. (4), we can see that (1) the lower the distance between two customers is, the higher relativeness of them is, (2) if two customers belong to the same route, the relativeness of them is higher, and (3) if the time windows of two customers belong to different clusters, the relativeness of them is greater.

Based on the definition of $\mathfrak{R}_{ij}$ (see Eq. (4)), the relativeness-based removing process can be illustrated by Algorithm 6.

Algorithm 6 Removing()

Input: An individual X and a set of its neighbors $\mathcal{N} = \emptyset$.
1: $Ns = \lceil N/10 \rceil$; // N is the number of customers in the individual X
2: $\check{X} = X$;
3: Random select a customer c_{r1} $(1 \le r1 \le D)$ in X;
4: $\mathcal{N} = \mathcal{N} \cup \{c_{r1}\}$; $\check{X} = \check{X} \setminus \{c_{r1}\}$;
5: **for** $i = 2 : Ns$ **do** // Ns is the number of customers should be removed
6: Random select a custom c_r from $Neighbors$;
7: Calculate the relatedness between c_r and other customers in $\check{X}$ based on Eq. (4);
8: Select a customs c_{ri} in $\check{X}$ who has the highest relatedness with c_r;
9: $\mathcal{N} = \mathcal{N} \cup \{c_{ri}\}$; $\check{X} = \check{X} \setminus \{c_{ri}\}$;
10: **end for**
Output: Temp individual $\check{X}$, $\mathcal{N}$.

4.5.2. Reinsert customers

To obtain an improved individual, the removed customers in $\mathcal{N}$ (see Algorithm 6) need to be reinserted into $\check{X}$ at suitable positions. In CLCS-GA, each customer c_i in $\mathcal{N}$ needs to find a *proper* insertion position in $\check{X}$. Obviously, how to define the *proper* is crucial for the reinsertion operator. In this study, according to the objectives of VRPTW, if customer c_i is reinserted at position p of route r in $\check{X}$, the time window constraints of all customers in route r should be considered firstly.

For convenience of description, the position p is named as a feasible insertion point, and the distance increment of route r after the customer c_i being reinserted in the position p is denoted as $DI_{i,p}$. While the customer c_i has multiple feasible insertion points, we need to choose an optimal point p^* for the customer c_i where DI_{i,p^*} has the minimum value. When all the customers c_i ($1 \le i \le Ds$) have detected their optimal points DI_{i,p^*}, the customer c_{i^*} who has the shortest DI_{i^*,p^*} needs to be reinserted at position p^* of the corresponding route, and then the customer c_{i^*} is removed from the set $\mathcal{N}$. If there is no feasible insertion point for the customer c_i, a new route needs to be generated, i.e., an additional car needs to be dispatched for servicing the customer c_i. After that, another customer randomly selected from the set $\mathcal{N}$ will perform similar detection and reinsertion operators until the set $\mathcal{N}$ is empty. The reinsertion operator can be described as Algorithm 7.

Algorithm 7 Reinsertion()

Input: A temp individual $\check{X}$; a set of neighbors $\mathcal{N}$.
1: **while** $\mathcal{N} \neq \varnothing$ **do**
2: $DI = \infty$; $p^* = -1$;
3: **for** $i = 1 : |\mathcal{N}|$ **do**
4: Random select a customer c_j from $\mathcal{N}$;
5: **for** Each route r in $\check{X}$ **do**
6: Find out an insertion point p in the route r for the customer c_j;
7: **if** p is a feasible insertion point **then**
8: Calculate the distance increment $DI_{j,p}$ for the point p;
9: **if** $DI_{j,p} < DI$ **then**
10: $DI = DI_{j,p}$; $c_{j^*} = c_j$; $p^* = p$; $r^* = r$;
11: **end if**
12: **end if**
13: **end for**
14: **if** $DI \neq \infty$ **then**
15: Reinsert the customer c_{j^*} at position p^* of route r^* in $\check{X}$;
16: **else**
17: Create a new route for the customer c_{j^*};
18: **end if**
19: $\mathcal{N} = \mathcal{N} \setminus \{c_{j^*}\}$;
20: **end for**
21: **end while**
Output: An improved individual X=$\check{X}$.

4.6. Evaluation criteria

As introduced in Section 3, *NV* and *TD* are two objectives in a VRPTW. During the last few years, various multiobjective evolutionary algorithms (MOEAs) have been proposed to deal with VRPTW. For example, Wang proposed a hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows (MOPVRPTW) [22], in which five objectives (i.e., the number of vehicles, total travel distance, makespan, total waiting time, and total delay time) are considered. Obviously, there are some conflicts between different objectives. Generally speaking, for example, less vehicles may extend the total waiting time and total delay time. On the contrary, more vehicles can shorten the two objectives, while it may increase the total travel distance. In this case, using a MOEA is a great choice.

However, in our study, the two objectives in VRPTW are not always conflict with each other. Concretely, we found through experiments that more vehicles yield longer total distance in the majority VRPTW instances. Only a few instances have multiple nondominated solutions. For example, in the 17 C-type instances, we only find out 2 nondominated solutions on 3 instances, i.e., C103100 (NV = 10, TD = 815.44 VS. NV = 9, TD = 947,68), C104100 (NV = 10, TD = 806.84 VS. NV = 9, TD = 823.33), C109100 (NV = 10, TD = 820.40 VS. NV = 9, TD = 853.70). Comparing the results, we notice that using more vehicles can only obtain a slightly shorter total distance in the 3 instances. In other words, the solutions are hard to be adopted by a logistics company in real applications.

Moreover, we regard the two objectives, i.e., *NV* and *TD*, in a VRPTW as having different weights in real applications. For a logistics company, for example, using a fewer vehicles is more attractive even the total distance of the vehicles is slightly longer because the cost increase of vehicles is far higher than that of running distance. Considering the factor, we regard that the primary objective of a VRPTW is to minimize the number of vehicles, while the secondary objective is to reduce the total distance. In other words, if two feasible solutions have different numbers of vehicles, the solution with fewer vehicles is better than the other one. While the two solutions have the same number of vehicles, the solution that has the shorter total distance is better than the other one.

However, designing an evaluation criterion for solutions is a crucial problem since it determines the population's evolutionary direction. Thus, in this study, one of the main objectives of the evaluation criterion is to "guide" the population evolves from unfeasible regions towards feasible regions. A simple evaluation criterion based on a penalty-function method is defined as Eq. (8) in this study.

$$f(X_i) = NV \times (TD + \varpi \times T_{delay}) \tag{8}$$

where NV and TD are the number of vehicles and the total distance of the route X_i; ϖ is a penalty coefficient, which is set as 2 in this study through the trail-and-error method; T_{delay} is the total delay time of all customers.

4.7. Flowchart of CLCS-GA

Based on the above discussions, the flowchart of CLCS-GA can be demonstrated in Fig. 4. From the figure we notice that there are six new proposed parts in CLCS-GA. The first and the second parts are the clustering and the initialization processes, respectively, executed only one time in the algorithm. In the third part, two archives applied to save LCS_{elite} and LCS_{infer} are obtained based on the evaluation results of individuals. After that, the last three parts, i.e., crossover, mutation, and local search operators, are performed in sequence.

4.8. Time complexity of CLCS-GA

From Fig. 4 we observe that there are six main parts in CLCS-GA. Thus, we first analyze the time complexity of each part, and then give an overall time complexity of CLCS-GA. For convenience, we assume that the population size of candidate solutions is PS, the number of customers is N, and the maximum generations is T.

The first part is the clustering process, in which the K-means is used to cluster the customers' positions and time windows. Thus, the complexity of the clustering process is $O(N)$. The second part is initialization, in which the time complexities of location-based and the time windows-based generation processes both are $O(N * K)$, where K is the number of clusters, and the time complexity of the random-based generation process is $O(N)$. In this study, K is set as 10. Thus, the time complexity of the initialization is $O(N)$. In the third part, including sorting process and the update of two achieves, the time complexity is $O(PS \log PS + N^2)$. The fourth part is the crossover, the time complexity of which is $O(PS)$. The mutate operator is the fifth part, and its time complexity is $O(PS * NV)$, where NV is the number of vehicles. In this study, the maximum number of vehicles is set as 20. Thus, the time complexity of the mutate operator can be simplified to $O(PS)$. The last part is the local search part, which is the most time consuming in CLCS-GA. The part includes the removing process and the reinsertion process. Thus, the time consuming of it is $O(PS * N * (NV + N)) + O(N^2)$. Since the maximum of NV is set as 20, which has been discussed above, the time consuming of the local search part is $O(PS * N^2)$. According to the above discussion, the time complexity of CLCS-GA in each generation can be simplified as $O(PS * N^2 + PS \log PS + N^2)$. Hence, the entire time complexity of CLCS-GA is $O(T * (PS * N^2 + PS \log PS + N^2))$.

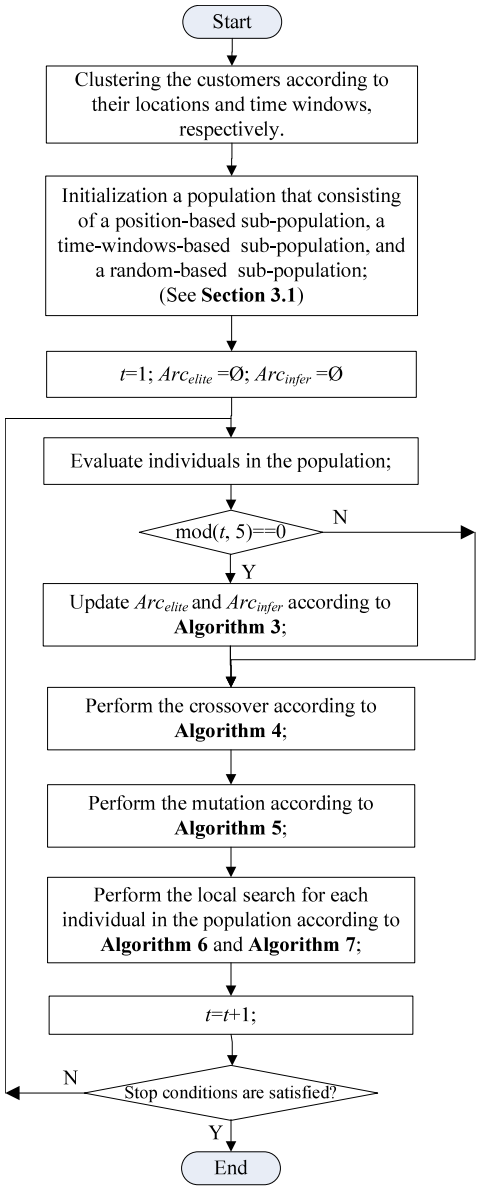


Fig. 4. The flowchart of CLCS-GA.

5. Experiments and discussions

5.1. Benchmark instances and experimental settings

To testify the comprehensive performance of CLCS-GA, six sets of VRPTW instances [45], named R1, R2, C1, C2, RC1, and RC2, are adopted in this study. The six types of instances have their own distinct characteristics. Concretely, customers' geographical data are randomly generated in R1 and R2, while customers' geographical data are clustered in C1 and C2. In RC1 and RC2, customers' geographical data are a mix of random and clustered structures. Moreover, instances in C1, R1 and RC1 have smaller vehicle capacities and more compact time windows. On the contrary, instances in C2, R2, and RC2 have larger vehicle capacities and longer dispatch cycles. We believe these instances can fully examine the overall performance of CLCS-GA with various properties.

In this study, the programming language is MATLAB R2022a; the simulation experiments are carried out on a PC equipped with a Core i7 4.2 GHz processor, 16 GB of RAM, and Windows 10. Parameters of GLCS-GA are defined as: sub-population size $PS_p = PS_t = PS_r = 50$, the stop condition is a predefined maximum generation $T = 10\,000$, crossover probability $p_c = 0.9$, mutation probability $p_m = 0.1$, rate gap $R = 0.9$, and destroy ratio $p_d = 0.1$. To fairly compare the performance among all the algorithms, each peer algorithm carried out 30 independent runs on each VRPTW instance.

5.2. Sensitivity analysis of proposed strategies in CLCS-GA

In CLCS-GA, there are three new proposed strategies, i.e., (1) the hybrid initialization method, (2) the LCS-based crossover and mutation operators, and (3) the relatedness-based removing strategy in the local search process. To verify the performance of the three proposed strategies, three sets of experiments are conducted.

5.2.1. Performance of the hybrid initialization strategy

In CLCS-GA, the initial population consists of three sub-populations which based on three different initialization methods, i.e., the random-based, the distance-based, and the time windows-based methods.

To verify the performance of the proposed initialization method, in the revised manuscript, we carry out a set of experiments to testify the first passage time. In the experiments, 3 types of instances (10 C-type instances, 10 R-type instances, and 10 RC-type instances) are adopted as benchmark functions, while *cluster* and *rnd* denote the initialization method in CLCS-GA are clustering-based and random-based initialization methods, respectively. For each instance, both the two algorithms are carried out 20 independent runs. Statistical results between the two algorithms are presented in Tables 1–3. The results include the generation (*Gen*) that an algorithm firstly finds out feasible solutions, the shortest distance of these obtained feasible solutions (D_{short}), and the average distance of these feasible solutions (D_{avg}) on 20 runs.

From Table 1 we can see that the clustering-based initialization method applied in our study exhibits more outstanding performance on the C-type instances. Concretely, it dominates the randomly-based initialization method on 9 out of the 10 instances, measured by all the 3 metrics. In other words, information of two clustering results, i.e., distances clustering and time windows clustering, can help a population find shorter routes with fewer generations. The favorable performance of the clustering-based initialization method is attributed to the fact that customers' geographical data are clustered in the selected C-type instances.

From Table 2, we notice that the two initialization strategies exhibit similar performance, in terms of *Gen*. For instance, the clustering-based method can find out feasible solutions with less generation on 5 out of the 10 R-type instances, while the random-based method also exhibits more promising performance on the other 5 instances. The comparison results indicate that the clustering-based method cannot exhibit significantly better results since customers' geographical data in R-type instances are randomly generated. However, the clustering-based method is slightly better than the random-based method on 7 out of the 10 instances, measured by D_{short} and D_{avg}. The results indicate that the clustering-based method is not significantly superior to the random-based initialization method since the former method cannot extract useful knowledge from those randomly generated geographical data in R-type instances.

The comparison results on the RC-type instances are illustrated in Table 3. The table shows that the clustering-based method exhibits more reliable performance than the random-based method. Concretely, the clustering-based method can quickly find out feasible solutions on 7 out of the 10 RC-type instances. Meanwhile, it also dominates the random-based method on 8 instances, measured by D_{short} and D_{avg}. Comparing the results in the 3 tables, we can obtain a preliminary conclusion that the clustering-based initialization method applied in our study has a positive effect on VRPTW, in which customers' geographical data are (partially) clustered.

Table 1
Comparison results of initialization strategies on 10 C-type instances.

		C-10125	C-10150	C-20125	C-20150	C-101100	C-102100	C-103100	C-104100	C-105100	C-106100
Gen	*cluster*	**1**	**6.1**	**1**	**1**	**23.8**	**21.7**	**16.5**	11.6	**19.2**	**24.2**
	rnd	5.2	14.1	1.7	7.8	31.6	23.7	17.9	**11.2**	28.1	29.3
D_{short}	*cluster*	**439.70**	**897.27**	**363.28**	**837.83**	**1943.63**	**2073.67**	**2143.05**	2329.63	**1852.50**	**1862.91**
	rnd	454.89	998.96	506.32	1230.58	2226.01	2103.37	2198.09	**2294.80**	2306.94	2275.71
D_{avg}	*cluster*	**473.93**	**898.82**	**390.50**	**849.77**	**1981.31**	**2089.45**	**2174.38**	2359.94	**1913.48**	**1817.07**
	rnd	535.58	1035.47	671.38	1443.88	2251.31	2115.66	2218.11	**2348.25**	2322.31	2280.76

Table 2
Comparison results of initialization strategies on 10 R-type instances.

		R-10125	R-10150	R-20125	R-20150	R-101100	R-102100	R-103100	R-104100	R-105100	R-106100
Gen	*cluster*	9.3	30.6	**1**	**1**	254.8	**40.3**	31.5	23.8	**39.9**	**35.6**
	rnd	**8.4**	**27.7**	2.8	9.7	**237.8**	45.8	**31.4**	**21.7**	42.4	35.7
D_{short}	*cluster*	**802.59**	1344.07	**795.36**	**1353.01**	2147.93	**2021.56**	1903.17	**1655.49**	**2006.63**	**1843.56**
	rnd	804.45	**1325.17**	865.80	1679.49	**2125.88**	2067.77	**1898.82**	1791.11	2026.35	1859.58
D_{avg}	*cluster*	**810.14**	1348.56	**822.00**	**1372.09**	2158.05	**2021.56**	1907.89	**1669.15**	**2014.47**	**1843.80**
	rnd	813.08	**1325.51**	974.84	1856.71	**2130.26**	2067.77	**1898.82**	1825.51	2027.41	1859.58

Table 3
Comparison results of initialization strategies on 10 RC-type instances.

		RC-10125	RC-10150	RC-101100	RC-102100	RC-103100	RC-104100	RC-105100	RC-106100	RC-107100	RC-108100
Gen	*cluster*	**5.4**	27.1	46.1	**32.6**	**31.0**	**22.7**	**39.5**	37.4	**30.0**	**23.9**
	rnd	8.1	**22.9**	**43.3**	37.6	32.4	24.2	46.7	**37.3**	31.9	24.5
D_{short}	*cluster*	**839.64**	1377.49	**2408.72**	**2292.18**	**2081.31**	**2054.66**	**2277.04**	2271.18	**2191.93**	**2061.75**
	rnd	845.77	**1253.83**	2471.68	2354.01	2363.85	2064.28	2429.69	**2162.39**	2328.64	2306.12
D_{avg}	*cluster*	**857.09**	1379.62	**2410.65**	**2292.18**	**2081.31**	**2066.75**	**2295.99**	2279.90	**2191.93**	**2063.23**
	rnd	872.08	**1255.00**	2480.02	2360.81	2366.89	2080.05	2429.69	**2162.39**	2328.64	2309.17

In addition, to further testify distinct performance of the two clustering results on initialization, we utilize figures to illustrate the comparison results. Due to space limitations, in the following experiments, only C101100, C201100, R101100, R201100, RC101100, and RC201100 are adopted as test problems since they are very representative. The convergence processes of the hybrid method and the three primary initialization methods (i.e., the random-based, the distance-based, and the time windows-based initialization methods) are presented in Fig. 5. Note that, curves in the figure have two colors, i.e., black and red, which denote that the current best solutions are feasible and infeasible, respectively.

From Fig. 5 we notice that the hybrid initialization offers outstanding performance on the majority of the six instances. In C101100 and C201100 where geographical data are clustered, the hybrid initialization achieves the most favorable performance. Concretely, it has the shortest total distance in the early stage in the two instances. Moreover, applying the hybrid initialization attains feasible solutions earlier than the other three initialization strategies. An interesting phenomenon is that the time windows-based initialization offers an auspicious result on C101100 who has more compact time windows, while the distance-based initialization displays unfavorable performance on the instance. On the contrary, the distance-based initialization offers more promising performance, in terms of the total distance, than the time windows-based initialization in C201100, who has wider time windows.

In R101100 and R201100 instances where customers' positions are randomly generated, the time windows-based initialization displays more favorable performance than other strategies, in terms of total distance. Moreover, it can be seen that solutions obtained by the four initialization strategies are unfeasible. However, it does not mean that the proposed hybrid initialization strategy is not conducive to find feasible solutions. In fact, based on the solutions generated by the hybrid initialization strategy, the following crossover and mutation operators, as well as the local search in CLCS-GA can easily find feasible and optimal solutions. Moreover, it also can be observed that the time windows-based initialization can generate feasible solutions at the initial stage in R201100, while the other strategies cannot obtain feasible solutions until the fifteenth generation. On the contrary, the hybrid initialization offers competitive performance on the two instances, especially in R201100.

In the third type instances, i.e., RC101100 and RC201100, the hybrid initialization displays the most outstanding performance, while the distance-based initialization and the time windows-based initialization achieve very favorable performance in RC101100 and RC201100, respectively. Although the distance-based initialization yields a promising result in RC101100, it is dominated by the other two initialization strategies.

From the above discussions, we can obtain a few preliminary conclusions: (1) the hybrid initialization, which organically integrates information about customers' positions and time windows, can generate an excellent initial population, (2) the time windows-based initialization can offer very promising results on instances who have wider time windows and random (or semi-random) clustered geographical data, and (3) random-based initialization is not a good choice for VRPTW instances, though it is easily to perform.

5.2.2. Performance of the LCS-based crossover and mutation operators

In CLCS-GA, two types of LCSs based on elite individuals and inferior individuals are applied in crossover and mutation operators, respectively. Thus, in this part, the performance of the LCS-based operators is testified by the comparison results between it with the traditional crossover and mutation operators, which have been introduced in Fig. 2 and the first paragraph of Section 4.4, respectively. The results measured by the convergence process on the six VRPTW problems are illustrated in Fig. 6.

From Fig. 6 we notice that the traditional crossover and mutation operators display very unfavorable performance on the majority of the selected instances, measured by convergence speed and solution accuracy. On the contrary, LCS-based crossover together with LCS-based mutation yields more favorable performance. Furthermore, the hybrid strategy can help CLCS-GA obtain feasible solutions earlier than the traditional operators. And separately utilizing the LCS-based crossover

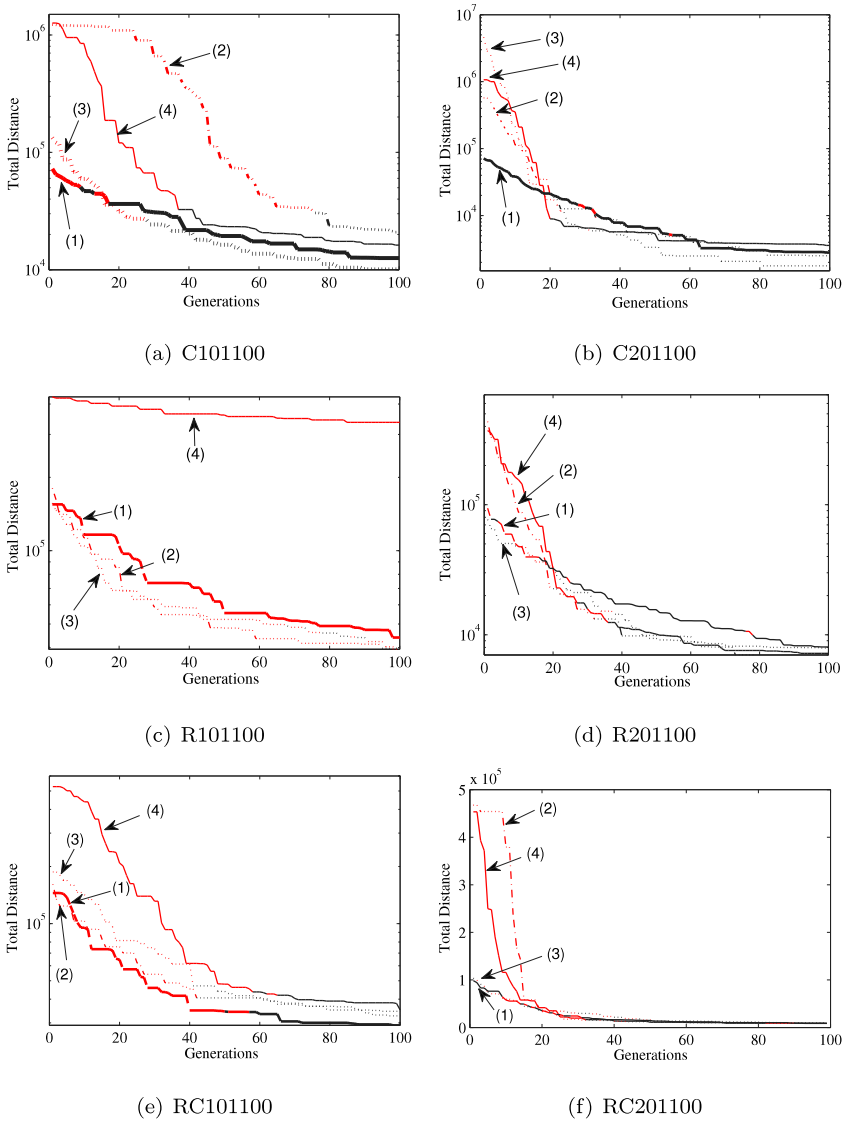


Fig. 5. Comparison results between hybrid initialization strategy and other 3 initialization strategies in 6 typical instances. Note that, numbers (1)–(4) in sub-figures denote hybrid, distance-based, time windows-based, and random-based initialization strategies, respectively.

or the LCS-based mutation exhibits a mediocre performance, compared with other strategies. We also notice an interesting phenomenon in the results of RC101100 that the LCS-based crossover operator yields the most unfavorable performance, in terms of the solutions' quality and the convergence speed. We think the reason may be that all the generated solutions are unfeasible, which causes the crossover operator to lose its correct guidance. The result indicates that feasible solutions contain more helpful and promising information. Moreover, in C201100 and R101100, LCS-based crossover offers slightly worse results than the traditional crossover and mutation operators. However, the LCS-based crossover displays more favorable performance along with the evolution process.

From all the results, we can say that adequately utilizing the LCS of different individuals is a promising method to improve the performance of CLCS-GA, though some characteristics of it need to be further studied.

5.2.3. *Performance of relatedness-based removing strategy*

In CLCS-GA, a relatedness-based removing strategy is applied in the local search operator. The aim of the strategy is to overcome the blindness of traditional local search methods and then improve the search efficiency of LCLS-GA. In this part, the performance of the relatedness removing strategy is verified by the comparison between it with the random-based removing strategy, in which a few customers are randomly removed from a solution and then to perform the local search. The comparison results in the six instances are presented in Fig. 7.

From Fig. 7 we notice that the relatedness-based removing strategy significantly dominates the random-based removing strategy on four test instances (i.e., C201100, R101100, R201100, and RC201100), measured by convergency speed and solution accuracy. On RC101100, the relatedness-based removing strategy is slightly better than the random-based removing strategy. Concretely, both strategies can firstly find a feasible solution after 50 generations. However, the feasible solution obtained by the relatedness-based removing strategy has a shorter distance than obtained by the random-based removing strategy. These experimental results verify that the relatedness-based removing strategy in CLCS-GA can offer reliable and promising results on different types of instances. In other words, when performing the local search, considering relatedness between different customers can overcome some shortcomings caused by the random-based removing strategy, and then enhance the efficiency of the local search.

In [39], two popular local search methods are introduce. The one is LNS-GENI, in which the customers in the longest detour have a

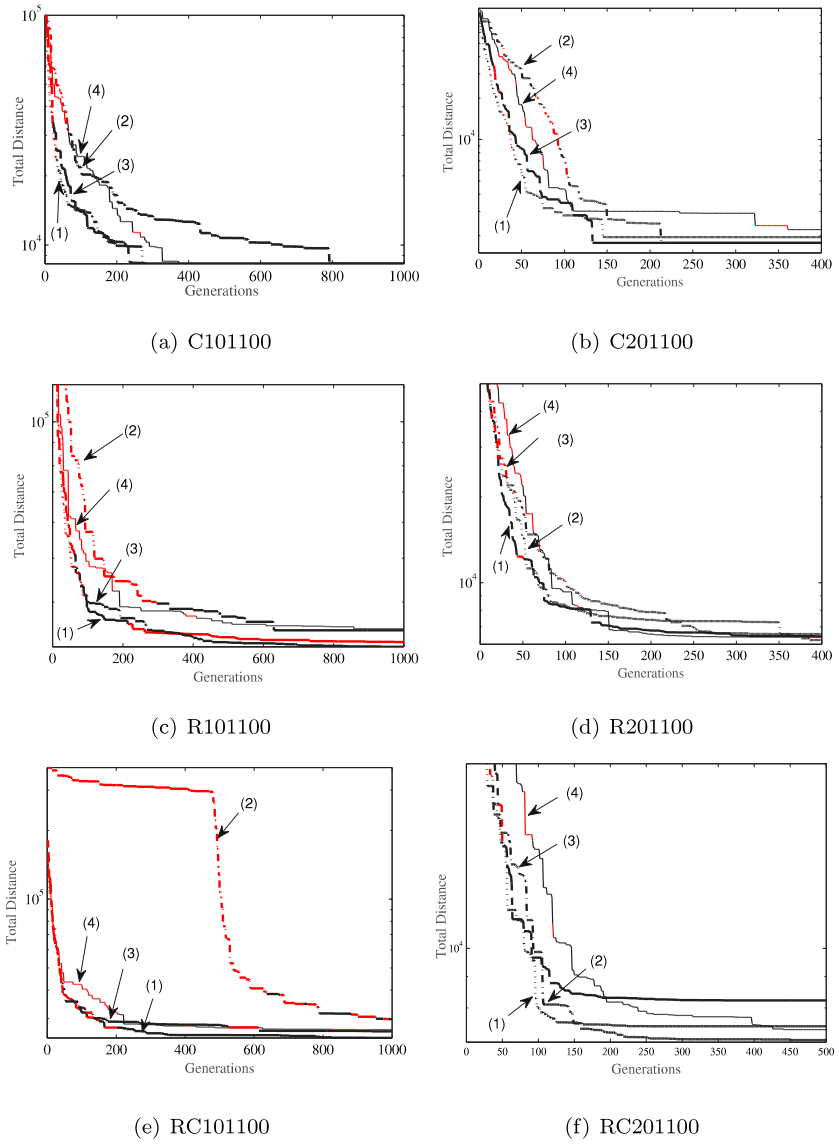


Fig. 6. Comparison results of different crossover and mutation operators in 6 typical instances. Note that, numbers (1)–(4) in sub-figures denote LCS-based crossover and mutation, LCS-based crossover, LCS-based mutation, and traditional crossover and mutation, respectively.

higher probability to be removed in the detour. The motivation is that removing a customer far away from the rest of the route will create temporal space in that route and thus facilitate future insertions. The main objective of the selection strategy is to reduce the total travel distance as it tends to replace long arcs with shorter ones. The other one is Small routing operator (SMART), which is suitable for solving a smaller VRP. Instead of removing customers, SMART removes arcs from a solution, thus creating an incomplete one to the original problem. Then the incomplete solution could be interpreted as a smaller VRP, which can be easier optimized.

In [39], the SMART and LNS-GENI operators presented above are used in a local descent strategy, as they never allow the objective function to increase. However, such strategies may inevitably get trapped in local minimum when no better solution is found in a neighborhood. Hence, the variable neighborhood descent (VND) is introduced to overcome such a problem. Based on the VND strategy, if the one operator is confirmed to be in a local minimum, the second operator can be started until it can no longer improve the solution. Similar to hybrid strategies, the VND strategy can take advantages merits of the SMART and LNS-GENI operators.

Table 4
Comparison results between the relatedness-based strategy with other 3 strategies on different types of instances.

Type	LNS-GENI		SMART		VND		CLCS-GA	
	Distance	Vehicle	Distance	Vehicle	Distance	Vehicle	Distance	Vehicle
R1	1276.39	12.17	1245.41	12.17	1225.19	**12.08**	**1204.30**	12.58
R1	999.43	3.09	971.17	3.09	954.07	**3.00**	**933.32**	3.64
C1	977.60	10.00	860.11	10.00	834.30	10.00	**824.43**	**9.89**
C2	697.88	**3.00**	599.78	**3.00**	591.06	**3.00**	**589.86**	3.00
RC1	1438.79	11.75	1426.80	11.63	1401.76	**11.63**	**1352.53**	12.63
RC2	1232.18	**3.38**	1156.42	**3.38**	1124.46	**3.38**	**1081.09**	4.00

To verify the local strategy proposed in this study, comparison results between it with the LNS-GENI, SMART, and VND are presented in Table 4.

From Table 4 we can see that, VDN dominates LNS-GENI and SMART on all the 6 types of instances, measured by distance and number of vehicles. Compared with VND, CLCS-GA offers the most outstanding performance on all the 6 types of instances, in terms of distance. In addition, it also yields the best results on C1 and C2 types,

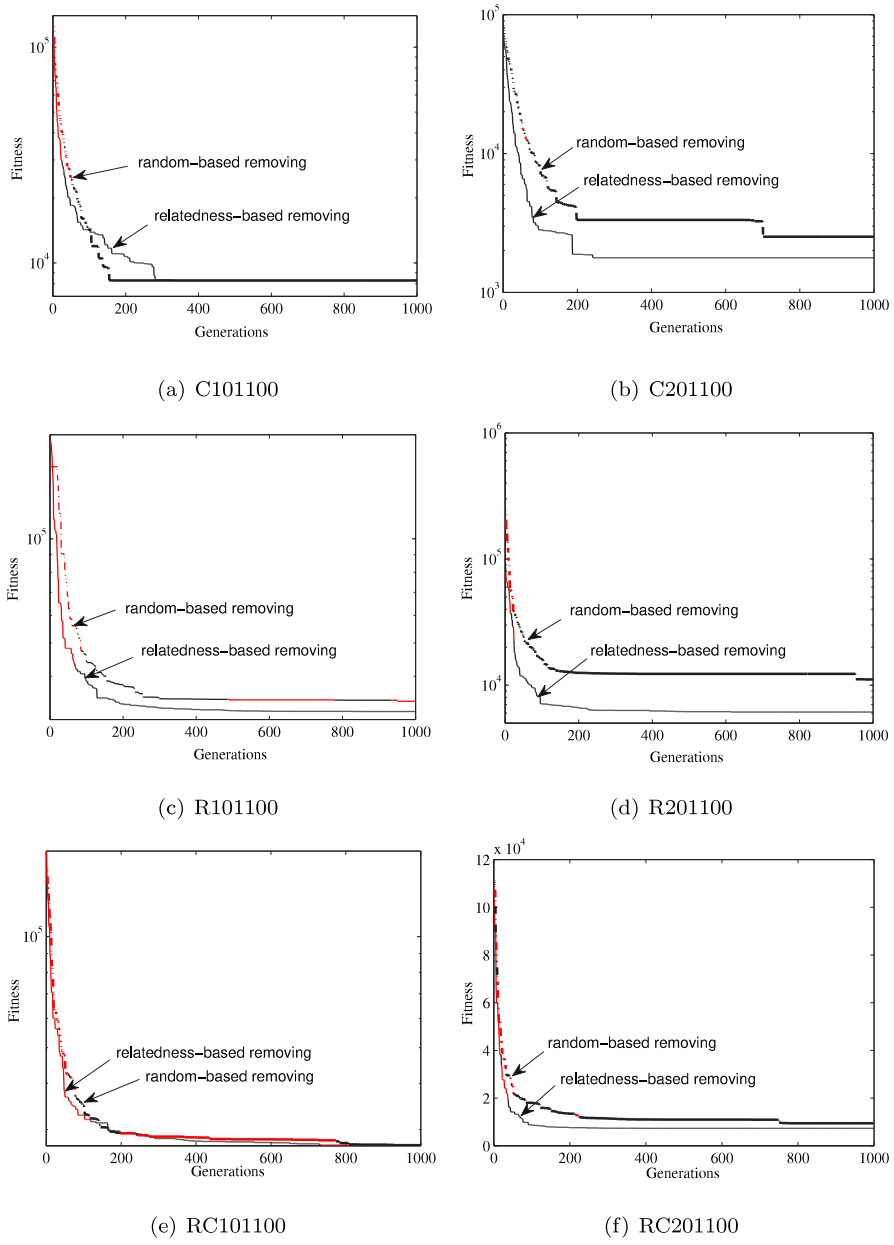


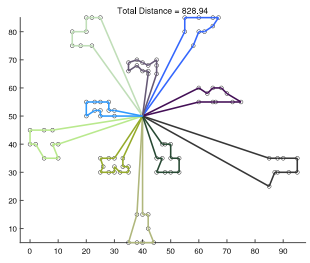
Fig. 7. Comparison results between the random-based removing strategy and the relatedness-based removing strategy in 6 typical instances.

measured by the number of vehicles. Thus, we regard that the favorable performance of the clustering-based initialization method is attributed to the fact that customers' geographical data being clustered in the selected C-type instances. Moreover, we notice that CLCS-GA is slightly inferior to other competitors. Thus, we will further improve CLCS-GA in our future work.

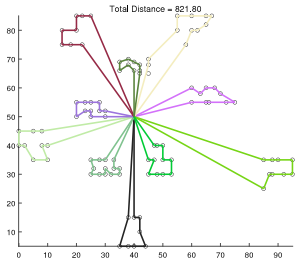
5.3. Comparison results between CLCS-GA and other peer algorithms

Extensive experiments are executed to test the comprehensive performance of CLCS-GA in this section. In the experiments, seven algorithms, including 3 heuristic algorithms (MOLNS [46], Tabu-ABC [47], and ACO-N [48]) and 4 metaheuristics algorithms (RRGA [49], MS-EAS [50], MOTPS [51], and ScPSO [52]), are selected as competitors for CLCS-GA. Note that the parameters of the selected seven competitors are similar to that those introduced in the corresponding literature. Experimental results of the 3 types of instances (R-type, C-type, and RC-type) are presented in Table 5, Table 6, and Table 7, respectively. Note that, the experimental results include two indicators: the best obtained result (*Best*) consisting of distance and the number of vehicles, and corresponding sort value (*Sort*). Generally, the number of vehicles should be considered with high priority for the VRPTW problems. In other words, a feasible solution with fewer vehicles is better than a feasible solution with more vehicles. If two feasible solutions have the same number of vehicles, the solution with a shorter distance is superior to the solution with a longer distance.

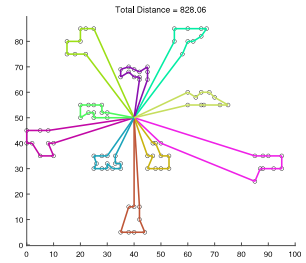
Note that the best results listed in the tables are highlighted with bold. Furthermore, the number of obtained best results ($Sum.Best$) by each algorithm and the average value of sort values ($Avg.Sort$) are also presented in the bottom of the tables. For example, in the top-left cell of Table 5, "1655.03(19)" denotes that, in the best result obtained by ACO-N, the number of vehicles is 19 and the corresponding distance is 1655.03, while "2" means the sort value of the result (the lower the better) is the 2nd. In the bottom-left cell of Table 5, "3" represents that ACO-N attains the best result in 3 instances, and "4.70" denotes the average value of $Sort$ of ACO-N.



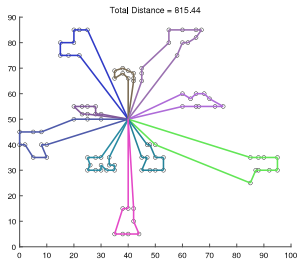
(a) the known best result on C102100



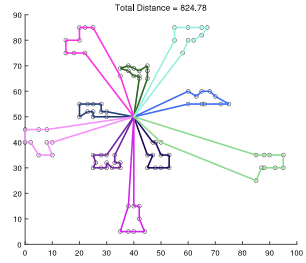
(b) the best solution obtained by CLCS-GA on C102100



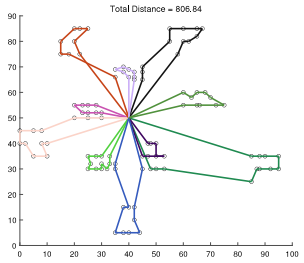
(c) the known best result on C103100



(d) the best solution obtained by CLCS-GA on C103100

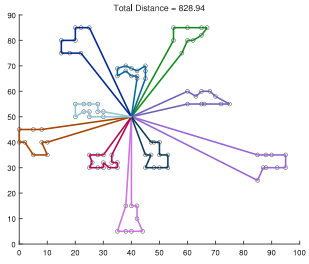


(e) the known best result on C104100

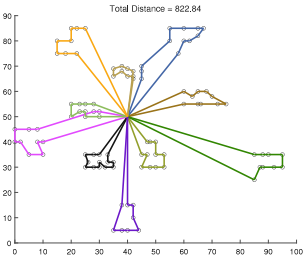


(f) the best solution obtained by CLCS-GA on C104100

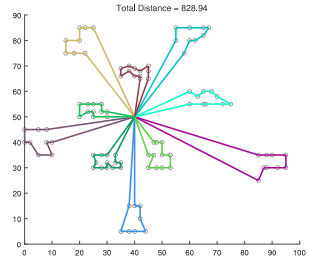
Fig. 8(a). The best solutions obtained by CLCS-GA and the known best solutions in C102100, C103100, and C104100.



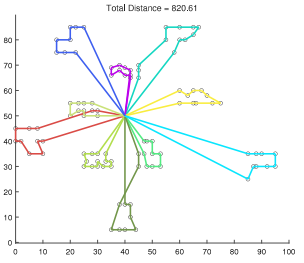
(a) the known best result on C105100



(b) the best solution obtained by CLCS-GA on C105100

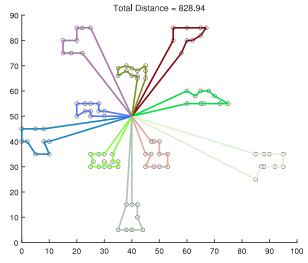


(c) the known best result on C107100

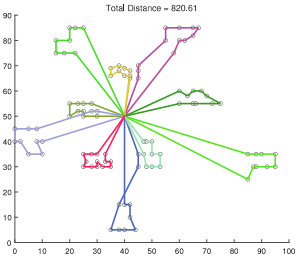


(d) the best solution obtained by CLCS-GA on C107100

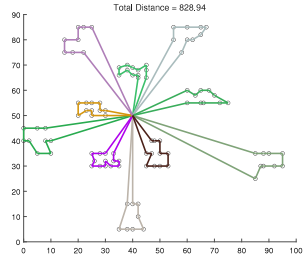
Fig. 8(b). The best solutions obtained by CLCS-GA and the known best solutions in C105100 and C107100.



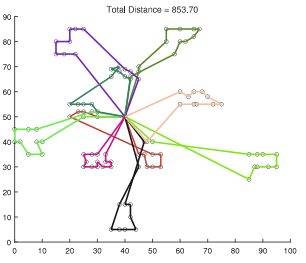
(a) the known best result on C108100



(b) the best solution obtained by CLCS-GA on C108100



(c) the known best result on C109100



(d) the best solution obtained by CLCS-GA on C109100

Fig. 8(c). The best solutions obtained by CLCS-GA and the known best solutions in C108100 and C109100.

5.3.1. Performance in R-type instances

In the R-type instances, where geographical data are randomly generated, MOTPS offers the best results on 9 out of the 23 instances, followed by CLCS-GA and MOLNS who attain the most favorable results on 5 and 4 instances, respectively. Moreover, MOTPS and CLCS-GA also display very stable performance in R-type instances since the two peer algorithms achieve better results, measured by "$Avg.Sort$". Comparing the two favorable algorithms, i.e., MOTPS and CLCS-GA, we can observe that MOTPS achieves better solutions measured by the number of vehicles than CLCS-GA on 7 instances, while CLCS-GA dominates MOTPS on 3 instances. When the two algorithms offer the same performance in terms of the number of vehicles, CLCS-GA attains a shorter distance than MOTPS. Thus, from the results, we can say that MOTPS and CLCS-GA are biased to optimize number of vehicles and total distance in the type instances, respectively. On the contrary, Tabu-ABC and MS-EAS cannot yield any favorable results on R-type instances. From the comparison results, we notice that great disparity exists among the eight peer algorithms, in terms of both the number of vehicles and the total distance. In other words, there are more optimal solutions in VRPTW if customers are randomly distributed.

5.3.2. Performance in C-type instances

In the C-type instances, where geographical data are clustered, all the peer algorithms display very promising performance. Obviously, the clustered geographical positions enable VRPTW to be easily dealt with. For instance, the majority of the peer algorithms display the same performance in C101100, C106100, C201100, C202100, C206100, and C207100. Although CLCS-GA cannot display the most favorable performance in the R-type instances, it attains the most outstanding result because it yields the best results in all the instances except C208100, where CLCS-GA is slightly worsen than ACO-N and MOTPS. In addition, CLCS-GA also displays the most stable performance because it

Table 5
Solution accuracy in R-type instances.

		ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
*R*101100	*Best*	1655.03(19)	1643.18(20)	1642.88(20)	**1654.93(19)**	2022.12(31)	1653.10(20)	1655.87(20)	1656.20(19)
	Sort	2	5	4	1	8	6	7	3
*R*102100	*Best*	1491.18(18)	1460.26(18)	1473.62(18)	1475.33(18)	1923.30(25)	**1480.70(17)**	1474.75(18)	1499.70(17)
	Sort	7	3	4	6	8	1	5	2
*R*103100	*Best*	1243.22(14)	1217.39(15)	1213.62(14)	1240.44(14)	1553.54(19)	1236.10(14)	1227.27(14)	**1258.34(13)**
	Sort	6	7	2	5	8	4	3	1
*R*104100	*Best*	982.01(10)	987.61(11)	976.61(11)	1010.72(10)	1271.94(13)	998.70(10)	**852.30(10)**	994.92(10)
	Mean	2	7	6	5	8	4	1	3
*R*105100	*Best*	1380.44(16)	1363.91(15)	1360.78(15)	1389.85(15)	1777.58(22)	**1369.30**(14)	1375.74(15)	1378.20(15)
	Sort	7	3	2	6	8	1	4	5
*R*106100	*Best*	1265.36(13)	1247.90(13)	1239.37(13)	1269.14(13)	1579.65(18)	1267.60(13)	1259.02(14)	**1332.60(12)**
	Sort	4	3	2	6	8	5	7	1
*R*107100	*Best*	1100.25(11)	1087.50(12)	**1072.12(11)**	1102.72(11)	1293.31(15)	1086.00(11)	1094.74(11)	1081.62(11)
	Sort	5	7	1	6	8	3	4	2
*R*108100	*Best*	**960.88**(9)	961.85(11)	951.22(10)	991.57(10)	1189.34(12)	960.20(10)	978.49(11)	958.04(10)
	Sort	1	6	2	5	8	4	7	3
*R*109100	*Best*	**1101.99**(12)	1152.99(13)	1151.84(13)	1177.76(12)	1459.08(17)	1156.80(12)	1185.77(12)	1156.62(12)
	Sort	1	7	6	4	8	3	5	2
*R*110100	*Best*	1119.53(12)	1091.50(12)	1072.42(12)	1129.60(12)	1262.72(14)	1096.70(11)	1108.15(12)	**1084.72**(11)
	Sort	6	4	3	7	8	2	5	1
*R*111100	*Best*	1091.11(12)	1067.46(12)	1053.50(12)	1108.70(12)	1323.24(15)	**1071.20**(11)	1071.86(11)	1085.32(11)
	Sort	6	5	4	7	8	1	2	3
*R*112100	*Best*	974.73(10)	973.25(10)	**959.97**(10)	964.15(10)	1148.53(12)	972.80(10)	1001.93(11)	965.37(10)
	Sort	6	5	1	2	8	4	7	3
*R*201100	*Best*	1214.22(7)	1174.69(6)	1147.8(8)	1305.25(4)	1537.09(17)	**1228.70(4)**	1168.07(7)	1224.20(5)
	Sort	6	4	7	2	8	1	5	3
*R*202100	*Best*	1105.20(5)	1046.10(5)	1034.35(8)	**1093.67(4)**	1537.09(12)	1153.00(4)	1043.68(6)	1096.70(4)
	Sort	5	4	7	1	8	3	6	2
*R*203100	*Best*	960.14(4)	884.02(5)	874.84(6)	915.43(4)	1177.65(8)	**979.90(3)**	882.44(5)	916.06(4)
	Sort	4	6	7	2	8	1	5	3
*R*204100	*Best*	771.47(4)	750.40(4)	735.80(5)	775.99(3)	969.51(5)	801.60(3)	746.47(5)	**757.79**(3)
	Sort	5	4	6	2	8	3	7	1
*R*205100	*Best*	1050.26(4)	960.75(5)	954.16(5)	1075.10(3)	1330.71(10)	**1018.80(3)**	962.98(5)	974.56(4)
	Sort	4	6	5	2	8	1	7	3
*R*206100	*Best*	954.85(4)	900.97(4)	884.85(5)	**979.21(3)**	1277.98(7)	963.90(3)	892.48(5)	1080.90(3)
	Sort	5	4	6	1	8	2	7	3
*R*207100	*Best*	870.33(3)	809.72(4)	797.99(4)	851.89(3)	1150.09(6)	875.80(3)	817.11(5)	**818.97**(3)
	Sort	3	6	5	2	8	4	7	1
*R*208100	*Best*	777.72(3)	723.14(5)	705.33(4)	**754.99(2)**	1046.26(3)	758.80(2)	710.30(4)	733.89(3)
	Sort	4	8	6	1	5	2	7	3
*R*209100	*Best*	934.21(3)	863.12(5)	860.46(5)	898.23(4)	1296.69(7)	**920.70(3)**	871.40(5)	877.15(4)
	Sort	2	7	6	5	9	1	8	4
*R*210100	*Best*	949.02(5)	927.54(5)	904.78(6)	941.58(4)	1192.05(9)	**970.90(3)**	921.42(5)	937.29(4)
	Sort	6	5	7	3	8	1	4	2
*R*211100	*Best*	877.55(4)	763.22(4)	764.69(4)	838.14(3)	1085.51(5)	**814.60(3)**	782.63(5)	849.02(3)
	Sort	6	4	5	2	8	1	7	3
Sum.	*Best*	2	0	2	4	0	9	1	5
Avg.	*Sort*	4.52	5.22	4.52	3.61	7.91	2.70	5.52	2.48

has the smallest "$Avg.Sort$". On the contrary, MOTPS offers the most unfavorable performance in C-type instances, though it yields the most comprehensive performance in R-type instances. Moreover, the best results in C102100, C103100, C104100, C105100, C107100, C108100, and C109100 obtained by CLCS-GA are better than the known best results.[1] In addition, details of solutions obtained by CLCS-GA and the known best results are compared in Figs. 8(a)–8(c). From the figure we notice a few interesting phenomena. In C104100, for instance, although some paths crossings appeared in the solution obtained by CLCS-GA, the solution has a shorter distance than the known best solution with no path crossing. The same phenomenon also can be seen from the results in C105100, C107100, and C107100.

[1] see: https://www.sintef.no/projectweb/top/vrptw/100-customers/.

The experimental results in this part indicate that the C-type instances are easier than R-type instances benefiting from the clustered geographical data. Moreover, taking advantages of the clustering of geographical data and time-windows helps CLCS-GA find more favorable results than the known best results in some instances.

The comparison results presented in Table 6 indicate that the location-based sub-population and the time windows-based sub-population introduced in CLCS-GA significantly improve its performance on C-type instances. The results verify that efficiently utilizing customers' locations and time windows is feasible for VRPTW.

5.3.3. Performance in RC-type instances

In the third type instances, i.e., RC-type instances, the geographical data of customers include random and clustered structures, which are more approximate to the realistic situation. The experimental results

Table 6
Solution accuracy in C-type instances.

		ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
*C*101100	*Best*	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**
	Sort	1	1	1	1	1	1	1	1
*C*102100	*Best*	828.94(10)	828.94(10)	828.94(10)	828.94(10)	834.53(10)	828.94(10)	828.94(10)	**821.93(10)**
	Sort	2	2	2	2	2	2	2	1
*C*103100	*Best*	828.07(10)	828.07(10)	828.07(10)	828.94(10)	863.04(10)	899.30(10)	828.07(10)	**815.44(10)**
	Sort	2	2	2	6	7	8	2	1
*C*104100	*Best*	828.20(10)	824.78(10)	824.78(10)	828.94(10)	893.23(10)	913.20(10)	852.30(10)	**806.84(10)**
	Sort	4	2	2	5	7	8	6	1
*C*105100	*Best*	828.90(10)	828.94(10)	828.94(10)	828.94(10)	828.94(10)	828.94(10)	828.94(10)	**822.84(10)**
	Sort	2	2	2	2	2	2	2	1
*C*106100	*Best*	**828.94(10)**	**828.94(10)**	**828.94(10)**	**828.94(10)**	830.33(10)	**828.94(10)**	**828.94(10)**	**828.94(10)**
	Sort	1	1	1	1	9	1	1	1
*C*107100	*Best*	828.94(10)	828.94(10)	828.94(10)	828.94(10)	828.94(10)	829.61(10)	828.94(10)	**820.61(10)**
	Sort	2	2	2	2	2	8	2	1
*C*108100	*Best*	830.94(10)	828.94(10)	828.94(10)	828.94(10)	837.06(10)	828.94(10)	828.94(10)	**820.61(10)**
	Sort	2	2	2	2	8	2	2	1
*C*109100	*Best*	829.22(10)	828.94(10)	828.94(10)	828.94(10)	833.61(10)	835.60(10)	854.78(10)	**853.70(9)**
	Sort	5	2	2	2	6	7	8	1
*C*201100	*Best*	**591.56(3)**	**591.56(3)**	**591.56(3)**	**591.56(3)**	643.15(4)	**591.56(3)**	**591.56(3)**	**591.56(3)**
	Sort	1	1	1	1	8	1	1	1
*C*202100	*Best*	**591.56(3)**	**591.56(3)**	**591.56(3)**	**591.56(3)**	627.84(4)	**591.56(3)**	**591.56(3)**	**591.56(3)**
	Sort	1	1	1	1	8	1	1	1
*C*203100	*Best*	593.25(3)	**591.17(3)**	**591.17(3)**	591.56(3)	711.99(4)	591.56(3)	**591.17(3)**	**591.17(3)**
	Sort	7	1	1	5	8	5	1	1
*C*204100	*Best*	595.55(3)	594.89(3)	**590.60(3)**	**590.60(3)**	727.22(4)	630.50(3)	596.55(3)	**590.60(3)**
	Sort	5	4	1	1	8	7	6	1
*C*205100	*Best*	**588.88(3)**	**588.88(3)**	**588.88(3)**	**588.88(3)**	592.90(3)	617.30(3)	**588.88(3)**	**588.88(3)**
	Sort	1	1	1	1	7	8	1	1
*C*206100	*Best*	**588.49(3)**	**588.49(3)**	**588.49(3)**	**588.49(3)**	600.08(3)	**588.49(3)**	**588.49(3)**	**588.49(3)**
	Sort	1	1	1	1	8	1	1	1
*C*207100	*Best*	**588.88(3)**	**588.29(3)**	**588.29(3)**	**588.29(3)**	618.74(3)	**588.29(3)**	**588.29(3)**	**588.29(3)**
	Sort	1	1	1	1	8	1	1	1
*C*208100	*Best*	**588.03(3)**	588.32(3)	588.32(3)	588.32(3)	635.21(3)	588.29(3)	588.32(3)	588.32(3)
	Sort	1	3	3	3	8	2	3	3
Sum.	*Best*	8	8	9	8	1	6	8	16
Avg.	*Sort*	2.29	1.71	1.53	2.17	5.12	3.82	2.41	1.12

demonstrated in Table 7 indicate that CLCS-GA and MOLNS yield the best performance since both of them attain the best results on 6 out of the 16 instances, followed by ACO-N and RRGA, who attain the best results on 2 instances. On the contrary, the other 4 algorithms, i.e., Tabu-ABC, MS-EAS, MOTPS, and ScPSO, cannot exhibit the best performance on any instances. Comparing the results of CLCS-GA and MOLNS, who are two outstanding algorithms in this experiment, we notice that the best results obtained by MOLNS are all RC2-type instances, which have larger vehicle capacities and longer dispatch cycles. On the contrary, the best results obtained by CLCS-GA include 4 RC1-type instances, which have smaller vehicle capacities and more compact time windows, and 2 RC2-type instances. Thus, we regard that utilizing characteristics of customers' time windows helps CLCS-GA efficiently deal with a VRPTW with more compact time windows. In other words, CLCS-GA can provide highly efficient solutions for those VRPTW instances with compact time windows, which have more value in many real applications, such as emergency rescue and delivery. Furthermore, compared with MOLNS, CLCS-GA shows the most favorable performance, in terms of the value of "$Avg.Sort$". The result verifies that CLCS-GA has more stable performance than MOLNS.

5.3.4. Overall performance of all the peer algorithms

Based on the comparison results in the three different types of instances, the overall performance of all the peer algorithms can be summarized in Table 8. The results indicate that CLCS-GA attains the best results in 27 out of the 56 instances, followed by MOLNS and MOTPS. Moreover, CLCS-GA also displays the most stable performance on the instances since the average sort values is 1.97. From the results, we can see that CLCS-GA dominates the other seven competitors in the selected VRPTW instances.

In addition, to display the overall performance of the peer algorithms on the 6 distinct types of instances (i.e., R1, R2, C1, C2, RC1, and RC2), average results of obtained best results, measured by the total distance (TD) and the number of vehicles (NV), are presented in Table 9. From the results, we can see that CLCS-GA yields the most comprehensive and outstanding performance in R1, C1, and C2 types. Moreover, it also displays very favorable performance in R2 and RC1 types, measured by TD and NV, respectively. Although CLCS-GA cannot offer the performance in RC2, in which RRGA attains the shortest distance, CLCS-GA can find the second shortest distance by less NV. Conversely, CLCS-GA can find a shorter distance than MOLNS, which yields the best performance in terms of NV, by slightly greater NV, i.e., 4:3.75. In other words, CLC-GA can offer a favorable balance between TD and NV on RC1 and RC2, which is more approximate to the realistic situation.

5.3.5. Statistics analysis

In this part, two statistical tests, i.e., the Wilcoxon test and the Friedman test, are conducted based on the experimental results presented in Table 9. Note that there is a significant difference between two algorithms only when the value of p is less than 0.05.

From the results of the Wilcoxon test, which are included in Table 10, we can see that CLCS-GA is significantly better than MOLNS,

Table 7
Solution accuracy in RC-type instances.

		ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
*RC*101100	*Best*	**1650.14(14)**	1646.17(16)	1623.59(15)	1662.56(15)	2228.64(22)	1664.00(15)	1665.27(16)	1656.30(15)
	Sort	1	6	2	4	8	5	7	3
*RC*102100	*Best*	**1514.85(13)**	1481.61(14)	1461.23(14)	1486.35(14)	1987.83(21)	1489.70(14)	1493.58(14)	1471.77(14)
	Sort	1	4	2	5	8	6	7	3
*RC*103100	*Best*	1277.11(11)	1280.76(12)	**1262.98(11)**	1291.95(12)	1616.17(15)	1295.60(12)	1305.40(12)	1278.25(12)
	Sort	2	4	1	5	8	6	7	3
*RC*104100	*Best*	1159.37(10)	1162.03(11)	1135.48(10)	1162.53(10)	1328.29(12)	1171.10(11)	1188.32(11)	**1117.50(10)**
	Sort	3	5	2	4	8	6	7	1
*RC*105100	*Best*	1617.88(15)	1545.30(16)	1518.58(16)	1604.53(15)	1976.94(21)	1558.60(15)	1553.15(15)	**1552.70(15)**
	Sort	5	7	6	4	8	3	2	1
*RC*106100	*Best*	1387.63(13)	1401.17(14)	**1377.35(13)**	1400.09(13)	1636.73(16)	1387.30(13)	1427.71(14)	1390.85(13)
	Sort	3	6	1	5	8	2	7	4
*RC*107100	*Best*	1280.01(11)	1235.28(12)	1212.83(12)	1259.55(12)	1478.84(14)	1265.7(12)	1280.45(12)	**1231.20(11)**
	Sort	2	4	3	5	8	6	7	1
*RC*108100	*Best*	1157.44(11)	1136.35(11)	1140.73(11)	1205.13(11)	1303.07(12)	1152.50(11)	1210.74(11)	**1121.70(11)**
	Sort	5	2	3	6	8	4	7	1
*RC*201100	*Best*	1279.65(5)	1271.78(7)	1266.11(9)	**1497.89(4)**	1962.33(16)	1392.90(5)	1286.62(8)	1330.10(5)
	Sort	2	5	7	1	8	4	6	3
*RC*202100	*Best*	1157.02(5)	1116.21(6)	1095.64(8)	**1199.53(4)**	1521.47(13)	1238.90(4)	1116.79(7)	1209.40(4)
	Sort	4	5	7	1	8	3	6	2
*RC*203100	*Best*	1046.33(6)	941.81(5)	926.82(5)	985.54(4)	1264.77(8)	1044.50(4)	945.76(5)	**960.43(4)**
	Sort	7	5	4	2	8	3	6	1
*RC*204100	*Best*	847.33(4)	801.87(4)	788.66(4)	**805.46(3)**	1007.49(4)	863.20(3)	791.40(4)	844.62(3)
	Sort	7	6	4	1	8	3	5	2
*RC*205100	*Best*	1334.55(5)	1165.82(7)	1157.55(7)	1340.38(5)	1630.07(15)	1278.10(5)	1159.42(7)	**1252.90(5)**
	Sort	3	7	5	4	8	2	6	1
*RC*206100	*Best*	1112.20(5)	1072.85(5)	1054.61(7)	**1316.42(3)**	1523.88(11)	1120.70(4)	1063.57(6)	1120.70(4)
	Sort	5	4	7	1	8	2	6	2
*RC*207100	*Best*	1078.52(5)	977.11(5)	966.08(6)	**1031.62(4)**	1523.88(8)	1051.20(4)	966.08(6)	1053.90(4)
	Sort	5	4	6	1	8	2	6	3
*RC*208100	*Best*	911.15(3)	792.33(5)	779.31(4)	**859.13(3)**	1137.74(5)	859.70(3)	780.72(4)	876.68(3)
	Sort	4	7	5	1	8	2	6	3
Sum.	*Best*	2	0	2	6	0	0	0	6
Avg.	*Sort*	3.69	5.06	4.06	3.13	8.00	3.69	6.13	2.13

Table 8
Overall performance measured by the number of achieved best results (*Sum.Best*) and average sort values (*Avg.Sort*).

		ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
Sum.	*Best*	12	8	13	18	1	15	9	27
Avg.	*Sort*	3.61	4.11	3.48	3.04	7.09	3.32	4.75	1.97

MS-EAS, and MOTPS measured by TD. According to the values of p, we also notice that CLCS-GA displays similar performance to ACO-N, Tabu-ABC, RRGA, and ScPSO measured by TD. When comparing NV, the statistical results indicate that CLCS-GA is significantly better than 4 algorithms (i.e., Tabu-ABC, RRGA, MS-EAS, and ScPSO), while there is no significantly different between it with other 3 algorithms. When considering both TD and NV, CLCS-GA significantly surpasses ACO-N, MOLNS, and ME-EAS. From the Wilcoxon test results, we notice that CLCS-GA yields more favorable performance than the other seven peer algorithms because it has not been dominated by any algorithm, in terms of the values of p.

The results of the Friedman test presented in Table 11 show that CLCS-GA and RRGA attain the most outstanding performance, measured by ranking values on NV and TD, respectively. Note that CLCS-GA also obtains the second-best result in terms of TD, which is slightly worse than RRGA. On the contrary, RRGA only yields the fifth-best result measured by NV though it displays the best performance measured by TD. When considering both NV and TD, we observe that CLCS-GA offers the best performance, followed by RRGA and MOLNS.

Together with the results of the two statistics tests, we can see that CLCS-GA exhibits most comprehensive and reliable performance than the other seven competitors in the selected VRPTW instances.

6. Conclusions

In this paper, a GA-based algorithm, named CLCS-GA, is used to optimize VRPTW. Considering that geographic distribution and time windows distribution of different customers may clustered together in real life, we first utilize the K-means algorithm to cluster the customers' positions and time windows. According to the two cluster results, two distinct initialization strategies are proposed. Moreover, a random initialization strategy, widely accepted in GA, is also utilized in this study. Based on the three initialization strategies, three sub-populations are generated at the beginning of optimization. Furthermore, to protect some promising gene blocks and destroy some unpromising gene blocks during the evolution process, we extract two types of longest common substring (LCS) from the elite individuals and inferior individuals, respectively. Based on the two kinds of LCSs, a novel crossover operator and a mutation operator are proposed to enhance the exploitation ability and exploration, respectively. Lastly, when performing the local search, relatedness between customers is considered, aiming to overcome the blindness of the common local search.

Extensive experiments manifested that CLCS-GA has promising and reliable performance on different types of VRPTW instances. From the results, we can obtain some preliminary conclusions: (1) Organically integrating information of customers' positions and time windows can generate a promising initial population, and then beneficial for its

Table 9
Average results of obtained best results, measured by the total distance (TD) and the number of vehicles (NV), on 6 different types of instances.

		ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
	TD	1197.14	1187.90	1180.66	1209.58	1483.70	1195.78	1190.49	**1204.30**
*R*1	*NV*	13.00	13.50	13.25	13.00	17.75	12.75	13.25	**12.58**
*R*2	*TD*	951.36	891.24	878.64	948.13	1236.42	953.34	890.82	**843.59**
	NV	4.18	4.72	5.45	3.36	8.09	**3.09**	5.18	3.64
*C*1	*TD*	829.01	828.38	828.38	828.94	842.07	846.93	834.31	**824.43**
	NV	10	10	10	10	10	10	10	**9.89**
*C*2	*TD*	590.78	590.40	**589.86**	589.91	644.64	598.44	590.60	**589.86**
	NV	**3**	**3**	**3**	**3**	3.50	**3**	**3**	**3**
*RC*1	*TD*	1380.55	1361.08	**1341.60**	1384.09	1694.56	1373.06	1390.58	1352.53
	NV	12.25	13.25	12.75	12.75	16.63	12.88	13.13	**12.63**
*RC*2	*TD*	1095.84	1017.47	**1006.87**	1129.50	1446.45	1106.15	1013.80	1081.09
	NV	4.75	5.50	6.25	**3.75**	10.00	4.00	5.88	4.00

Table 10
Wilcoxon results between CLCS-GA and other seven algorithms.

	CLCS-GA vs.	ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO
	R^+	5	4	2	6	6	5	4
TD	R^+	1	2	3	0	0	1	2
	p	0.116(≈)	0.917(≈)	0.500(≈)	**0.028**(+)	**0.028**(+)	**0.046**(+)	0.753(≈)
NV	R^+	4	5	5	3	6	3	5
	R^+	1	0	0	2	0	1	0
	p	0.138(≈)	**0.043**(+)	**0.043**(+)	0.863(≈)	**0.028**(+)	0.715(≈)	**0.043**(+)
TD & *NV*	R^+	9	9	7	9	12	8	9
	R^+	2	2	3	2	0	2	2
	p	**0.041**(+)	0.248(≈)	0.799(≈)	**0.033**(+)	**0.002**(+)	0.059(≈)	0.213(≈)

Table 11
Friedman results of all the eight algorithms.

	ACO-N	Tabu-ABC	RRGA	MOLNS	MS-EAS	MOTPS	ScPSO	CLCS-GA
TD	5.33(5)	3.08(3)	**1.50**(1)	5.33(5)	7.83(8)	6.00(7)	4.33(4)	2.58(2)
NV	3.58(4)	5.50(7)	5.33(5)	3.17(2)	7.50(8)	3.25(3)	5.42(6)	**2.25**(1)
TD & *NV*	4.46(5)	4.29(4)	3.42(2)	4.25(3)	7.67(8)	4.63(6)	4.88(7)	**2.42**(1)

subsequent evolution; (2) Saving some common genes in different elites is beneficial for speeding up the convergence, while destroying some common genes in different inferiors can help an individual jump out of a local optimum; and (3) Considering relatedness between different customers can overcome some shortcomings caused by the random-based removing strategy, and then enhance the efficiency of the local search.

Although CLCS-GA yields very reliable performance on the selected instances, some issues must be further studied. First, efficiently utilizing obtained common genes in elites or inferiors is crucial for improving the convergence speed and enhancing the population diversity, respectively. Moreover, some experiments have verified that a local search strategy is indispensable when optimizing VRPTW. Thus, our future research will focus on designing an efficient local search strategy based on some helpful knowledge extracted from customers position and time windows information. Last, how to define *relatedness* between two different customers during the evolution process is also essential for enhancing an algorithm's local search ability.

CRediT authorship contribution statement

Jiani Liu: Writing – original draft, Data curation. **Lei Tong:** Software, Investigation. **Xuewen Xia:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xuewen Xia reports financial support was provided by National Natural Science Foundation of China. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was funded by the National Natural Science Foundation of China (Grant no. : 61663009) and the Outstanding Young and Middle-aged Scientific and Technological Innovation Teams in Higher Education Institutions of Hubei Province (Grant no. : T2023038).

Data availability

Data will be made available on request.

References

[1] K. Dorling, J. Heinrichs, G.G. Messier, S. Magierowski, Vehicle routing problems for drone delivery, IEEE Trans. Syst. Man Cybern. Syst. 47 (1) (2017) 70–85.
[2] C. Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, H.Y. Lam, Survey of green vehicle routing problem: Past and future trends, Exp. Syst. Appl. 41 (4) (2014) 1118–1138.
[3] N.A. Theeb, H.J. Smadi, T.H. Al-Hawari, M.H. Aljarrah, Optimization of vehicle routing with inventory allocation problems in cold supply chain logistics, Comput. Ind. Eng. 142 (2020) 106341.
[4] E. Pourjavad, E. Almehdawe, Optimization of the technician routing and scheduling problem for a telecommunication industry, Ann. Oper. Res. 315 (2022) 371–395.
[5] Y.J. Gong, J. Zhang, O. Liu, et al., Optimizing the vehicle routing problem with time windows: a discrete particle swarm optimization approach, IEEE Trans. Syst. Man Cybern. Appl. Rev. 42 (2) (2012) 254–267.

[6] G.B. Alvarenga, G.R. Mateus, G.D. Tomi, A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows, Comput. Oper. Res. 34 (6) (2007) 1561–1584.
[7] K. Ghoseiri, S.F. Ghannadpour, Hybrid genetic algorithm for vehicle routing and scheduling problem, J. Appl. Sci. 9 (1) (2009) 79–87.
[8] H. Nazif, L.S. Lee, Optimized crossover genetic algorithm for vehicle routing problem with time windows, Amer. J. Appl. Sci. 7 (1) (2010) 95–101.
[9] W.-J. Wang, Improved genetic algorithm for vehicle routing problem with time windows, in: Proceedings of 2010 International Conference on Intelligent Computing and Cognitive Informatics, 2010, pp. 203–206.
[10] B. Rabbouch, F. Saâdaoui, R. Mraihi, Efficient implementation of the genetic algorithm to solve rich vehicle routing problems, Oper. Res. 21 (2021) 1763–1791.
[11] H.F. Zhang, H.W. Ge, J.L. Yang, Y.B. Tong, Review of vehicle routing problems: models, classification and solving algorithms, Arch. Comput. Method Eng. 29 (2022) 195–221.
[12] X.Y. Wang, T.-M. Choi, Z.Y. Li, S. Shao, An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem, IEEE Trans. Syst. Man Cybern.: Syst. 5 (12) (2020) 4948–4958.
[13] X.Y. Wang, S. Shao, J.F. Tang, Iterative local-search heuristic for weighted vehicle routing problem, IEEE Trans. Intell. Transp. Syst. 22 (6) (2021) 3444–3454.
[14] K.K. Zhang, Y.Q. Cai, S.K. Fu, H.Z. Zhang, Multiobjective memetic algorithm based on adaptive local search chains for vehicle routing problem with time windows, Evol. Intell. 15 (4) (2022) 2293–2294.
[15] P. Kilby, P. Prosser, P. Shaw, A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints, Constraints 5 (2000) 389–414.
[16] P. Kilby, P. Prosser, Solving vehicle routing problems using constraint programming and metaheuristics, J. Heuristics 6 (2000) 501–523.
[17] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithms, European J. Oper. Res. 59 (3) (1992) 345–358.
[18] G. Laporte, M. Gendreau, J.-Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem, Int. Trans. Oper. Res. 7 (4–5) (2000) 285–300.
[19] L.J. Ke, A brain storm optimization approach for the cumulative capacitated vehicle routing problem, Memet. Comput. 10 (2018) 411–421.
[20] B. Ombuki, B.J. Ross, F. Hanshar, Multi-objective genetic algorithms for vehicle routing problem with time windows, Appl. Intell. 24 (2006) 17–30.
[21] V. Tam, K.T. Ma, Combining meta-heuristics to effectively solve the vehicle routing problems with time windows, Artif. Intell. Rev. 21 (2004) 87–112.
[22] J.H. Wang, W.B. Ren, Z.Z. Zhang, H. Huang, Y.R. Zhou, A hybrid multiobjective memetic algorithm for multiobjective periodic vehicle routing problem with time windows, IEEE Trans. Syst. Man Cybern.: Syst. 50 (11) (2020) 4732–4745.
[23] Z. Yang, J.P. van Osta, B. van Veen, R. van Krevelen, R. van Klaveren, A. Stam, J. Kok, T. Bäck, M. Emmerich, Dynamic vehicle routing with time windows in theory and practice, Nat. Comput. 16 (1) (2017) 119–134.
[24] J.C. Jee, Solving vehicle routing problems with time windows using micro-genetic algorithms, in: Proceedings of the 6th National Ungergraduate Research Opportunities Programme Congress, 2000, pp. 7–8.
[25] Y. Caseau, F. Laburthe, Heuristics for large constrained vehicle routing problems, J. Heuristics 5 (1999) 281–303.
[26] J.Y. Potvin, C. Duhamel, F. Guertin, A genetic algorithm for vehicle routing with backhauling, Appl. Intell. 6 (1996) 345–355.
[27] S. Maruyama, T. Tatsukawa, A parametric study of crossover operators in Pareto-based multiobjective evolutionary algorithm, in: Proceedings of the 8th International Conference in Swarm Intelligence, ICSI 2017, Vol. 10386, Springer Verlag, pp. 3–14.
[28] Y. Wang, M.X. Zhang, Y.J. Zheng, A hyper-heuristic method for UAV search planning, in: Proceedings of the 8th International Conference in Swarm Intelligence, ICSI 2017, vol. 10386, Springer Verlag, pp. 454–464.
[29] I.M. Oliver, D.J. Smith, J.R.C. Holland, A study of permutation crossover operators on the TSP, in: Proceedings of the 2nd International Conference in Genetic Algorithms and their Applications, 1987, pp. 224–230.
[30] L. Davis, Applying adaptive algorithms to epistatic domains, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1985, pp. 162–164.
[31] G. Syswerda, Schedule optimization using genetic algorithms, in: L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991, 332C349.
[32] Q.C. Wu, X.W. Xia, H.J. Song, et al., A neighborhood comprehensive learning particle swarm optimization for the vehicle routing problem with time windows, Swarm Evol. Comput. 84 (2024) 101425.
[33] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, Y. Yurramendi, Searching for the best ordering in the structure learning of Bayesian networks, IEEE Trans. Syst. Man Cybern. 26 (4) (1996) 487–493.
[34] W. Banzhaf, The molecular traveling salesman, Biol. Cybernet. 64 (1990) 7–14.
[35] D.B. Fogel, An evolutionary approach to the traveling salesman problem, Biol. Cybernet. 60 (1988) 139–144.
[36] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, I. Inza, S. Dizdarevic, Genetic algorithms for the travelling salesman problem: A review of representation and operators, Artif. Intell. Rev. 13 (1999) 129–170.
[37] M.W.P. Savelsbergh, Local search in routing problems with time windows, Ann. Oper. Res. 4 (1985) 285–305.
[38] X.M. Yan, H. Huang, Z.F. Hao, J.H. Wang, A graph-based fuzzy evolutionary algorithm for solving two-echelon vehicle routing problems, IEEE Trans. Evol. Comput. 24 (1) (2020) 129–141.
[39] L.M. Rousseau, M. Gendreau, Using constaint-based operators to solve the vehicle routing problem with time windows, J. Heuristics 8 (2002) 43–58.
[40] L.H. Lee, K.C. Tan, K. Ou, Y.H. Chew, Vehicle capacity planning system: a case study on vehicle routing problem with time windows, IEEE Trans. Syst. Man Cybern. A 33 (2) (2003) 169–178.
[41] L. Bouthillier, A. Crainic, T.G. Grainic, P. Kropf, A guided cooperative search for the vehicle routing problem with time windows, IEEE Intell. Syst. 20 (4) (2005) 36–42.
[42] Z.Z. Zhang, Y.Y. Sun, H. Xie, Y. Teng, J.H. Wang, GMMA: GPU-based multiobjective memetic algorithms for vehicle routing problem with route balancing, Appl. Intell. 49 (2019) 63–78.
[43] X.W. Xia, H.X. Qiu, X. Xu, Y.L. Zhang, Multi-objective workflow scheduling based on genetic algorithm in cloud environment, Inform. Sci. 606 (2022) 38–59.
[44] H.X. Qiu, X.W. Xia, Y.X. Li, X.L. Deng, et al., A dynamic multipopulation genetic algorithm for multiobjective workflow scheduling based on the longest common sequence, Swarm Evol. Comput. 78 (2023) 101291.
[45] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (2) (1987) 254–265.
[46] G.D. Konstantakopoulos, S.P. Gayialis, E.P. Kechagias, G.A. Papadopoulos, I.P. Tatsiopoulos, A multiobjective large neighborhood search metaheuristic for the vehicle routing problem with time windows, Algorithms 13 (10) (2020) 243.
[47] 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, Inform. Sci. 394–395 (2017) 167–182.
[48] 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.
[49] T.S. Khoo, B.B. Mohammad, Y.H. Wong, Y.H. Tay, M. Nair, A two-phase distributed ruin-and-recreate genetic algorithm for solving the vehicle routing problem with time windows, IEEE Access 8 (2020) 169851-169871.
[50] X.M. Yan, L.C. Tan, J.Y. Chen, A multi-strategy elite ant system algorithm for vehicle routing problem with time window, in: Proceedings of 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education, ICISCAE, IEEE, 2020, pp. 215–220.
[51] 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 (2021) 1–16.
[52] Y.F. Wang, X. Chen, Z. Shuang, Self-competition particle swarm optimization algorithm for the vehicle routing problem with time window, IEEE Acess 12 (2024) 127470-127488.