

## AN IMPROVED SEQUENTIAL INSERTION ALGORITHM AND TABU SEARCH TO VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

BIN YUE<sup>ID</sup>, JIE YANG, JUNXU MA, JINFU SHI\* AND LINJIAN SHANGGUAN

**Abstract.** The vehicle routing problem (VRP) is a well-researched problem in the operations research literature. This paper studies a vehicle routing problem with time windows. Based on Solomon's research, an improved sequential initialization algorithm, Solomon i1 with DTWC (Solomon i1 with Dynamic Time Windows Compatibility), is proposed in this paper to construct feasible initial solutions with higher quality in less calculation time. A tabu search with VNS (Variable Neighborhood Search) and the Metropolis criterion is used to further optimize the solutions that the proposed initialization algorithm provides. We run computational experiments on cases of well-known problems. Results show that compared to the initialization method before the change, our improved initialization approach performs well in terms of providing a better feasible initialization solution in less time, with an average 10% improvement in solution quality and an average 34% computational time savings. The proposed initialization algorithm's superiority is also demonstrated by the tabu search's better performance on the initial solutions it produces.

**Mathematics Subject Classification.** 90C27, 90C59, 90-80, 90B06, 68T20.

Received February 14, 2023. Accepted March 16, 2024.

### 1. INTRODUCTION

The problem of vehicle scheduling was first brought up by Dantzig and Ramser [1], and it has been a popular subject of study in the optimization community ever since. Vehicle routing Problems with Time Windows appear in a variety of scenarios, including grocery distribution [2], and especially surged during the recent COVID-19. The distribution of ready-mixed concrete [3], the supply of oil and petroleum, and other engineering and construction-related studies have all focused on VRPTW [4]. Many extra restrictions or characteristics are frequently present in these connected applications, which could make finding solutions more challenging. In general, the goal of the Vehicle Routing Problem with Time Windows is to arrange vehicle routes so that each customer is served within the time windows that have been set. Otherwise, the vehicles will be subject to the relevant penalties (penalty of the late arrival or early arrival). This kind of problem's main goal is to reduce the value of an objective function, which is often the total distribution cost or total distribution distance.

The scheduling and routing of vehicles through a group of service-requiring consumers is a crucial component of many distribution systems. Customers and delivery services have different concerns in real life. Customers' main priority is getting their deliveries on schedule. To the delivery businesses, getting the items to clients as

---

*Keywords.* VRPTW, sequential initialization algorithm, Dynamic Time Windows Compatibility, Tabu search, VNS.

North China University of Water Resources and Electric Power, Zhengzhou 450046, P.R. China.

\*Corresponding author: [shijinfu@ncwu.edu.cn](mailto:shijinfu@ncwu.edu.cn)

quickly as possible is their main priority. The problem that we research extends the traditional vehicle routing problem by considering a hard time windows and time windows compatibility. The VRPTW belongs to the class of NP-hard combinatorial optimization problems as it generalizes the traveling salesman problem [5] which is NP-hard. According to Kallehauge *et al.* [6], some exact methods have been used to effectively tackle some small-scale problems. However, The exact methods are time-consuming and do not produce satisfactory results for medium- to large-sized instances due to the numerous limitations. In other words, it means that no exact methods exist to solve such problems with large-size. Therefore, some soft computing methods and metaheuristic algorithms have been emphasized and developed for solving optimization problems with a large number of constraints. In handling real-world issues like construction management [7] and numerous engineering optimization problems, soft computing and metaheuristic techniques are crucial. In resource allocation, finding a near-optimum scheduling plan is a common issue with many disciplines such as logistics and transportation problems [8, 9]. Metaheuristics are usually expressed in the form of rules, which can be interpreted as policies. With the introduction of artificial intelligence methods, the latest research trend makes machine learning (including deep learning) an alternative method. Wang *et al.* [10] apply the deep learning method to the capacitated vehicle routing problem (CVRP) and present an overview on how combine deep reinforcement learning for the NP-hard combinatorial optimization, emphasizing general optimization problems as data points and exploring the relevant distribution of data used for learning in a given task. However, it is more difficult to apply deep learning to the other variants of VRP, such as CVRPTW, TDVRP, *et al.* To address NP-hard problems like VRP and variants of VRP, the majority of artificial intelligence algorithms use supervised learning methods [11]. However, this method is more challenging to use in practice since it requires a huge amount of label data. Additionally, as the performance of the developed training model is dependent on the quality of the label data, it is impossible for it to perform better than the label solution. In contrast, when it comes to solving medium- to large-sized VRPTW problems, metaheuristic algorithms significantly outperform exact methods [12]. Metaheuristics do not require label solutions for training; instead, they just require a randomly generated set of initial solutions, followed by the application of specialized search criteria to find near-optimal solutions. As a result, several optimization studies have employed metaheuristics such the genetic algorithm [13], ant colony optimization [14], and particle swarm algorithm [15]. However, they perform badly locally at later stages and have a tendency to lead to premature local convergence. These metaheuristics perform better globally and do not require higher quality initial solutions. Additionally, some metaheuristics with good local search performance include the tabu search [16] and Simulated Annealing [17, 18]. To attain global optimality, these algorithms need more rigorous initial solution quality, nevertheless. Therefore, it is crucial to initially produce higher quality initial solutions when employing an initialization algorithm. Initialization algorithms can be roughly split into sequential and parallel methods [19]. Sequential methods construct one route at a time until all customers are routed [20, 21]. Compared to sequential insertion methods, parallel methods construct a number of routes simultaneously [22, 23], however the number of parallel routes need to be limited to a specific number. Solomon [19] suggests using the sequential insertion method with Insertion-Criterion 1 after thoroughly comparing experimental findings and the stability of various starting algorithms. Solomon i1 will be used to refer to this algorithm moving forward. The sequential insertion method's drawback is that all unrouted consumers are taken into account when determining the insertion and selection criteria for each iteration. This generates a considerable amount of extraneous computation during operation.

In this study, we introduce the DTWC (Dynamic Time Windows Compatibility) to the sequential insertion method Solomon i1 to perform prior evaluation of whether the customer meets the insertion condition and thus reduce unnecessary time wastage on unrouted customers that do not satisfy the insertion condition. Finally, we improve tabu search to further optimize the obtained initial solutions and get final feasible solution. The final result obtained by our tabu search also confirms the effectiveness of the proposed initialization method.

## 2. PROBLEM DESCRIPTION

The VRPTW and the necessary notation for defining the VRPTW are briefly introduced in this section. A bi-directed complete graph  $G = (V, E)$ , where  $V = \{v_0, v_1, v_2, \dots, v_n\}$  is a set of vertices and  $v_0$  representing the depot with  $m$  identical cars, each having a capacity of  $Q$ . Every customer has a non-negative demand and a non-negative service time, as shown by the remaining vertices besides  $v_0$ . The edge set connecting the vertices is given by  $E = \{(v_i, v_j) | v_i, v_j \in V, i \neq j\}$ . A distance matrix  $D = \{d_{ij}\}$  is defined on  $E$ . Each customer has a predefined service window  $[e_i, l_i]$ , where  $e_i$  denotes the earliest start time of the service that the customer  $i$  can accept and  $l_i$  denotes the latest start time of the service that the customer  $i$  can accept. The VRPTW consists of designing a set of routes having a minimum total length such that:

- I. The depot  $v_0$  is where each distribution route begins and ends.
- II. The total demand of all customers on a route does not exceed the capacity  $Q$  of each vehicle.
- III. The total duration (including service time and travel time) of a route dose not exceed  $l_0$ .
- IV. All vertices other than  $v_0$  can only be visited once.
- V. The vehicle arrives at customer  $i$  no earlier than  $e_i$  and no later than  $l_i$ .
- VI. Each vehicle departs from the depot  $v_0$  and returns to the depot  $v_0$  in the time interval  $[e_0, l_0]$ .

The total travel distance of route  $i$  is defined as follow:

$$\text{Dist}_i = \sum_{j=0}^{N_i} d_{k(j,i)k(j+1,i)}. \tag{1}$$

The  $i$ th route has  $N(i)$  customers;  $k(j, i) \in V$  denotes the  $j$ th customer on the  $i$ th route. For each distribution route  $i$ , customer  $j \in \{1, 2, 3, \dots, N(i)\}$ .  $d_{k(j,i)k(j+1,i)}$  denotes the distance between two successively served customers on the  $i$ th route.  $j = 0$  and  $j = N(i) + 1$  represent depot  $v_0$ . In order to calculate the overall travel time, we need to get the time for the vehicle to arrive at the customer location and the time to depart from the customer location. Let  $a_{k(j,i)}$  denote the arrival time of vehicle  $i$  at the  $j$ th customer on the  $i$ th route, and  $f_{k(j,i)}$  denote the departure time of vehicle  $i$  from the  $j$ th customer on the  $i$ th route. The arrival time of vehicle  $i$  at the  $j$ th customer on the  $i$ th route is defined as follows:

$$a_{k(j,i)} = f_{k(j-1,i)} + t_{k(j-1,i)k(j,i)}. \tag{2}$$

The travel time between customer  $j - 1$  and customer  $j$  is denoted by  $t_{k(j-1,i)k(j,i)}$ . The departure time of vehicle  $i$  from depot  $v_0$  is considered to be 0. If the vehicle arrives early at the customer, then the vehicle needs to wait until the customer's earliest service start time. The waiting time of vehicle  $i$  at  $j$ th customer  $k(j, i)$  is computed as follows:

$$w_{k(j,i)} = \begin{cases} 0, & \text{if } a_{k(j,i)} \geq e_{k(j,i)} \\ e_{k(j,i)} - a_{k(j,i)}, & \text{otherwise.} \end{cases} \tag{3}$$

Then the departure time of vehicle  $i$  from  $j$ th customer  $k(j,i)$  can be expressed as follows:

$$f_{k(j,i)} = a_{k(j,i)} + w_{k(j,i)} + s_{k(j,i)}. \tag{4}$$

The overall travel time of the  $i$ th route is computed as follows:

$$T_i = \sum_{j=0}^{N(i)} (t_{k(j,i)k(j+1,i)} + w_{k(j+1,i)} + s_{k(j+1,i)}) \tag{5}$$

$s_{k(j+1,i)}$  represents the service time required by the  $(j + 1)$ th customer of the  $i$ th route, while the waiting time of the depot and the service time required by the depot are both 0, because the depot does not require service.

Hence, the overall waiting time of the  $i$ th route is computed as follows:

$$W_i = \sum_{j=0}^{N(i)} w_{k(j,i)}. \quad (6)$$

There are two types of restrictions on the service time window. The first type with soft time window [24, 25] allows a small time window violation that provides a relaxation to the latest start time of service, while the second type with hard time window [26] requires the vehicle to arrive at a customer no later than the latest arrival time requested by the customer. In this study, we employ the hard time window constraint, which, on the one hand satisfies the strict time requirements of certain items while also reducing the difficulty of calculation to a certain extent.

The late arrival will cause a delay time for a vehicle. When a delay time is generated, it means that the current solution does not meet the hard time window constraints. As a result, the current solution with a delay time will eventually be rejected. The following formula is used to calculate the delay time of vehicle  $i$  at  $j$ th customer on route  $i$ .

$$\text{delay}_{k(j,i)} = \begin{cases} 0, & \text{if } a_{k(j,i)} \leq l_{k(j,i)} \\ a_{k(j,i)} - l_{k(j,i)}, & \text{otherwise.} \end{cases} \quad (7)$$

In contrast to the late arrival of vehicles, the early arrival of vehicles results only in waiting time. Hence, the total delay time of route  $i$  is expressed by the following equation.

$$\text{Delay}_i = \sum_{j=0}^{N(i)} \text{delay}_{k(j,i)}. \quad (8)$$

The vehicle on the route to which it belongs will pay for the waiting and delay time. The following are the three goals of this VRPTW, each of which needs to be minimized:

$$\text{Minimize } f_1 = \sum_{i=1}^N \text{Dist}_i. \quad (9)$$

Objective  $f_1$  denotes the overall travel distance of all vehicles;  $N$  represents the number of vehicles and distribution routes.

$$\text{Minimize } f_2 = \sum_{i=1}^N W_i. \quad (10)$$

Objective  $f_2$  denotes the overall waiting time due to the early arrival of vehicles.

$$\text{Minimize } f_3 = \sum_{i=1}^N \text{Delay}_i. \quad (11)$$

Objective  $f_3$  denotes the overall delay time due to the late arrival of vehicles. Then the overall time violation and travel distance of a solution  $X$  are calculated as follows:

$$T_x = \sum_{i=1}^N (W_i + \text{Delay}_i) \quad (12)$$

$$D_x = \sum_{i=1}^N \text{Dist}_i. \quad (13)$$

The following constraints should be satisfied during the optimization of the above three objectives.

I. Return to depot constraint: all vehicles must return to the depot before the depot's due time.

$$a_{k(N(i)+1,i)} \leq l_{k(N(i)+1,i)}, \quad i \in \{1, 2, \dots, N\}. \quad (14)$$

II. Vehicles' load constraint: the total demand of each route cannot exceed the capacity of vehicle.

$$\sum_{j=1}^{N(i)} q_{k(j,i)} \leq Q, \quad i \in \{1, 2, \dots, N\} \quad (15)$$

$q_{k(j,i)}$  indicates the amount of goods needed by the  $j$ th customer in the  $i$ th route,  $Q$  denotes the maximum capacity of each vehicle, and  $N$  denotes the number of distribution routes and vehicles.

### 3. SOLUTION METHODS

We split the algorithm used to solve the problem that we research into the following two sections.

- I. Finding a feasible initial solution by using the improved initialization algorithm Solomon i1 with DTWC.
- II. We develop a solution method based on the research of Cordeau *et al.* [28] and improve the feasible initial solution through the proposed solution method.

#### 3.1. Initialization algorithm with DTWC

In this section, we use DTWC to improve the Solomon's insertion heuristic and construct a feasible initial solution by taking the time window into account. Solomon's sequential insertion algorithm has a drawback in that it calculates insertion and selection criteria for all unrouted consumers in each iteration. In actuality, this would add a great deal of unnecessary calculations. The introduction of the DTWC can assist in identifying and eliminating the obvious infeasible nodes during the process of node insertion. This result in a more effective and robust construction heuristic.

The purpose of introducing the DTWC is to determine the time overlap of all edges, or node combinations. Before the route construction phase, the DTWC of each customer will be calculated, and the DTWC of each customer will be tested prior to the insertion of customers. Customers that are obviously infeasible will be eliminated from the set of considered nodes. To facilitate the calculation of DTWC, we set the DTWCM (Dynamic Time Windows Compatibility Matrix). DTWCM is a non-symmetrical matrix that stores the DTWC (time compatibility values) for any two customers. Before specifying the calculation of DTWC, it is necessary to explain the meaning of the following notations.

$e_i$ : The earliest allowable arrival time of customer  $i$ .

$l_i$ : The latest allowable arrival time of customer  $i$ .

$s_i$ : The service time required by customer  $i$ .

$t_{ij}$ : The travel time from customer  $i$  to customer  $j$ .

$a_j^{e_i}$ : The actual arrival time at customer  $j$ , given that customer  $j$  is visited directly after customer  $i$  and that the actual arrival time at customer  $i$  is  $e_i$ .

$a_j^{l_i}$ : The actual arrival time at customer  $j$ , given that customer  $j$  is visited directly after customer  $i$  and that the actual arrival time at customer  $i$  is  $l_i$ .

$DTWC_j^i$ : The time windows compatibility value of customer  $j$ , given that customer  $i$  is the predecessor node of customer  $j$  and customer  $i$  has been visited.

There will be six possible scenarios during the delivery of goods. The scenarios depend on the degree and direction of overlap between the time windows of two consecutively visited customers. Figure 1 depicts the six delivery scenarios.

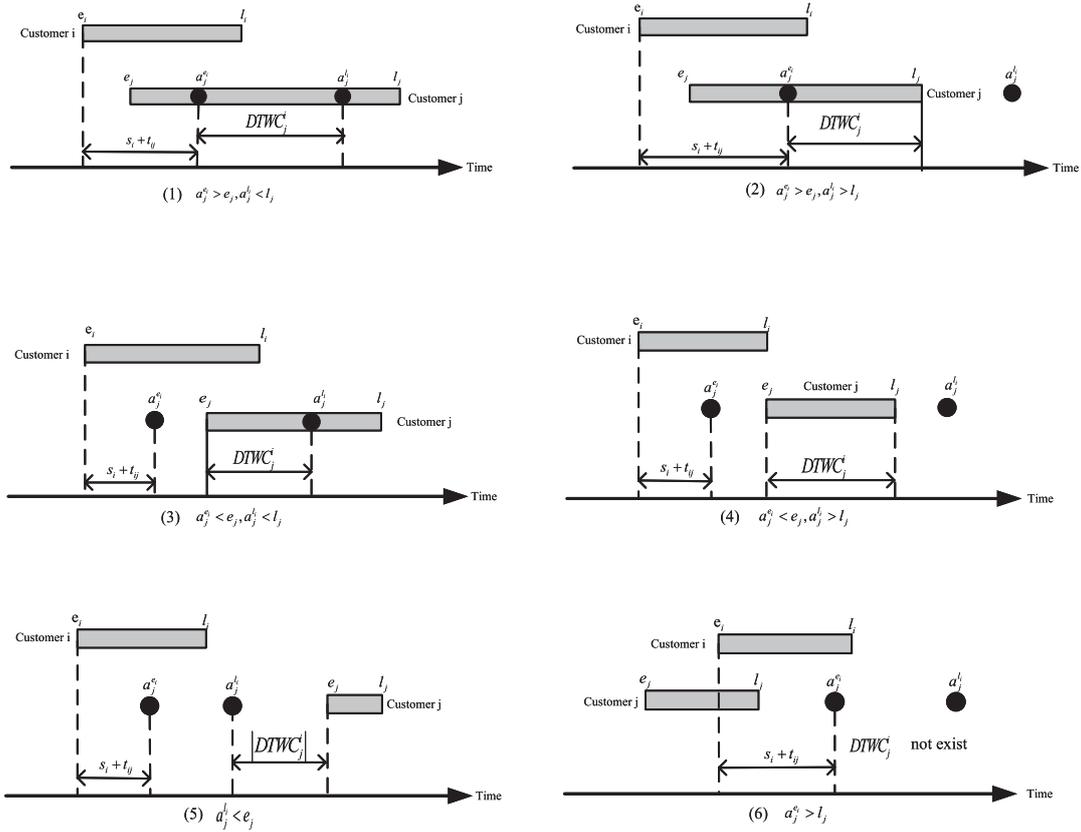


FIGURE 1. Six different delivery scenarios.

Each delivery scenario assumes that customer  $j$  is visited directly after customer  $i$  and describes the relationship between  $e_i, l_i, a_j^{e_i},$  and  $a_j^{l_i}$ . The calculation of DTWC is given below.

$$DTWC_{j}^i = \begin{cases} \left| \min(a_j^{l_i}, l_j) - \max(e_j, a_j^{e_i}) \right|, & \text{if } l_j - a_j^{e_i} > 0 \\ -\infty, & \text{otherwise} \end{cases} \quad i, j \in V. \quad (16)$$

The higher the value of  $DTWC_{j}^i$ , the greater the compatibility of the two customers' time windows. The  $DTWC_{j}^i$  of the incompatible time window is defined to be negative infinity. As the DTWCM is already determined, it is simple to determine the time compatibility of customer  $u$  with customers  $i$  and  $j$  when an alternative customer  $u$  is inserted between customers  $i$  and  $j$ . If either  $DTWC_{u}^i$  or  $DTWC_{j}^u$  is negative infinity, then the time window is incompatible. Without wasting computational effort on computing the insertion cost and selection criteria, the insertion algorithm will move on and examine the next insertion position. Insertion compatibility and cost are calculated only if the time window is compatible.

Gmira *et al.* [27] employed the most common initialization criteria, which are the customer with the earliest deadline and the customer with the greatest distance to the depot, to initialize each route with an initial customer. In this study, the initial customer for a route is determined using the DTWCM. The customer with the highest DTWCS value (the total number of DTWCs that are not incompatible), is chosen as the initial

customer. It is determined how to calculate DTWCS. The value of DTWCS is calculated as follows:

$$TWC_{ij} = \begin{cases} 1, & \text{if } DTWC_j^i \neq -\infty \\ 0, & \text{otherwise} \end{cases} \quad i, j \in V \tag{17}$$

$$assigned_j = \begin{cases} 1, & \text{if customer } j \text{ was visited} \\ 0, & \text{otherwise} \end{cases} \quad j \in V \tag{18}$$

$$DTWCS_i = \sum_{j \in V \setminus \{i\}} (TWC_{ij} - assigned_j), \quad i \in V. \tag{19}$$

The value of  $assigned_j$  represents whether customer  $j$  has been visited. The value of  $TWC_{ij}$  represents whether the time window between customer  $i$  and  $j$  is compatible.  $assigned_j$  and  $TWC_{ij}$  are used to dynamically calculate the DTWCS for each customer. When there are no infeasible time window instances, another method of identifying the initial customer will be used: identifying the initial customer by calculating the compatibility value of the unvisited customers. The compatibility value of unvisited customer is calculated in the following way:

$$compatibility_u = \sum_{\substack{i \in V \setminus \{u\} \\ assigned_i = 0}} DTWC_u^i + \sum_{\substack{j \in V \setminus \{u\} \\ assigned_j = 0}} DTWC_j^u + DTWC_u^u, \quad u \in V, \quad assigned_u = 0 \tag{20}$$

where  $u$  represents the alternate customer to be visited. The initial customer is chosen from those with the lowest compatibility. Figure 2 depicts the sequential initialization approach with DTWC.

The original feasible solution will be further optimized using the improved tabu search algorithm after being achieved using the improved sequential insertion heuristic.

### 3.2. Tabu search with VNS

The structure of the tabu search algorithm is based on the procedure given by Cordeau *et al.* [28]. In this paper, we modify the tabu search with the VNS and metropolis criteria according to the characteristics of the problem that we studied. The obvious difference is that the modified tabu search always starts with a feasible initial solution. This feature ensures that the algorithm will end with a feasible solution. In each iteration, the neighborhood solution of the current solution is generated. In this neighborhood solution, the best solution is selected as the new current solution. Then the algorithm iterates further using this new solution. In the process of algorithm iteration, due to the introduction of metropolis criteria, solutions that violate time windows or vehicle load constraints also have a chance to be accepted, which improves the diversity and global convergence of solutions to a certain extent.

From the structure of solutions, a solution is a set of distribution routes. The solution  $X$  may be infeasible due to the vehicle capacity constraint, and the delivery time of the vehicle exceeds the time window requested by the customer. The degree of load violation and time window violation is the key factor to evaluate the merits of a solution and determine whether to keep the current solution or accept the new solution. The cost function  $C_x$  that is used to evaluate the solution is defined as follows:

$$C_x = D_x + P_t \times T_x + P_l \times Q_x \tag{21}$$

where  $D_x$  represents the total travel distance of all vehicles,  $T_x$  represents the overall time violation of all vehicles, and  $Q_x$  is the overall load violation of all vehicles for each solution generated during tabu search. Parameters  $P_t$  and  $P_l$  are adjusted according to whether parameters  $T_x$  or  $Q_x$  violate the service time window constraint or vehicle load constraint, and the variation of parameters  $P_t$  and  $P_l$  has an upper and lower bound. When either  $T_x$  or  $Q_x$  is eligible, that is, when  $T_x$  or  $Q_x$  equals 0,  $P_t$  or  $P_l$  will be scaled down by a factor that is greater than 1. If  $T_x$  or  $Q_x$  do not satisfy the constraints,  $P_t$  or  $P_l$  will be increased by the same factor. The

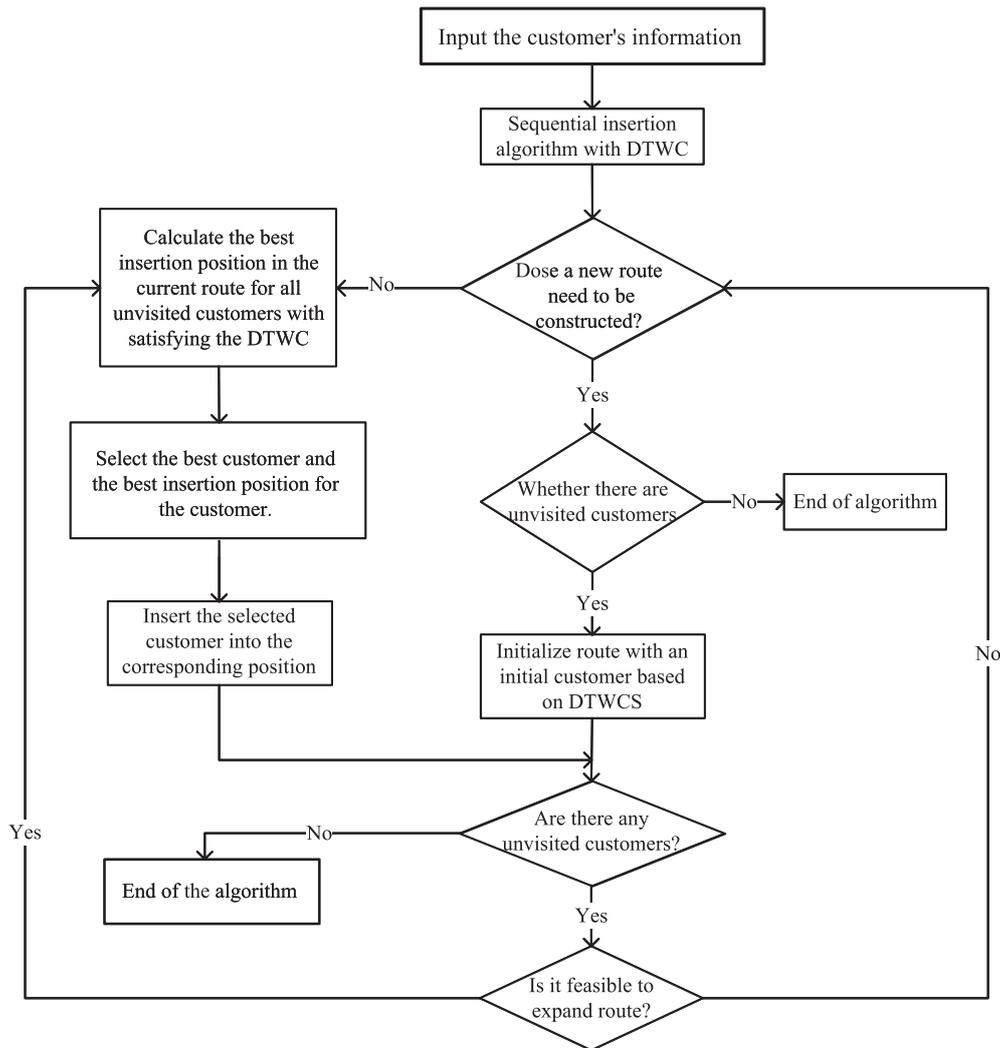


FIGURE 2. The flow chart of the initialization algorithm with DTWC.

adjustment of  $P_t$  and  $P_l$  in turn adjusts the choice of subsequent solutions. Setting the initial value of  $P_t$  and  $P_l$  equal to 1. Parameter  $k(0 < k < \text{max\_iterations})$  in the algorithm ensures that tabu search will terminate when solutions can no longer be optimized further in  $k$  iterations. Setting the tabu length of the solution to be  $L$ . Additionally, the Metropolis criterion is applied during tabu search to improve the variety of solutions, which can largely avoid the algorithm from falling into a local optimal solution.

The neighborhood solution of each iteration of the current solution is obtained by employing three types of relocation operations, defined as follows:

- I. On the same route, swap the locations of two different consumers (Fig. 3).
- II. Exchange two customers between two different routes (Fig. 4).
- III. Swap the remaining parts of two different customers that belong to two different routes (Fig. 5).

The algorithm steps and pseudo code can be viewed in Appendix A.1.

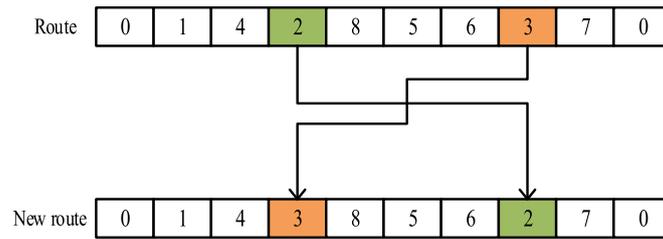


FIGURE 3. Relocation operator I.

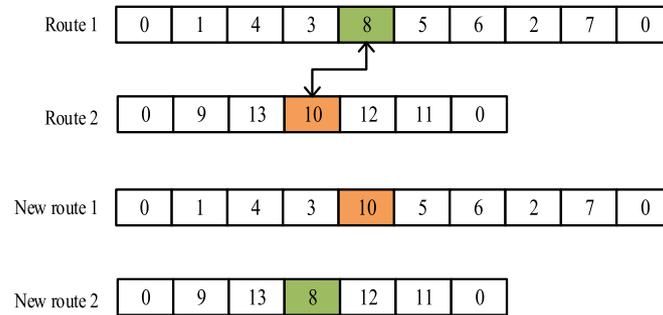


FIGURE 4. Relocation operator II.

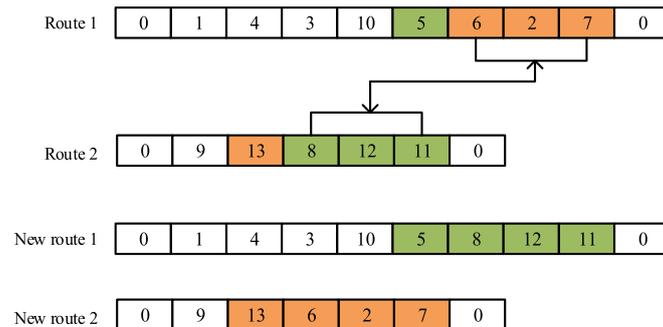
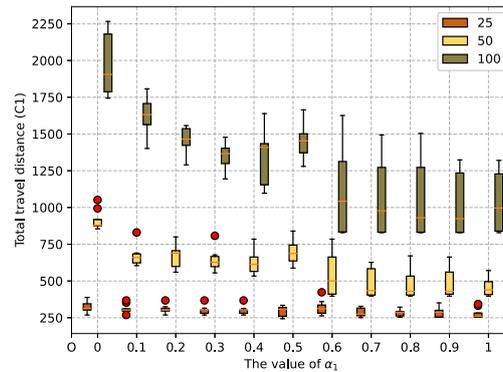
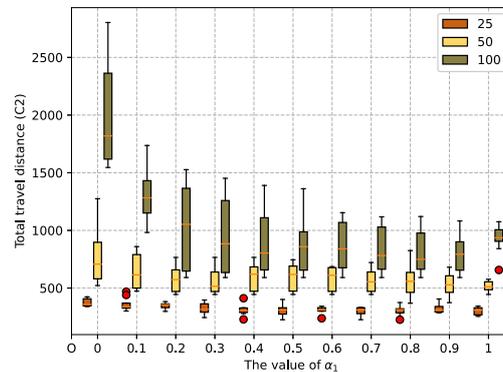


FIGURE 5. Relocation operator III.

#### 4. COMPUTATIONAL EXPERIMENTS AND RESULTS

For the experiments in this part, we make use of Solomon’s data set. In terms of the size of the datasets, three types of data are included in the data set: one depot with 25 customers, 50 customers, and 100 customers. In terms of the distributional characteristics of the dataset, the dataset can be divided into three categories: R, C, RC. Where R stands for customers that are randomly distributed, C stands for customer clustered, and RC stands for mixed data, each of which contains two parts labeled with suffixes using Arabic numerals 1 and 2. Overall, the dataset used in the experiment contains six parts namely R1, R2, C1, C2, RC1, RC2. The vehicle capacities in the dataset are 200, 700, and 1000, respectively. The expected travel time is equal to the corresponding Euclidean distances. The algorithm program used in this paper is written in python, and all experiments are performed on Intel Core i7-10700 CPU @ 2.90 GHz and 8 GB RAM.

FIGURE 6. Results for the C1 dataset with different  $\alpha_1$ .FIGURE 7. Results for the C2 dataset with different  $\alpha_1$ .

#### 4.1. Parameter setting

The external parameters that need to be entered into our initialization algorithm include  $\alpha_1$ ,  $\alpha_2$ ,  $\lambda$ , and  $\mu$ . Referring to the parameters Setting of Solomon i1 [19], the initial values of parameters  $\lambda$  and  $\mu$  are set to 1.  $\alpha_1$  and  $\alpha_2$  have to satisfy the constraint that  $\alpha_1 + \alpha_2 = 1$ . Referring to the approach in [29,30] and in conjunction with the scale of the problem studied in this paper, the parameters ( $L$ ,  $\max\_iteration$ ,  $T$ ,  $\text{cool\_cof}$ ) of our tabu search are equal to (20, 200, 100, 0.4). To find the ideal values of  $\alpha_1$  and  $\alpha_2$ , we set a series of values for  $\alpha_1$  and  $\alpha_2$  to compare the effects of different parameter combinations on the experimental results.

Based on Solomon's experience in parameter setting, this paper is more rigorous and refined in the pre-setting of the parameter  $\alpha_1$ . We Set  $\alpha_1 = \{1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0\}$ ,  $\alpha_2 = \{1 - \alpha_1\}$  for the comparison experiment. The set parameters are applied to the method proposed in this paper to solve the datasets R1, R2, C1, C2, RC1 and RC2 respectively, and the following experimental results are obtained (Figs. 6–11).

Box plots of the above experimental results show that, when  $\alpha_1$  is set to 0.7~0.9, the algorithmic solution results have the overall optimal upper quartile and median values. This suggests that when the value of  $\alpha_1$  falls between 0.7 and 0.9, the insertion initialization method proposed in this paper performs optimally.

#### 4.2. Computational experiments

Based on the parameter settings in the previous section, we conduct comparative experiments with data sets containing 25, 50, and 100 customers. For the fairness and accuracy of comparative experiments, We

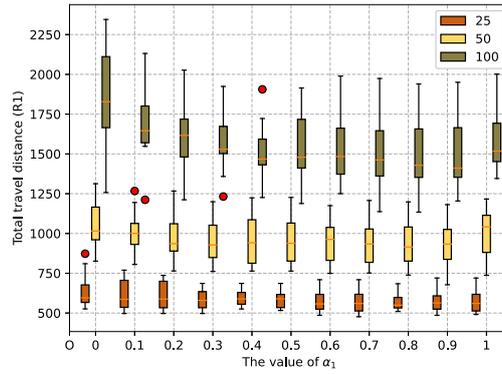


FIGURE 8. Results for the R1 dataset with different  $\alpha_1$ .

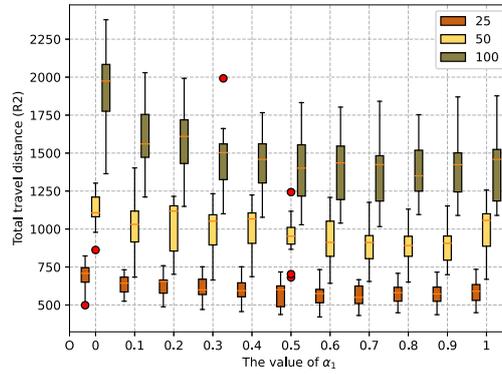


FIGURE 9. Results for the R2 dataset with different  $\alpha_1$ .

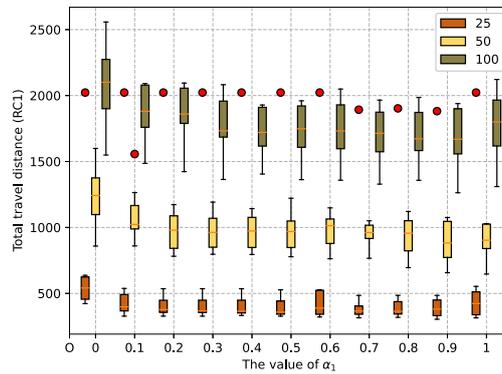


FIGURE 10. Results for the RC1 dataset with different  $\alpha_1$ .

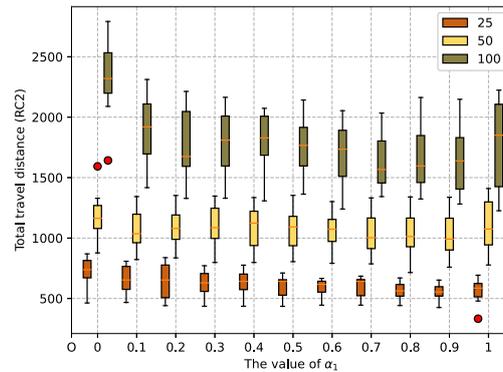
FIGURE 11. Results for the RC2 dataset with different  $\alpha_1$ .

TABLE 1. The results of initialization algorithms and the tabu search: one depot, 100 customers.

Solomon sets	Initial solution				Tabu search with VNS		
	Algorithm	ATD	BTD	ATC	BTC	BTD with minimum $C_x$	Average number of vehicles used
R1	Aprch	1578.89	1349.53	0.1	0.09	1096.7	16
R1	Solomon i1	1523.96	1375.67	0.46	0.32	1158.87	17
R1	Solomon i1 with DTWC	1479.78	1343.58	0.28	0.09	1048.87	16
C1	Aprch	1273.69	855.06	0.22	0.19	839.43	11
C1	Solomon i1	1086.01	875.45	0.74	0.52	828.93	10
C1	Solomon i1 with DTWC	997.21	828.93	0.38	0.11	828.93	8
R2	Aprch	1617.95	1230.29	0.04	0.03	867.44	3
R2	Solomon i1	1439.07	949.3	4.04	2.63	847.88	3
R2	Solomon i1 with DTWC	1393.15	919.29	2.82	0.69	842.87	3
C2	Aprch	1003.69	591.56	0.06	0.04	591.55	4
C2	Solomon i1	915.75	603.88	2.67	2.52	591.55	4
C2	Solomon i1 with DTWC	819.61	591.55	1.18	0.29	591.55	3
RC1	Aprch	1782.26	1537.67	0.11	0.11	1273.19	14
RC1	Solomon i1	1644.76	1328.48	0.45	0.39	1251.9	15
RC1	Solomon i1 with DTWC	1660.08	1327.86	0.28	0.1	1223.65	13
RC2	Aprch	1812.04	1297.2	0.04	0.04	938.94	4
RC2	Solomon i1	1765.87	1321.83	3.31	2.47	1058.03	4
RC2	Solomon i1 with DTWC	1643.52	1240.01	2.32	0.58	957.85	5

independently conducted the experiment three times for each of the three values of  $\alpha_1$  and finally averaged the results obtained. In addition, in order to test the performance of the proposed method on solution quality and solving efficiency, we also include the parallel initialization algorithm. Based on the literature [19], we designed a parallel method called Aprch. Aprch procedures are characterized by the construction of a number of routes simultaneously, the pseudo-code of Aprch is detailed in Appendix A.1. Tables 1–3 show that experimental results on R1, C1, R2, C2, RC1, RC2 of three types of data set, and the results are calculated and obtained at the optimal cost function value  $C_x$ . The specific meanings of the four fields ATD, BTD, ATC and BTC in the following tables are respectively average travel distance, best travel distance, average time consumption, best time consumption.

TABLE 2. The result of initialization algorithms and the tabu search: one depot, 50 customers.

Solomon sets	Initial solution				Tabu search with VNS		
	Algorithm	ATD	BTD	ATC	BTC	BTD with minimum $C_x$	Average number of vehicles used
R1	Aprch	1118.78	784.34	0.02	0.02	681.79	9
R1	Solomon i1	1019.44	792.05	0.1	0.07	696.04	8
R1	Solomon i1 with DTWC	963.47	745.16	0.06	0.02	677.9	8
C1	Aprch	509.33	424.14	0.04	0.03	363.25	5
C1	Solomon i1	501.1	389.82	0.13	0.12	363.24	5
C1	Solomon i1 with DTWC	485.91	397.12	0.07	0.03	363.24	5
R2	Aprch	1009.77	769.23	0.02	0.01	618.54	3
R2	Solomon i1	944.59	667.1	0.65	0.47	551.55	3
R2	Solomon i1 with DTWC	894.76	651.03	0.47	0.11	540.17	3
C2	Aprch	618.93	444.96	0.01	0.01	444.96	2
C2	Solomon i1	524.27	444.96	0.54	0.52	378.79	3
C2	Solomon i1 with DTWC	485.92	373.79	0.26	0.07	352.68	2
RC1	Aprch	923.36	747.26	0.05	0.03	710.17	7
RC1	Solomon i1	903.29	712.02	0.11	0.09	712.02	8
RC1	Solomon i1 with DTWC	862.3	656.92	0.06	0.02	646.82	7
RC2	Aprch	1073.21	727.55	0.02	0.01	576.05	3
RC2	Solomon i1	1000.25	717.87	0.52	0.35	628.03	3
RC2	Solomon i1 with DTWC	999.19	714.58	0.37	0.09	556.5	3

TABLE 3. The result of initialization algorithms and the tabu search: one depot, 25 customers.

Solomon sets	Initial solution				Tabu search with VNS		
	Algorithm	ATD	BTD	ATC	BTC	BTD with minimum $C_x$	Average number of vehicles used
R1	Aprch	536.73	511.21	0.01	0.01	432.12	4
R1	Solomon i1	614.10	548.30	0.05	0.03	427.96	4
R1	Solomon i1 with DTWC	515.70	507.98	0.02	0.01	423.00	4
C1	Aprch	242.36	241.98	0.01	0.01	191.81	3
C1	Solomon i1	251.09	191.81	0.03	0.03	191.81	3
C1	Solomon i1 with DTWC	272.83	190.56	0.01	0.01	191.81	3
R2	Aprch	622.18	513.30	0.01	0.01	490.56	2
R2	Solomon i1	610.59	475.58	0.08	0.06	443.34	1
R2	Solomon i1 with DTWC	587.66	519.65	0.05	0.02	431.21	1
C2	Aprch	313.37	275.62	0.01	0.01	300.77	1
C2	Solomon i1	334.65	237.79	0.07	0.07	261.46	1
C2	Solomon i1 with DTWC	322.96	205.62	0.03	0.01	263.12	1
RC1	Aprch	539.86	485.70	0.01	0.01	395.70	6
RC1	Solomon i1	557.26	318.80	0.02	0.02	334.06	4
RC1	Solomon i1 with DTWC	513.98	318.44	0.01	0.01	298.95	3
RC2	Aprch	598.32	465.61	0.01	0.01	441.64	2
RC2	Solomon i1	558.22	336.28	0.07	0.07	336.28	2
RC2	Solomon i1 with DTWC	533.24	332.67	0.04	0.02	332.67	2

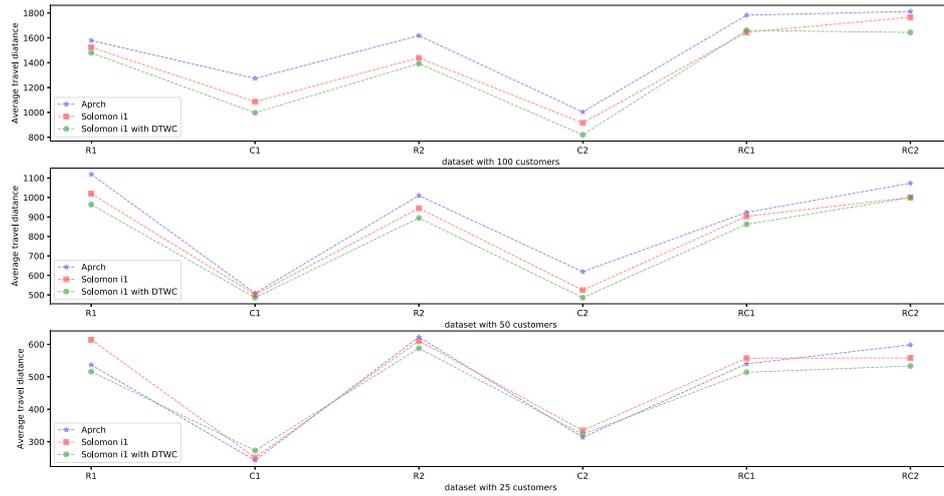


FIGURE 12. The minimum travel distance of different initialization methods.

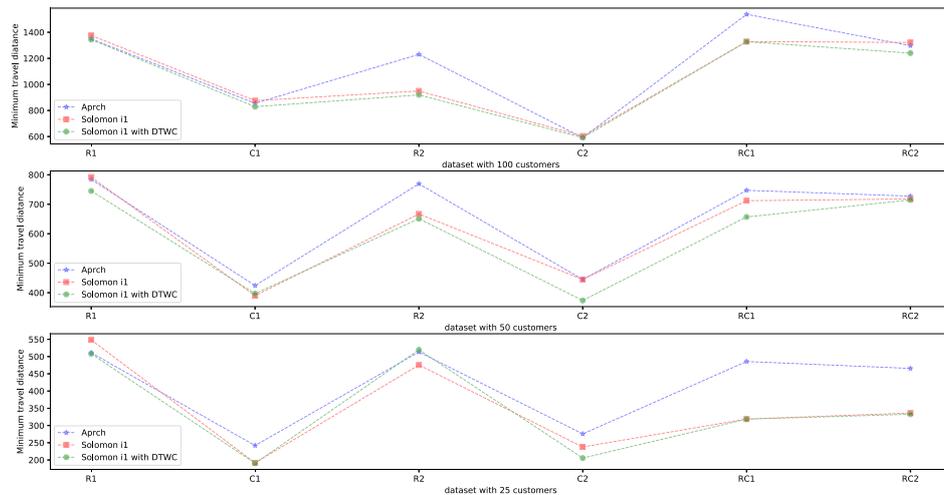


FIGURE 13. Average time consumption of different initialization methods.

Based on the experimental data from Tables 1 to 3, we compare the performance stability (average travel distance of solutions), optimal results (minimum travel distance), time consumption, and final optimization results of three different initialization algorithms, and plot the results in the graphs below.

Three different initialization algorithms are used to solve each data set. On the clustered datasets (C1, C2), all three algorithms perform optimally in terms of average travel distance. From Figure 12, it can be seen that our improved initialization algorithm outperforms the Solomon i1 algorithm on each dataset with a maximum improvement of 13% and a minimum improvement of 8.3%. Aprch has the best performance in terms of time consumption because of its parallel processing advantage; on average, it cuts the running time compared to Solomon i1 by roughly 84%. On all datasets, Aprch has the worst solution quality, nevertheless. Figure 13 shows that the improved initialization algorithm reduces the running time by about 34% on average compared to Solomon i1. The initial solution provided by our proposed initialization method offers the best optimization results when employing tabu search for the final optimization of the initial solutions obtained from the three

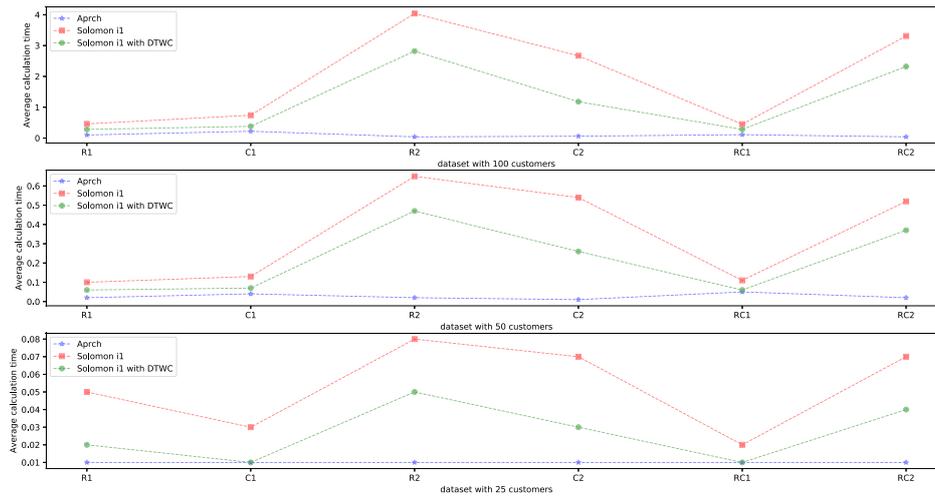


FIGURE 14. The final optimization results of Tabu search.

initialization algorithms. The proposed initialization algorithm’s superiority over the other two initialization algorithms can also be seen from Figure 14, as the optimization result of the tabu search largely depend on the quality of the initial solutions.

### 5. CONCLUSIONS AND FUTURE RESEARCH

In this paper, we focus on a vehicle routing problem with hard time windows and introduce the concept of DTWC to reduce unnecessary calculations. The results show that the proposed initialization algorithm is able to obtain better initial feasible solutions in less time, with a maximum time consumption reduction of 78%. The average running time can be reduced by about 34%. In addition, we propose an initial customer selection strategy based on the DTWCM: the customer with the highest number of infeasible DTWC is identified as the initial customer. This is different from the two most typical initialization criteria, namely the customer with the earliest deadline and the farthest customer. To some extent, it makes the initial customer’s choice more balanced. After obtaining the initial solution, we further optimize it using improved tabu search. The experiment results show that the solution obtained by our improved initialization algorithm has better quality, and the further optimization results of the tabu search corroborate this. Although the parallel algorithm Aprch has a significant advantage in computation time, the initial solution obtained by Aprch is clearly worse than the other two initialization algorithms. From the aspect of dataset, Figure 15 shows that Aprch performs best on clustered (C) datasets containing 25 customers, our improved initialization algorithm performs better on the randomly distributed (R) datasets and the mixed datasets (RC). However, as the number of customers grows to 50 and 100, the advantage of the Aprch does not exist on the clustered dataset. The proposed initialization algorithm consistently performs better. As can be seen, the improved initialization algorithm performs better when there are more customers.

In future studies, we will deal with the specific application of VRPTW in reality, such as the ready-mixed concrete distribution problem. Unlike the VRPTW described in this paper, ready-mixed concrete has more stringent requirements for the delivery time, and one vehicle can only service one customer in one trip. This will involve time dependence and the stochasticity of travel time, and in order to improve the utilization of the vehicle, the multi-site collaboration distribution will be included. The VRPTW that we research in the future need adjustments to the delivery model and algorithms with respect to the distribution of the arrival time.

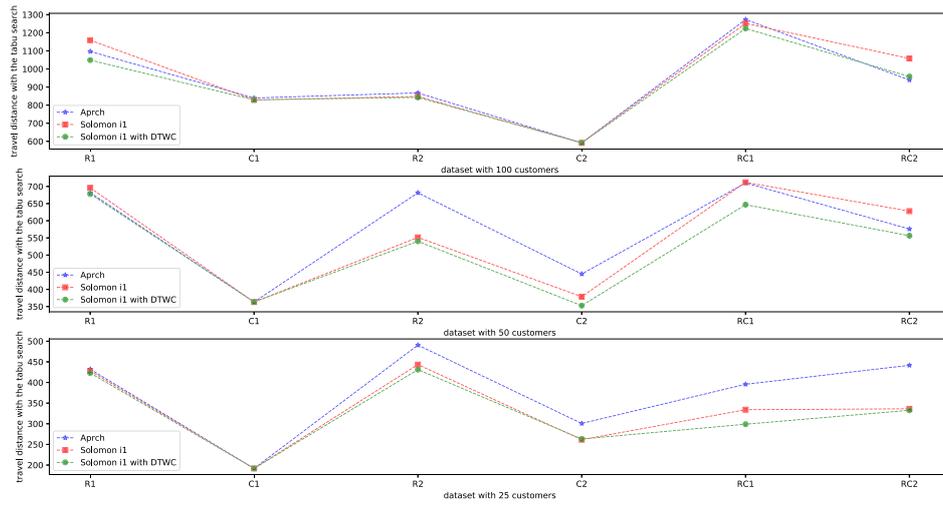


FIGURE 15. Average travel distance under different initialization methods.

## APPENDIX A.

### A.1. Appendix\_1

See Figures A.1–A.3.

**Algorithm1:** Tabu\_search

Set the initial solution  $x$  obtained in the initialization algorithm as the initial solution of tabu search algorithm.

Set current optimal solution  $x^*$  be equal to  $x$ .

Input  $sita$ ,  $P_l$ ,  $P_t$ ,  $upper\_limit$ ,  $lower\_limit$ .

Initialize  $j = 0$ .

**While**  $j \leq L$  (tabu length):

$j += 1$

    Select a solution  $x' \in \mathbf{neighborhood}(x)$  that meet  $C_{x'} < C_x$ , and the solution is not tabu  
 if not find eligible solution:

        Choose a solution  $x' \in \mathbf{neighborhood}(x)$  that minimizes  $C_{x'}$  value and is not tabu

**end**

**if**  $x'$  is feasible **and**  $C_{x'} < C_x$ :

$x^* = x'$  and  $C_{x^*} = C_{x'}$

**end**

**if**  $C_{x^*}$  is update for  $L$  times:

$x = x^*$  and  $C_x = C_{x^*}$

        Update the tabu list

**else:**

$x = x'$  and  $C_x = C_{x'}$

**end**

**Processing of dynamically changing parameters**  $P_l$  and  $P_t$

**if**  $Q_x == 0$  **and**  $P_l \geq lower\_limit$ :

$P_l /= (1+sita)$

**else if**  $Q_x != 0$  **and**  $P_l \leq upper\_limit$ :

$P_l *= (1+sita)$

**if**  $T_x == 0$  **and**  $P_t \geq lower\_limit$ :

$P_t /= (1+sita)$

**else if**  $T_x != 0$  **and**  $P_t \leq upper\_limit$ :

$P_t *= (1+sita)$

**if**  $Q_x == 0$  **and**  $T_x == 0$ :

**return**  $P_l$ ,  $P_t$

**end**

**if**  $x^*$  is not updated for  $k$  times:

**break**

**end**

**end**

FIGURE A.1. Algorithm 1.

---

**Algorithm 2** Tabu search with VNS

---

```

Set  $flag = 1$ ; Set current initial solution as the global
optimal solution.
Input: parameter  $L, max\_iteration, T, cool\_cof.$ 
 $metropolis\ probability = e^{((C_i - C_j)/T)}$ 
While  $flag == 1$  do
    Select two customers in two different routes and each route
    needs to satisfy  $length(route) > 2$ 
If  $customer\_1 == 0$  and  $customer\_2 == 0$ :
        Select two customers in two different routes and each route
        needs to satisfy  $length(route) > 2$ 
else:
        Swap the parts behind two customers that belong two
        different routes.
        cross_exchange( $route\_1, customer\_1, route\_2, customer\_2$ )
end
perform tabu search:
solution  $x = \text{tabu\_search}(route, L, max\_iteration)$ 
if solution  $x$  is feasible and solution  $x$  is better than optimal
solution and current optimal solution:
    update the global optimal solution and the current optimal
solution
else:
    if random probability  $\leq$  metropolis probability:
        update the current optimal solution
    else:
         $flag = flag + 1$ 
end
Update the route for each customer
while  $flag == 2$  do
    Select two customers in two different routes and each route
    needs to satisfy  $length(route) > 2$ 
if  $customer\ 1 == 0$  or  $customer\ 2 == 0$ :
        Select two customers in two different routes and each route
        needs to satisfy  $length(route) > 2$ 
else:
        Exchange two customers in two different routes.
        Exchange_two_route( $route\_1, customer\_1, route\_2,$ 
 $customer\_2$ )
end
end
perform tabu search:
solution  $x = \text{tabu\_search}(route, L, max\_iteration)$ 
if solution  $x$  is feasible and solution  $x$  is better than optimal
solution and current optimal solution:
    update the global optimal solution and the current optimal
solution
else:
    if random probability  $\leq$  metropolis probability:
        update the current optimal solution
    else:
         $flag = flag + 1$ 
end
update the route of each customer
return global_best_solution, global_best_obj

```

---

FIGURE A.2. Algorithm 2.

**Algorithm3:** Aprch (adaptive parallel search)

Input the total number of customers:  $N$

Initialize the set of visited customer:  $assigned\_node = \{\}$

#Calculation of the minimum number of vehicles to be used

$num\_veh\_least = \text{int}(\text{total demand of all customers} / \text{the vehicle capacity}) + 1$

Initialize  $num\_veh\_least$  delivery vehicles.

#Initialize a variable  $flag$  to indicate whether the current client can be inserted at the current position.

$flag = 1e6$  #Initialize the variable  $flag$  to a minimal value.

**While**  $\text{length}(assigned\_node) \leq N$ :

    #Initialize a matrix to store the fitness of the customer after inserting a route.

    #Matrix elements are initialized to infinite.

$fitness = \text{numpy.ones}((num\_veh\_least, N)) * \text{numpy.inf}$

**for**  $k = 1$  **to**  $num\_veh\_least$ :

**for**  $n = 1$  **to**  $N$ :

**if**  $n$  **not in**  $assigned\_node$ :

                #Calculate the fitness of each point inserted at the end of each route.

                #fit\_urg indicates the urgency of the delivery task.

                #fit\_wai indicates the waiting time of customer  $n$ .

$fit\_urg = \text{customer}[n].\text{due\_time} - \text{customer}[n].\text{due\_time}$

$fit\_wai = \max(0, \text{customer}[n].\text{start\_time} - \text{customer}[n].\text{due\_time})$

**if**  $fit\_urg \geq 0$ :

**if**  $\text{sum}(\text{current load, demand of customer } n) \leq \text{capacity of vehicle}$ :

$fitness[k][n] = \text{distance}[n-1][n] + fit\_urg + fit\_wai$ .

**else**:

$fitness[k][n] = flag$  #the customer is not inserted in the current position.

**end if**

**end if**

**end if**

**end for**

**end for**

    #Ascertain whether a new delivery truck need to be activated.

**for**  $i = 1$  **to**  $num\_veh\_least$ :

**if** there exist matrix elements that is equal to 0:

            initialize a new delivery vehicle

            recalculate the fitness matrix

**else**:

**continue**

**end if**

**end for**

    #Select the optimal insertion route and position in the route

    #Select the position with the smallest fitness

$route\_index, node\_index = \text{numpy.where}(fitness == \text{numpy.min}(fitness))$

$\text{insert}(node\_index, route\_index)$

**end while**

FIGURE A.3. Pseudo-code of Aprch.

### Acknowledgements

The project is supported by the National Natural Science Foundation of China: Research on Enterprise Resource location Optimization based on the Internet of things (71371172) and Henan Province 2023 Key R&D and Promotion Special Project: Intelligent Key Technology and Cloud Platform System Research and Development of Ready-mixed Concrete Materials. We gratefully thank the National Natural Science Foundation of China and Henan province for financial support. We would also like to show our gratitude to editor and the anonymous reviewer for their valuable comments and suggestion to improve the paper.

### REFERENCES

- [1] G.B. Dantzig and J.H. Ramser, The truck dispatching problem. *Manage. Sci.* **6** (1959) 80–91.
- [2] B.Y. Ekren, S.K. Mangla, E.E. Turhanlar, Y. Kazancoglu and G. Li, Lateral inventory share-based models for IoT-enabled E-commerce sustainable food supply networks. *Comput. Oper. Res.* **130** (2021) 105237.
- [3] Z. Liu, Y. Zhang, M. Yu and X. Zhou, Heuristic algorithm for ready-mixed concrete plant scheduling with multiple mixers. *Autom. Constr.* **84** (2017) 1–13.
- [4] K.M. Ferreira, T.A. de Queiroz and F.M.B. Toledo, An exact approach for the green vehicle routing problem with two-dimensional loading constraints and split delivery. *Comput. Oper. Res.* **136** (2021) 105452.
- [5] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of vehicle routing and scheduling problems. *Networks* **11** (1981) 221–227.
- [6] B. Kallehauge, N. Boland and O.B.G. Madsen, Path inequalities for the vehicle routing problem with time windows. *Networks* **49** (2007) 273–293.
- [7] R. Tao and C.-M. Tam, System reliability theory based multiple objective optimization model for construction projects. *Autom. Constr.* **31** (2013) 54–64.
- [8] N.N. Yan and D. Zheng, A study on the agent-based vehicles dispatching optimization at container terminals. *Appl. Mech. Mater.* **241–244** (2012) 1745–1750.
- [9] D.-Y. Lin and Y.-H. Ku, Using genetic algorithms to optimize stopping patterns for passenger rail transportation. *Comput.-Aided Civil Infrastruct. Eng.* **29** (2013) 264–278.
- [10] Q. Wang and C. Tang, Deep reinforcement learning for transportation network combinatorial optimization: a survey. *Knowl.-Based Syst.* **233** (2021) 107526.
- [11] R. Basso, B. Kulcsár and I. Sanchez-Diaz, Electric vehicle routing problem with machine learning for energy prediction. *Transp. Res. Part B: Methodol.* **145** (2021) 24–55.
- [12] K.-C. Ying and S.-W. Lin, Minimizing total completion time in the no-wait jobshop scheduling problem using a backtracking metaheuristic. *Comput. Ind. Eng.* **169** (2022) 108238.
- [13] W. Ongcunaruak, P. Ongcunaruak and G.K. Janssens, Genetic algorithm for a delivery problem with mixed time windows. *Comput. Ind. Eng.* **159** (2021) 107478.
- [14] L. Pasandi, M. Hooshmand and M. Rahbar, Modified A\* Algorithm integrated with ant colony optimization for multi-objective route-finding; case study: Yazd. *Appl. Soft Comput.* **113** (2021) 107877.
- [15] A.M. Altabeeb, A.M. Mohsen, L. Abualigah and A. Ghallab, Solving capacitated vehicle routing problem using cooperative firefly algorithm. *Appl. Soft Comput.* **108** (2021) 107403.
- [16] Z.H. Ahmed and M. Yousefikhoshbakht, An improved tabu search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows. *Alexandria Eng. J.* **64** (2023) 349–363.
- [17] Y. Meliani, Y. Hani, S.L. Elhaq and A. El Mhamedi, A tabu search based approach for the heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. *Appl. Soft Comput.* **126** (2022) 109239.
- [18] İ. İlhan, An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem. *Swarm Evol. Comput.* **64** (2021) 100911.
- [19] M.M. Solomon, Algorithm for the vehicle routing and scheduling problems with time windows constraints. *Oper. Res.* **35** (1987) 254–265.
- [20] M.A. Masmoudi, S. Mancini, R. Baldacci and Y.-H. Kuo, Vehicle routing problems with drones equipped with multi-package payload compartments. *Transp. Res. Part E: Logistics Transp. Rev.* **164** (2022) 102757.
- [21] Y. Niu, D. Kong, R. Wen, Z. Cao and J. Xiao, An improved learnable evolution model for solving multi-objective vehicle routing problem with stochastic demand. *Knowl.-Based Syst.* **230** (2021) 107378.
- [22] A. Gutiérrez-Sánchez and L.B. Rocha-Medina, VRP variants applicable to collecting donations and similar problems: a taxonomic review. *Comput. Ind. Eng.* **164** (2022) 107887.
- [23] K.-W. Pang, An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Syst. App.* **38** (2011) 11939–11946.
- [24] C. Chen, E. Demir and Y. Huang, An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *Eur. J. Oper. Res.* **294** (2021) 1164–1180.
- [25] A. Escudero-Santana, J. Muñuzuri, P. Cortés and L. Onieva, The one container drayage problem with soft time windows. *Res. Transp. Econ.* **90** (2021) 100884.

- [26] F. Errico, G. Desaulniers, M. Gendreau, W. Rei and L.-M. Rousseau, The vehicle routing problem with hard time windows and stochastic service times. *Eur. J. Transp. Logistics* **7** (2018) 223–251.
- [27] M. Gmira, M. Gendreau, A. Lodi and J.-Y. Potvin, Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *Eur. J. Oper. Res.* **288** (2021) 129–140.
- [28] J.-F. Cordeau, G. Laporte and A. Mercier, A unified tabu search heuristic for vehicle routing problem with time window constraints. *J. Oper. Res. Soc.* **52** (2001) 928–936.
- [29] V.F. Yu, P. Jewpanya, A.A.N. Perwira Redi and Y.-C. Tsao, Adaptive neighborhood simulated annealing for the heterogeneous fleet vehicle routing problem with multiple cross-docks. *Comput. Oper. Res.* **129** (2021) 105205.
- [30] Y. Meliani, Y. Hani, S.L. Elhaq and A. El Mhamedi, A developed tabu search algorithm for heterogeneous fleet vehicle routing problem. *IFAC-PapersOnLine* **52** (2019) 1051–1056.



**Please help to maintain this journal in open access!**

This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting [subscribers@edpsciences.org](mailto:subscribers@edpsciences.org).

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.