
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 1, pp. 123–135, DOI 10.12694/scpe.v26i1.3128

RESEARCH ON VEHICLE ROUTING PROBLEM WITH TIME WINDOW BASED ON IMPROVED GENETIC ALGORITHM

XU LI*, ZHENGYAN LIU† AND YAN ZHANG‡

Abstract. This article conducts a detailed study on the vehicle routing problem with time window constraints. We constructed an objective function for the vehicle routing problem with time windows, established a mathematical model, and proposed an improved genetic algorithm to solve the problem. The algorithm first constructs a chromosome encoding method, designs a heuristic initialization algorithm to generate a better initial population, and determines the fitness function. During the operation of the algorithm, selection, crossover, and mutation operations are designed to generate offspring populations, enhancing the diversity of the population and avoiding premature convergence of the algorithm. Meanwhile, in order to improve the optimization and local search capabilities of genetic algorithms, this paper constructs a local search operation. Finally, the algorithm implements an elite retention strategy on the parent population and reconstructs a new population. We conducted simulation experiments on the algorithm using MATLAB and selected examples from the Solomon dataset for testing. The simulation experiment results have verified that the improved genetic algorithm is feasible and effective in solving vehicle routing problems with time windows.

Key words: vehicle routing problem, time window, genetic algorithm, local search

1. Introduction. With the rapid development of the logistics industry, optimizing vehicle delivery routes has become increasingly urgent. How to improve logistics distribution efficiency and reduce distribution costs while meeting customer node needs has become an urgent problem that logistics distribution enterprises need to solve [8]. Vehicle routing problem with time windows (VRPTW) has become a hot research topic in recent years due to its strong practical significance [1, 12, 16, 2, 17]. The vehicle routing problem with a time window is an NP hard problem. If the problem is solved using an exact algorithm, it will take too long, while the solution quality obtained by traditional heuristic algorithms is not high. The swarm intelligence optimization algorithm is increasingly being applied to the solution of this problem due to its characteristics of parallelism, universality, and stability. Mst. Anjuman Ara et al. [3] proposed a hybrid genetic algorithm which incorporates three different heuristics for generating initial solution including sweep algorithm, time oriented heuristics and swap heuristic. Khoo Thau Soon et al. [9] proposed the parallelization of a two-phase distributed hybrid ruin-and-recreate genetic algorithm for solving multi-objective vehicle routing problems with time windows. Hongguang Wu et al. [18] proposed a hybrid ant colony algorithm based on ant colony algorithm and mutation operation. Guo Ning et al. [7] proposed an adaptive variable neighborhood search ant colony algorithm to solve the vehicle routing problem with soft time windows. Chen Ying et al. [4] proposed a hybrid particle swarm optimization algorithm based on hierarchical learning and differential evolution.

Due to the strong global optimization ability and good robustness of genetic algorithms, this paper proposes an improved genetic algorithm for solving vehicle routing problems with time windows. We first analyze in detail the characteristics of the vehicle routing problem with time windows and establish a problem model. In algorithm design, we construct a chromosome encoding method, design a heuristic initialization algorithm, and design selection, crossover, and mutation operations. Meanwhile, a local search operation is designed to address the drawback of genetic algorithms being prone to falling into local optima. Finally, simulation experiments are conducted to verify the effectiveness, reliability, and universality of the constructed model and the designed algorithm.

*School of Computer and Information Engineering, Fuyang Normal University, Fuyang, China
†School of Computer and Information Engineering, Fuyang Normal University, Fuyang, China (Corresponding author, zhyliu@fynu.edu.cn).
‡School of Computer and Information Engineering, Fuyang Normal University, Fuyang, China

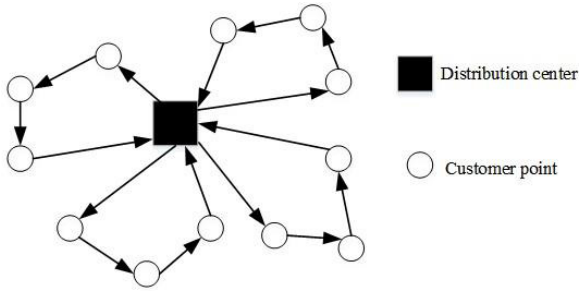


Fig. 2.1: Schematic diagram of vehicle routing problem

2. Vehicle Routing Problem (VRP).

2.1. Problem description. The distribution center distributes goods to customer nodes, and the number, location, and demand for goods at each customer node are known. Transport vehicles start from the distribution center, conduct orderly access to designated customer nodes, and meet some constraints, such as the demand for goods from customer nodes, the time to receive goods, vehicle load capacity limitations, etc., and finally return to the logistics distribution center. Reasonable distribution routes need to be designed to minimize vehicle routes, time consumption, and transportation costs [5, 14]. The schematic diagram of the vehicle routing problem is shown in Fig. 2.1.

2.2. Vehicle routing problem with time windows (VRPTW). The vehicle routing problem with time windows is based on the vehicle routing problem model, which adds a time window as a constraint condition to the customer node. VRPTW is an important branch of VRP, generally described as: several delivery vehicles depart from the distribution center, deliver goods to customer nodes in sequence, complete the delivery task, and return to the distribution center. During the delivery process, there can only be one vehicle serving the customer node, and it is required to deliver goods to the customer node within the specified time period [13]. If the vehicle arrives early or late, corresponding penalties will be imposed.

The vehicle routing problem with time windows can be further divided into soft time window vehicle routing problem and hard time window vehicle routing problem. The soft time window vehicle routing problem refers to the customer's request for logistics delivery vehicles to complete delivery tasks as soon as possible within the specified time period, otherwise they will be subject to corresponding penalty fees. The hard time window vehicle routing problem refers to the logistics delivery vehicle needing to complete the delivery task to the customer within the specified time period, otherwise the customer refuses to receive the goods.

3. Establishment of a Model for VRPTW.

3.1. Model assumptions. In order to facilitate algorithm design and problem research, the following reasonable assumptions are made before modeling [10].

1. The location of the distribution center and customer nodes is determined.
2. Delivery vehicles are of the same type, and each vehicle has the same load capacity.
3. The distance from the distribution center to customer nodes and the distance between customer nodes are known.
4. Assuming that the delivery vehicle departs from the distribution center at a time of 0.
5. Each delivery vehicle can serve multiple customer nodes, but each customer node has only one delivery vehicle serving it.
6. The demand for goods at each customer node is clear.
7. The customer node's cargo demand is less than the maximum load capacity of the delivery vehicle, and cannot be served by multiple delivery vehicles or the same vehicle multiple times.
8. Delivery vehicles can only serve one delivery route.

3.2. Objective function. The optimization objective of the vehicle routing problem with time windows is to select a reasonable driving route while satisfying capacity and time window constraints, in order to minimize the total cost of the entire delivery process [19]. Based on this, the objective function *Cost* designed in this article consists of three parts, namely transportation cost D, capacity penalty cost W, and time penalty cost P. The specific definition is shown in formula (3.1).

$$Min\ Cost = D + W + P \tag{3.1}$$

The calculations for each section are shown below.

1) Transportation cost D. The transportation distance affects the transportation cost, which in turn affects the total cost of logistics distribution. The transportation cost in this article is represented by the transportation distance, which is the total distance traveled by all participating delivery vehicles. The calculation of D is shown in formulas 3.2 and 3.3.

$$D = \sum_{k=1}^{K} \sum_{i=0}^{N} \sum_{j=0}^{N} d_{ij} * x_{ij}^{k} \tag{3.2}$$

$$x_{ij}^{k} = \begin{cases} 1 & \text{if the delivery vehicle } k \text{ goes from customer point } i \text{ to customer point } j \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

Among them, K is the number of vehicles participating in the delivery task; N is the total number of customer points; d_{ij} is the Euclidean distance between customer point i and customer point j; x_{ij}^{k} is a decision variable that indicates whether delivery vehicle k is from customer point i to customer point j; If so, x_{ij}^{k} is 1, otherwise it is 0.

2) Capacity penalty cost W. Delivery vehicles depart from the distribution center and deliver goods to customer points in sequence. After completing the delivery task, they return to the distribution center to form a distribution route. If the total cargo capacity of the delivery vehicle on the delivery route exceeds the maximum load capacity of the vehicle, which violates the capacity constraint, there will be a capacity penalty cost. The calculation of W is shown in formulas 4 to 6.

$$C_k = \sum_{i=1}^{l} q_i \tag{3.4}$$

$$w_k = \alpha * max(C_k - Q, 0) \tag{3.5}$$

$$W = \sum_{k=1}^{K} w_k \tag{3.6}$$

Among them, q_i represents the demand for goods at customer point i; l is the length of the delivery route, which refers to the number of customer points that the vehicle passes through; Q is the maximum load capacity of the vehicle; α is the penalty coefficient for violating capacity constraints; k is the number of vehicles participating in the delivery task; C_k is the sum of the demand for customer points that vehicle k passes through; w_K is the penalty cost for vehicle k violating capacity constraints; W is the penalty cost for the capacity of all delivery vehicles.

3) Time penalty cost P. This article combines the actual situation of logistics distribution and studies the vehicle routing problem with soft time windows. Delivery vehicles are required to deliver goods to customer points within the specified time frame, and there will be no time penalty costs incurred. If the logistics delivery vehicle fails to deliver the goods to the customer's location within the specified time window, there will be

a certain time penalty cost. Based on the established problem model, while considering the complexity and operational efficiency of the algorithm, this paper only penalizes late arriving vehicles and calculates the time penalty cost. The calculation of P is shown in formulas (3.7) to (3.9).

$$P_i^k(S_i) = \beta * \max(S_i - L_i, 0) \tag{3.7}$$

$$P_0^k(B_0) = \beta * \max(B_0 - L_0, 0) \tag{3.8}$$

$$P = \sum_{k=1}^{K} \sum_{i=1}^{N} P_i^k(S_i) + \sum_{k=1}^{K} P_0^k(B_0) \tag{3.9}$$

Among them, β Is the penalty coefficient for violating the time window constraint; S_i is the actual start time of service for customer point i; L_i is the right time window for customer point i, which is the latest service start time for customer point i; $P_i^k(S_i)$ is the penalty cost for delivery vehicle k violating the time window constraint at customer point i; $P_0^k(B_0)$ is the penalty cost for violating the end time of the distribution center time window after vehicle k returns to the distribution center; B_0 is the time when the vehicle returns to the distribution center; L_0 is the end time of the distribution center's time window, which is the latest time the vehicle returns to the distribution center; P is the penalty cost for all delivery vehicles violating time window constraints at all customer points and returning to the distribution center.

The calculation of S_i and B_0 is shown in formulas (3.10) to (3.12).

$$S_i = Max(R_i, E_i) \tag{3.10}$$

$$R_i = S_{i-1} + T_{i-1} + d_{(i-1,i)} \tag{3.11}$$

$$B_0 = S_{end} + T_{end} + d_{(end,0)} \tag{3.12}$$

Among them, R_i is the time when the vehicle arrives at customer point i; E_i is the left time window of customer point i, which is the earliest service start time of customer point i; S_{i-1} is the actual service start time of the previous customer at customer point i; T_{i-1} is the service time required by the previous customer at customer point i; $d_{(i-1,i)}$ is the distance between customer i and the previous customer; S_{end} is the start time of service for the last customer on the delivery route; T_{end} is the service time required for the last customer; $d_{(end,0)}$ is the distance between the last customer and the distribution center.

4. Design of Improved Genetic Algorithm for VRPTW.

4.1. Chromosome coding.

4.1.1. Encoding method. Due to the fact that the quality of encoding directly affects the efficiency and results of algorithm operations, the primary issue in implementing genetic algorithms is to choose the appropriate chromosome encoding method. This article focuses on the characteristics of vehicle routing problems with time windows and adopts natural number encoding. The specific encoding method is as follows.

Number N delivery customer points sequentially, using $1, 2, , N$. M is the predetermined maximum number of vehicles used. K is the actual number of vehicles participating in delivery. A chromosome encoded using natural numbers is shown in Fig. 4.1. R_i is the i-th customer point. Chromosome length is $N + M$.

4.1.2. Conversion between chromosome and delivery route. Find the position of the vehicle number in the chromosome, i.e. Chrome> N, and extract the route before the vehicle number, which is the corresponding customer point route that the vehicle passes through. The driving routes of all delivery vehicles constitute a delivery plan.

For example, assuming that the number of customer points N is 7, the maximum number of vehicles used M is 10, and the actual number of vehicles participating in delivery K is 3, a chromosome is shown in Fig. 4.2.

The delivery plan is as follows:

Delivery route for the first vehicle: Route $\{1\} = \{3, 5, 6\}$;

Delivery route for the second vehicle: Route $\{2\} = \{1, 2\}$;

Delivery route for the third vehicle: Route $\{3\} = \{4, 7\}$.

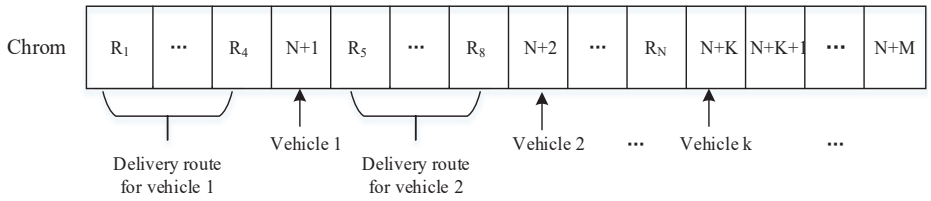


Fig. 4.1: Chromosome coding diagram

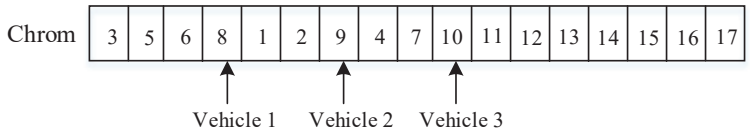


Fig. 4.2: A hypothetical chromosome diagram

4.2. Initialize population. The initial population has a direct impact on the solution of vehicle routing problems with time windows. If the initial population is randomly generated, the generated chromosomes may not be excellent enough, which will have a certain impact on the optimization speed and quality of the optimal solution of the algorithm, leading to the inability of the algorithm to seek the global optimal solution [20]. Therefore, this article designs a heuristic initialization algorithm to generate a better initial population.

The specific steps for initializing the algorithm are as follows.

Step 1. Establish a sequence of traversing customer points The method is as follows:

First, randomly select a customer point i from N customer points;

Then, follow the following sequence seq to traverse each customer point;

If $i = 1$, $seq = [1 : n]$

If $i = n$, $seq = [n, 1 : i - 1]$;

Otherwise, $seq = [i : n, 1 : i - 1]$.

Step 2. Start traversing to obtain the vehicle delivery route

The method is as follows:

First, create a cell array called $Route = cell(k, 1)$ to store the vehicle delivery route. The initial value of k is 1, and $Route\{k\}$ stores the customer point route passed by the kth vehicle.

Then, according to the seq sequence, add the customer points to $Route\{k\}$ in sequence.

When adding a customer point, it is necessary to determine whether the total demand for goods at the customer point Rout $\{k\}$ after addition exceeds the maximum load capacity of the vehicle. If not, add it; otherwise, it cannot be added and an additional vehicle needs to be added to store the customer point.

Finally, arrange the customer points in $Route\{k\}$ in ascending order of the left time window values.

Step 3. Generate initial population

Convert the delivery routes obtained in step 2 into chromosome and generate the initial population.

4.3. Determine fitness function. In genetic algorithms, the fitness function value is used to evaluate the quality of chromosomes and is also the basis for individual evolution. If the fitness value of a chromosome is higher, it indicates a higher degree of excellence of the chromosome [6]. There is a high probability that the individual will be replicated to the next generation. The fitness function value of chromosomes is generally related to the objective function value. In the model constructed in this article, the objective function is to minimize the total logistics cost. Therefore, the fitness function value adopts the reciprocal of the objective function value. The smaller the objective function value, the larger the fitness function value. The definition of fitness function is shown in formula (4.1).

$$Fitness_i = \frac{1}{Cost_i}, \quad i = (1, 2, ..., Z) \tag{4.1}$$

Among them, $Cost_i$ is the objective function value of the i-th chromosome, and the specific calculation is shown in 3.2; $Fitness_i$ is the fitness function value of the i-th chromosome; Z represents the population size.

4.4. Selection operation. The selection operation is based on the fitness value of chromosomes to determine whether they can enter the next generation. The higher the fitness value, the higher the probability of entering the next generation population. On the contrary, the smaller the fitness value, the less likely it is to enter the next generation. This article selects some individuals from the population according to the set generation gap. The selection method adopts random traversal sampling, which generates multiple equally spaced marker pointer positions at once to select the corresponding individuals.

The specific steps for selecting operation are as follows.

Step 1. Calculate the spacing P between pointers, as shown in formulas (4.2) and (4.3);

$$P = \frac{\sum_{i=1}^{Z} Fitness_i}{N} \tag{4.2}$$

$$N = Gap * Z \tag{4.3}$$

Among them, Z is the population size, Gap is the generation gap, and N is the number of individuals to be selected.

Step 2. Randomly generate the starting point pointer position, denoted as *start*, which is a random number between 0 and P;

Step 3. Calculate the positions of each *pointer*, denoted as pointers, as shown in formula (4.4);

$$pointers = start + i * P, \quad (i = 0, 1, \cdots, N-1) \tag{4.4}$$

Step 4. Select N individuals based on the positions of each pointer.

4.5. Cross operation. Cross operation refers to the operation of exchanging one or more bits between two parent individuals to generate a new individual. Cross operation is an important operation in searching the solution space, which can search for the optimal solution within the maximum range. It not only affects the computational efficiency of genetic algorithms, but also affects the computational results of genetic algorithms. This article performs crossover operations on parent individuals based on a certain probability of crossover.

The specific steps for cross operation are as follows.

Step 1. Generate a random number *rand* between 0 and 1. If $rand <= P_c$, P_c is the crossover probability, then go to Step 2 and perform a crossover operation on two adjacent individuals of the parent;

Step 2. Randomly generate two intersection points, denoted as r_1 and r_2, respectively. Swap the two sets of gene sequences between r_1 and r_2, placing them at the forefront of the corresponding chromosome;

Step 3. Eliminate duplicate gene loci to obtain the final offspring chromosome.

For example, A and B are a pair of chromosomes in the parent generation, A' and B' are the offspring chromosomes obtained after crossover operation, as shown in Fig. 4.3.

4.6. Mutation operation. Mutation operator refers to a change in the gene value of one or a few positions in a chromosome, which transforms into other alleles and generates a new individual. The mutation operator can fine tune the new individuals generated by the crossover operator, thereby improving the algorithm's local search ability.

The specific steps for mutation operation are as follows.

Step 1. Generate a random number *rand* between 0 and 1. If $rand <= P_m$, P_m is the mutation probability, then go to Step 2 and perform a mutation operation on the parent chromosome;

Step 2. Randomly generate two gene variants and swap the genes at the two variant positions.

For example, C is the parent chromosome. C' is he offspring chromosome obtained after mutation operation, as shown in Fig. 4.4.

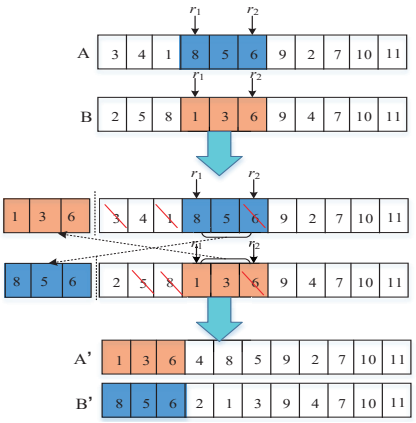


Fig. 4.3: Example of cross operation

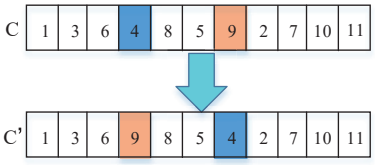


Fig. 4.4: Example of mutation operation

4.7. Local search operation. After selection, crossover, and mutation operations, the diversity of the population is increased, and the algorithm can avoid premature convergence. However, the quality of the optimal solution obtained by the algorithm is not high [11]. In order to improve the optimization and local search capabilities of the algorithm, this paper constructs a local search operation, which mainly includes creating local correlation group and reconstructing vehicle path.

1) Creating local correlation group. Create a local correlation group D based on the correlation between customer points. The algorithm for creating D is described as follows.

Step 1. Randomly select a customer point from the original customer point set and move it to D;

Step 2. Calculate the correlation between the customer point and the other customer points, as shown in formulas (4.5) to (4.7);

$$R_{ij} = \frac{1}{C_{ij} + V_{ij}} \tag{4.5}$$

$$C_{ij} = \frac{d_{ij}}{md_i} \tag{4.6}$$

$$V_{ij} = \begin{cases} 0 & \text{if customer point } i \text{ and customer point } j \text{ are on the same vehicle path} \\ 1 & \text{otherwise} \end{cases} \tag{4.7}$$

Among them, d_{ij} is the distance between customer point i and customer point j; mdi is the maximum distance between customer point i and other customer points.

From the above formulas, it can be seen that the R_{ij} value depends on d_{ij} and V_{ij}. If $V_{ij} = 0$, d_{ij} is smaller and R_{ij} is larger.

Step 3. Select the customer point with the highest relevance and move it to D;

Step 4. If the population size of D is not satisfied, then go to Step 2; otherwise, the algorithm stops.

2) Reconstructing vehicle path. The customer points in the local correlation group D need to be reinserted back into the vehicle path to obtain a new vehicle delivery plan. In order to improve the optimization ability and solution quality, this article ensures that the inserted vehicle path satisfies capacity constraints and time window constraints when inserting customer points back into the vehicle path. Based on this, we construct a reinsertion heuristic algorithm to find the optimal insertion vehicle and location, and reconstruct the vehicle path. The description of the reinsertion heuristic algorithm is as follows.

Step 1. For a certain customer point i in D, first determine whether it can be added to the path $r(r_1, r_2, ..., r_l)$ passed by vehicle k, that is, first determine whether it meets the capacity constraint, and the judgment method is as shown in formula (4.8). If the capacity constraint is not met, customer point i cannot be inserted into the vehicle path $r(r_1, r_2, ..., r_l)$. If the capacity constraint is met, go to Step 2;

$$\sum_{a=1}^{a=l} q\left(r_a\right) + q(i) \leq Q \tag{4.8}$$

Among them, $\sum_{a=1}^{a=l} q\left(r_a\right)$ is the sum of the demand for goods from all customer points in path $r(r_1, r_2, ..., r_l)$; $q(i)$ is the demand for goods at customer point i; Q is the maximum load capacity of the vehicle.

Step 2. Insert customer point i into path $r(r_1, r_2, ..., r_l)$. There will be $l+1$ insertion point positions and generate $l+1$ new paths. Determine whether all customer points on the new path and vehicle returning to the distribution center meet the time window constraint, as shown in formula (4.9).

$$S_j \leq L_j,\ B_0 \leq L_0 \tag{4.9}$$

Among them, j is the customer point on the new path; S_j and B_0 are the actual start time of service at customer point j and the time when the vehicle returns to the distribution center, and the specific calculations are shown in formulas (3.10) to (3.12); L_j and L_0 are the end times of the time windows for customer point j and distribution center.

Step 3. If the time window constraint is met, record the insertion point position, vehicle number, and distance increment.

Step 4. Find the insertion point position and vehicle number with the smallest distance increment, which is the optimal insertion vehicle and position. If it exists, insert customer point i and reconstruct the vehicle path; otherwise, add a new vehicle and deliver to customer point i.

4.8. Reconstruct a new population and save the current optimal solution. The new population consists of elite individuals from the parent generation and offspring. Descendants are obtained through selection, crossover, mutation, and local search operations on their parents. Parent elite individuals are obtained by performing elite preservation operations on the parent population, that is, individuals with smaller objective function values selected in proportion to (1-*Gap*).

Calculate the objective function value for the new population, select the chromosome with the smallest objective function value, and save it as the current optimal solution.

4.9. Calculate the total distance and record the number of paths that violate constraints. Based on the optimal solution, obtain the delivery plan and calculate the total distance traveled by all delivery vehicles. Determine whether each vehicle path violates the load capacity constraint or time window constraint. If it does, record the number of vehicle paths that violate constraints.

4.10. Algorithm framework. The algorithm framework proposed in this article is as follows.

Step 1: Initialize parameters;
Step 2: Initialize the population;
Step 3: Calculate the fitness value of each chromosome in the population;
Step 4: Perform selection operation;
Step 5: Perform cross operation;
Step 6: Perform mutation operation;
Step 7: Perform local search operation;
Step 8: Reconstruct a new population and save the optimal solution;
Step 9: Calculate the total distance and record the number of paths that violate constraints;
Step 10: If the algorithm does not meet the termination condition, go to Step 3.

5. Simulation Experiments. In order to effectively verify the feasibility and effectiveness of the proposed improved genetic algorithm (IGA) in solving VRPTW problems, this paper conducts comparative experiments with the genetic algorithm without local search operation(GA) and the hybrid genetic algorithm (HGA) improved by others [3].

Table 5.1: Parameter Settings

Parameter	*Value*
Population size (Z)	100
maximum iterations	100
Generation gap (Gap)	0.9
crossover probability (P_c)	0.9
mutation probability (P_m)	0.05
The size of local correlation group D	15
Penalty coefficient for violating capacity constraint (α)	10
Penalty coefficient for violating time window constraints (β)	100

Table 5.2: Statistical Results of C101

Algorithm	*GA*	*HGA*	*IGA*
Number of vehicles used	22	11	10
The total distance traveled by vehicles	3557.7041	904.2931	828.9369
Number of paths that violate constraints	12	0	0
The number of times the optimal solution found in 10 runs	3	7	10
The iteration number of the earliest discovered optimal solution in 10 runs	72	68	41

Table 5.3: Statistical Results of C201

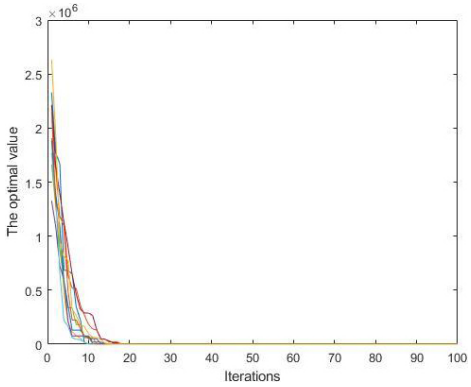
Algorithm	*GA*	*HGA*	*IGA*
Number of vehicles used	21	4	3
The total distance traveled by vehicles	3395.7609	621.5892	591.5566
Number of paths that violate constraints	15	0	0
The number of times the optimal solution found in 10 runs	4	8	10
The iteration number of the earliest discovered optimal solution in 10 runs	8	39	25

5.1. Test examples. The C-type dataset in the Solomon dataset [15] represents that the location of customer nodes is generated based on a structural distribution, divided into two different series of data, namely C1 and C2. We selected two instances, C101 and C201, as representatives of the C-type dataset to analyze the efficiency of our algorithm in solving this type of problem. The calculation example contains some known information, such as the maximum number of vehicles used, the maximum load capacity of vehicles, the location coordinates and time windows of 100 customer points, and the demand for goods.

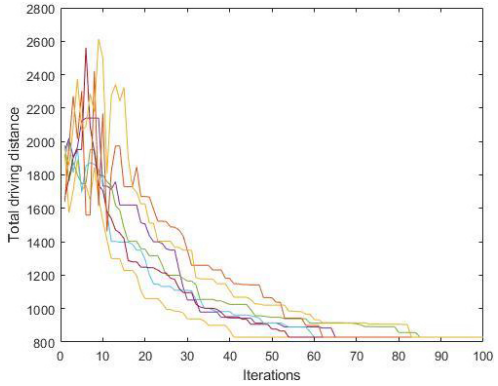
5.2. Experimental environment and parameter settings. The computer configuration used in the simulation experiment is: Core dual core CPU, 2.50GHz, 32GB memory, and Windows 10 system. The simulation software used is MATLAB R2021b. The parameter settings in the algorithm are shown in Table 5.1.

5.3. Experimental results and comparative analysis. To avoid the influence of randomness on the algorithm, we run GA, HGA, and IGA algorithms 10 times on each of the two examples, and recorded the relevant experimental data of the obtained optimal solution. The statistical results are shown in Table 5.2 and Table 5.3.

In order to better observe the evolution process of the proposed algorithm (IGA) and more intuitively reflect its performance, we draw an evolution graph of the optimal value and total distance obtained in each

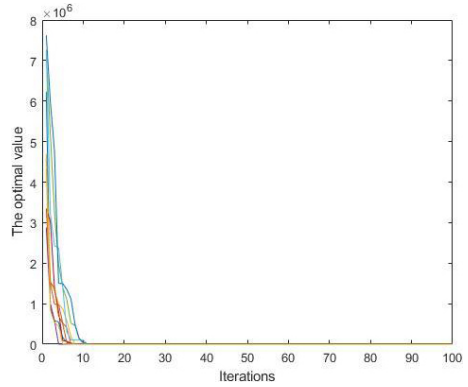


(a) Evolution diagram of the optimal value

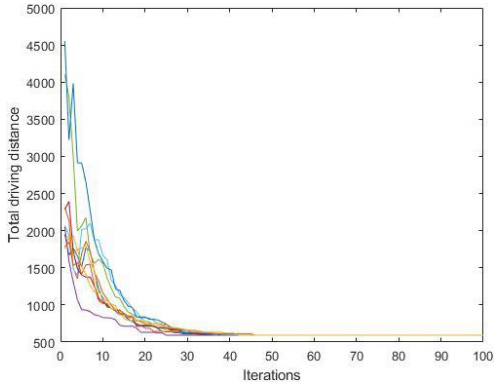


(b) Evolution diagram of total driving distance

Fig. 5.1: Evolution diagram of optimal solution related data for C101



(a) Evolution diagram of the optimal value



(b) Evolution diagram of total driving distance

Fig. 5.2: Evolution diagram of optimal solution related data for C201

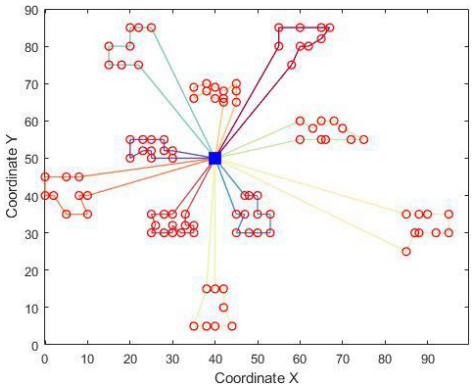
iteration. Taking into account the impact of randomness on algorithm performance, we superimpose the results of 10 runs to evaluate algorithm performance more realistically and objectively. The experimental results are shown in Fig. 5.1 and Fig. 5.2.

From Table 5.2 and Fig. 5.1, it can be seen that the IGA algorithm can find the currently known optimal total distance of C101, which is 828.9369, in each of the 10 runs. At the earliest, it can be found in 41 iterations. From Table 5.3 and Fig. 5.2, it can be seen that the IGA algorithm can find the currently known optimal total distance of C201, which is 591.5566, in each of the 10 runs. At the earliest, it can be found in 25 iterations. At the same time, it can be clearly seen from Fig. 5.1 and Fig. 5.2 that in the early stages of evolution, the total delivery cost and total driving distance of C101 and C201 both show a significant decrease, which also reflects the good optimization ability of the proposed algorithm in solving vehicle routing problems with time windows.

The optimal delivery plan for C101 is shown in Fig. 5.3, requiring 10 delivery vehicles.

The optimal delivery plan for C201 is shown in Fig. 5.4, requiring 3 delivery vehicles.

In order to provide a more intuitive comparison of algorithm performance, we draw an evolutionary comparison chart of the total driving distance obtained by the three algorithms in a random experiment, as shown in Fig. 5.5 for C101 and Fig. 5.6 for C201.

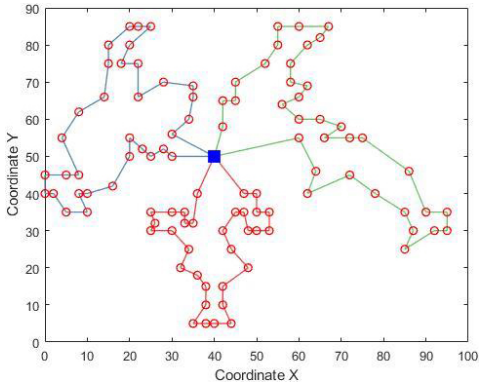


(a) Delivery route map

```
Delivery route for vehicle 1: 0->20->24->25->27->29->30->28->26->23->22->21->0
Delivery route for vehicle 2: 0->67->65->63->62->74->72->61->64->68->66->69->0
Delivery route for vehicle 3: 0->13->17->18->19->15->16->14->12->0
Delivery route for vehicle 4: 0->90->87->86->83->82->84->85->88->89->91->0
Delivery route for vehicle 5: 0->57->55->54->53->56->58->60->59->0
Delivery route for vehicle 6: 0->81->78->76->71->70->73->77->79->80->0
Delivery route for vehicle 7: 0->5->3->7->8->10->11->9->6->4->2->1->75->0
Delivery route for vehicle 8: 0->32->33->31->35->37->38->39->36->34->0
Delivery route for vehicle 9: 0->43->42->41->40->44->46->45->48->51->50->52->49->47->0
Delivery route for vehicle 10: 0->98->96->95->94->92->93->97->100->99->0
```

(b) Specific delivery route map

Fig. 5.3: The optimal distribution plan for C101



(a) Delivery route map

```
Delivery route for vehicle 1:0->20->22->24->27->30->29->6->32->33->31->35->37->38->39->36->34->28->26->23->18->19->16->14->12->15->17->13->25->9->11->10->8->21->0
Delivery route for vehicle 2:0->67->63->62->74->72->61->64->66->69->68->65->49->55->54->53->56->58->60->59->57->40->44->46->45->51->50->52->47->43->42->41->48->0
Delivery route for vehicle 3:0->93->5->75->2->1->99->100->97->92->94->95->98->7->3->4->89->91->88->84->86->83->82->85->76->71->70->73->80->79->81->78->77->96->87->90->0
```

(b) Specific delivery route map

Fig. 5.4: The optimal distribution plan for C201

It can be clearly seen from Fig. 5.5 and Fig. 5.6 that the genetic algorithm without local search operation (GA) cannot converge to the global optimal solution and has poor performance, while the improved genetic algorithm with local search operation (IGA) shows the best performance. The performance of HGA is also inferior to that of IGA. This further confirms the effectiveness of the local search operation in the algorithm proposed in this paper. It not only avoids the defect of genetic algorithms easily falling into local optima, but also enhances algorithm diversity and obtains better optimal solutions.

The above experiment is a specific analysis of the C101 and C201 examples in the Solomon dataset, and has achieved good results. These results indicate that the improved genetic algorithm in this paper is significantly better than GA and HGA in solving quality and stability issues. It is a feasible and effective method for solving vehicle routing problems with time windows.

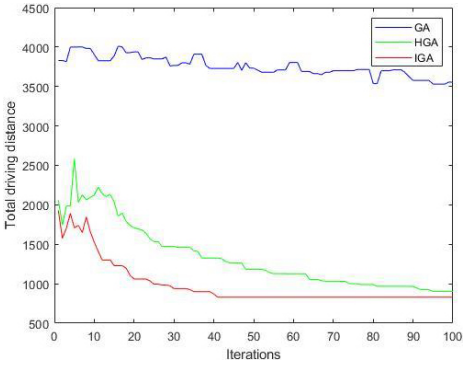


Fig. 5.5: Comparison chart of C101 total driving distance

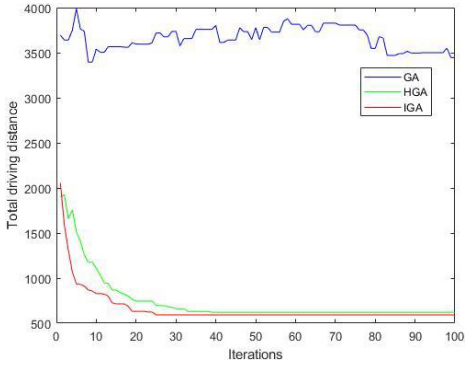


Fig. 5.6: Comparison chart of C201 total driving distance

6. Conclusions and Further Work. This article first analyzes the vehicle routing problem with time windows, constructs the objective function of the problem, and establishes a mathematical model for the problem. In order to improve the optimal solution quality of genetic algorithm in solving vehicle routing problems with time windows, this paper proposes an improved genetic algorithm. The main improvements are as follows: (1) A heuristic initialization algorithm is designed to generate a high-quality initial population; (2) In order to enhance the diversity of the population and avoid premature convergence of the algorithm, selection, crossover, and mutation operations are designed; (3) A local search operation is designed to improve the optimization and local search capabilities of genetic algorithms. Finally, simulation experiments have verified that the improved genetic algorithm is a feasible and effective method for solving vehicle routing problems with time windows. Our future work will be to combine other intelligent optimization algorithms for more in-depth theoretical analysis and algorithm research. At the same time, we will further expand our application scope and conduct research on other types of vehicle routing problems.

Acknowledgments. This work was supported by Anhui University Natural Science Research Key Project under Grant (No.KJ2020A0539, No.2022AH051325, No.2024AH051466).

REFERENCES

[1] H. T. T. Ai, N. T. Thi, and N. V. Can, *A multiple objective model for vehicle routing problem with time windows: a case study*, Applied Mechanics and Materials, 889 (2019), pp. 588–596.
[2] C. Altman, G. Desaulniers, and F. Errico, *The fragility-constrained vehicle routing problem with time windows*, Transportation Science, 57 (2023), pp. 552–572.
[3] M. A. Ara, M. T. Ahmed, and N. Yeasmin, *Optimisation model for simultaneous delivery and pickup vehicle routing problem with time windows*, International Journal of Services and Operations Management, 43 (2022), pp. 145–168.
[4] Y. Chen, P. Huang, J. Chen, Z. Wang, Y. Shen, and X. Fan, *Hybrid particle swarm optimization algorithm based on hierarchical learning and different evolution for solving capacitated vehicle routing problem*, Computer Science, 49 (2022), p. 7.
[5] L. M. Dalbah, M. A. Al-Betar, M. A. Awadallah, and R. A. Zitar, *A modified coronavirus herd immunity optimizer for capacitated vehicle routing problem*, Journal of King Saud University-Computer and Information Sciences, 34 (2022), pp. 4782–4795.
[6] C. S. Ganesh, R. Sivakumar, and N. Rajkumar, *Retraction note: Soft computing-based fuzzy time series model for dynamic vehicle routing problem*, 2023.
[7] M. He, Z. Wei, X. Wu, and Y. Peng, *An adaptive variable neighborhood search ant colony algorithm for vehicle routing problem with soft time windows*, IEEE Access, 9 (2021), pp. 21258–21266.
[8] H.-w. JIANG, T. GUO, and Z. YANG, *Research progress of vehicle routing problem*, ACTA ELECTONICA SINICA, 50 (2022), p. 480.
[9] T. S. Khoo and B. B. Mohammad, *The parallelization of a two-phase distributed hybrid ruin-and-recreate genetic algorithm for solving multi-objective vehicle routing problem with time windows*, Expert Systems with Applications, 168 (2021), p. 114408.

[10] Y. Liu, Y. Yu, Y. Zhang, R. Baldacci, J. Tang, X. Luo, and W. Sun, *Branch-cut-and-price for the time-dependent green vehicle routing problem with time windows*, INFORMS Journal on Computing, 35 (2023), pp. 14–30.
[11] S. Muñoz-Herrera and K. Suchan, *Local optima network analysis of multi-attribute vehicle routing problems*, Mathematics, 10 (2022), p. 4644.
[12] D. A. Neira, M. M. Aguayo, R. De la Fuente, and M. A. Klapp, *New compact integer programming formulations for the multi-trip vehicle routing problem with time windows*, Computers & Industrial Engineering, 144 (2020), p. 106399.
[13] B. Pan, Z. Zhang, and A. Lim, *A hybrid algorithm for time-dependent vehicle routing problem with time windows*, Computers & Operations Research, 128 (2021), p. 105193.
[14] M. Roboredo, R. Sadykov, and E. Uchoa, *Solving vehicle routing problems with intermediate stops using vrpsolver models*, Networks, 81 (2023), pp. 399–416.
[15] M. M. Solomon, *Algorithms for the vehicle routing and scheduling problems with time window constraints*, Operations research, 35 (1987), pp. 254–265.
[16] Y. Wang, S. Luo, J. Fan, M. Xu, and H. Wang, *Compensation and profit allocation for collaborative multicenter vehicle routing problems with time windows*, Expert Systems with Applications, 233 (2023), p. 120988.
[17] Y. Wang, Y. Wei, X. Wang, Z. Wang, and H. Wang, *A clustering-based extended genetic algorithm for the multidepot vehicle routing problem with time windows and three-dimensional loading constraints*, Applied Soft Computing, 133 (2023), p. 109922.
[18] H. Wu, Y. Gao, W. Wang, and Z. Zhang, *A hybrid ant colony algorithm based on multiple strategies for the vehicle routing problem with time windows*, Complex & intelligent systems, (2021), pp. 1–18.
[19] Y. Xiang, Y. Zhou, H. Huang, and Q. Luo, *An improved chimp-inspired optimization algorithm for large-scale spherical vehicle routing problem with time windows*, Biomimetics, 7 (2022), p. 241.
[20] N. Yu, B. Qian, R. Hu, Y. Chen, and L. Wang, *Solving open vehicle problem with time window by hybrid column generation algorithm*, Journal of Systems Engineering and Electronics, 33 (2022), pp. 997–1009.

Edited by: Jingsha He
Special issue on: Efficient Scalable Computing based on IoT and Cloud Computing
Received: Jan 3, 2024
Accepted: Sep 3, 2024