# Optimization for Vehicle Routing Problems: From Basic to Capacitated Models


Jiayuan Shen
MPML
Email: jshen20@umd.edu
UID: 118464947

Qiao Qin
MPML
Email: qqin@umd.edu
UID: 120409875

Ke Xu
MPML
Email: kxu233@umd.edu
UID: 120161216


## I. Introduction

A vehicle routing problem(VRP) is a combinatorial optimization that seeks an optimal solution within a finite set; VRP optimization aims to locate cost-effective routes to service all customers. It is a variation of the Travelling salesman problem (TSP). It first appeared in an article by George Dantzig and John Ramser in 1959.[Dantzig and Ramser(1959)] As an NP-hard problem, VRP presents significant computational difficulties in finding exact solutions for large datasets, necessitating using heuristics to find approximate solutions. To ensure the feasibility of the solutions, we adopt a preset distance matrix that guarantees the existence of the global optimal solution.

Furthermore, by considering the limitation of our knowledge about the optimization field, we solve the VRP problem from the rudimentary one, TSP, and increment the complexity hierarchy, including adding the number of vehicles and eventually transforming it to the Capacity vehicle routing problem(CVRP) to seek the best route that minimizes the overall cost within the requirement of serving all the customers. This step-by-step approach not only assists us in examining the fitness of the two global search algorithms we learned from the lecture to determine the suitable algorithm to solve the CVRP problem but also consolidates our knowledge about implementing the Simulated Annealing and Genetic Algorithm. Finally, we leveraged Google OR-Tools, renowned for their capability to tackle advanced VRP variations like the VRP with Time Windows (VRPTW), to benchmark our algorithms against industry-standard solutions. This comparative analysis not only highlighted the strengths and limitations of our methods but also enriched our understanding of practical optimization in real-world scenarios.

## II. Traveling Salesman Problem

### A. Background

The TSP can be perceived as the fundamental underlying problem of our exploration of seeking the solution of the VRP. The object of the TSP problem is finding the shortest possible route that visits a set of locations exactly once and returns to the origin point(depot). By considering the consistency across all problems we intend to unravel, we utilize the same distance matrix acquired from the Google OR-Tools documentation, which had been used initially to demonstrate the API's performance for different types of VRP problems, ensuring that the solution is computationally achievable and considering finding the solution within a timely manner. We opted to use this particular matrix not only for its demonstrated feasibility but also because it allows us to compare and contrast the solid industry-standard solution in determining the suitable global search algorithm for the CVRP problem, also examining the programming logic while coding the algorithm.

In our configurations, we utilize a symmetric distance matrix to represent the cost of travel between nodes. The matrix is a 17×17 square symmetric matrix where the diagonal elements are all zeros, signifying that the distance from any node to itself is zero. This setup is crucial for the algorithms' accuracy, as it ensures that unnecessary travel to and from the same location is not calculated into the route solutions. For example, the element at matrix[0][1] represents the distance from the depot (node 0) to node 1. By the nature of our symmetric matrix, this value is identical to that at matrix[1][0], reflecting the distance from node 1 back to the depot. This symmetry simplifies the computations and is reflective of real-world scenarios where the route between two points often has equivalent distance and cost regardless of the direction of travel. Here's a simple representation of a portion of this matrix for clarity:

$$\begin{pmatrix} 0 & 548 & 776 & \cdots & 662 \\ 548 & 0 & 684 & \cdots & 868 \\ 776 & 684 & 0 & \cdots & 1552 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 776 & 868 & 1552 & \cdots & 0 \end{pmatrix}$$

Each elements $c_{ij}$ denote the cost (distance) from node $i$ to node $j$, with $c_{ij} = c_{ji}$, emphasizing the symmetric nature of the matrix.

### B. Optimization Problem Formulation

Based on the established background, we can straightforwardly formulate our optimization problem. Consider a set of customers $\mathcal{C} = \{1, \ldots, 16\}$; each route must begin and end at the depot, visiting a subset of these customers exactly once.

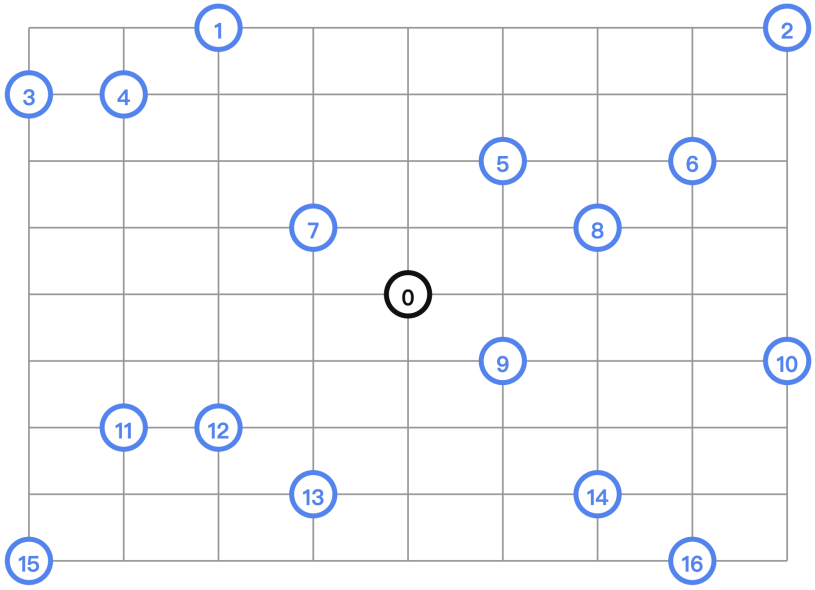


Fig. 1. A visual representation of our scenario, adapted from the Google OR Tools documentation

We represent the scenario using a complete graph, where each vertex is connected to every other vertex.

The graph is denoted as $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \mathcal{C} \cup \{0\}$ and 0 represents the depot. The set $\mathcal{E}$ contains arcs $(i, j)$ for each pair of nodes $i, j \in \mathcal{N}$. The cost of traversing an arc $(i, j) \in \mathcal{E}$ is denoted by $c_{ij}$. A binary decision boundary $x_{ij}$, $i, j \in \mathcal{N}$, is: 1 if and only if there is a route that goes from customer $i$ to $j$; 0 otherwise. Fig.1 is a visual representation of our scenario, sourced from the Google OR-Tools documentation.

**Objective Function:**

$$\text{Minimize} \quad \sum_{i=0}^{16} \sum_{j=0}^{16} c_{ij} x_{ij} \tag{1}$$

**Constraints:**

$$\sum_{i \in \mathcal{N}} x_{ij} = 1 \quad \forall j \in \mathcal{N} \setminus \{0\} \tag{2}$$

$$\sum_{j \in \mathcal{N}} x_{ij} = 1 \quad \forall i \in \mathcal{N} \setminus \{0\} \tag{3}$$

$$\sum_{i \in \mathcal{N} \setminus \{0\}} x_{i0} = \sum_{j \in \mathcal{N} \setminus \{0\}} x_{0j} \leq 1 \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in N \tag{5}$$

The objective function aims to minimize the total travel cost across all routes in the network. Constraint (1) and (2) ensures that each customer, excluding the depot (node 0), is visited exactly once by exactly one vehicle, preventing any customer from being visited more than once. Constraint (3) controls the number of vehicles used in the solution, ensuring that the total number of routes (vehicles entering and leaving the depot) does not exceed 1. [REN(2012)]

### C. Algorithm Implementation

*1) Simulated Annealing:* In our implementation of the Simulated Annealing algorithm for the Traveling Salesman Problem, we carefully selected the following parameters to optimize the search for a near-optimal solution:

- **Initial Temperature** ($T$): We set $T = 10000$. This high initial temperature allows the algorithm to accept worse solutions with a higher probability at the beginning of the annealing process. The purpose of this approach is to avoid premature convergence to local optima, thereby enhancing the exploration of the solution space.
- **Cooling Rate** ($\alpha$): We chose a cooling rate of $\alpha = 0.95$. This relatively slow cooling rate ensures that the temperature decreases gradually, allowing the algorithm more time to explore the solution space thoroughly before it starts to focus more on exploitation, refining towards the best-found solution.
- **Maximum Iterations (max_iter)**: The number of iterations is controlled by the parameter max_iter, which defines the total number of steps the algorithm will perform before stopping. This parameter is crucial for ensuring that the algorithm runs long enough to potentially reach a near-optimal solution but does not run indefinitely.
- **Result**: After several times running the Simulated Annealing code, we acquire the result that the Optimal Route: [0, 13, 12, 11, 15, 3, 4, 1, 7, 5, 8, 6, 2, 10, 16, 14, 9, 0] and Optimal Distance: 4544. Fig.2 shows the relationship between the total distance and the number of iterations.

*2) Genetic Algorithm:* In our implementation of the Genetic Algorithm to solve the Traveling Salesman Problem,

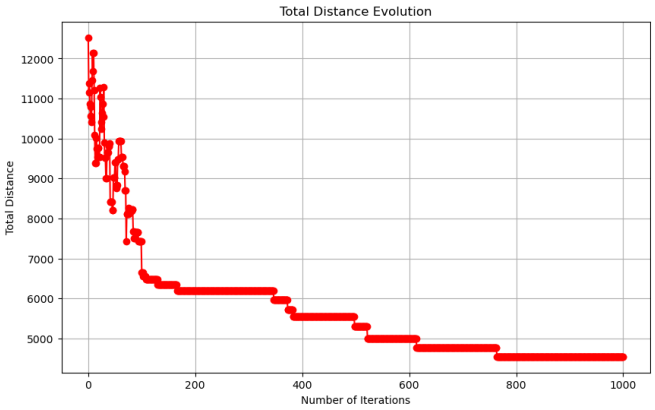


Fig. 2. Simulated Annealing Progression: Total Distance Reduction over Iterations

we meticulously defined several key parameters to govern the evolutionary process:

- **Number of Generations**:We set the algorithm to run for a total of 500 generations. This parameter establishes the termination condition of the algorithm, limiting the evolutionary process to a fixed number of iterations.
- **Fitness Function**:The fitness of each individual (route) is evaluated based on the total travel distance, which serves as the objective function.
- **Population Size**: Each generation consists of 100 potential solutions. This population size ensures a diverse genetic pool while maintaining computational manageability.
- **Tournament Size**: For the selection phase, a tournament size of 5 is used. In each tournament, five individuals are randomly selected, and the best among them (the route with the minimal total distance) is chosen to proceed to the crossover phase.
- **Crossover**: A crossover operation is performed on selected parents to produce offspring for the next generation, which ensures that the offspring is a valid route (no repeated cities) and contains a mix of genes from both parents. Specifically, a segment randomly chosen from the first parent is copied directly to the child. Then, the remaining cities are filled in the order they appear in the second parent, avoiding duplicates.
- **Mutation**: We set the mutation rate at 0.05, indicating that each offspring has a 5 percent chance of undergoing a mutation that perform a random swap of two cities between the offspring, excluding the depot, introducing variability and allowing the algorithm to explore new areas of the solution space.
- **Population Update**: The population for the next generation is formed by replacing the older generation with the new offspring. To prevent a decline in solution quality, the best solutions from the current generation are carried over to the next. This approach helps preserve excellent genetic material and stabilizes performance across generations.
- **Result**: After several times running the Genetic Algorithm, we acquire the the same result with the simulate annealing that the Optimal Route: [0, 13, 12, 11, 15, 3, 4, 1, 7, 5, 8, 6, 2, 10, 16, 14, 9, 0] and Optimal Distance: 4544. Fig.3 shows the relationship between the total distance and the number of generation.

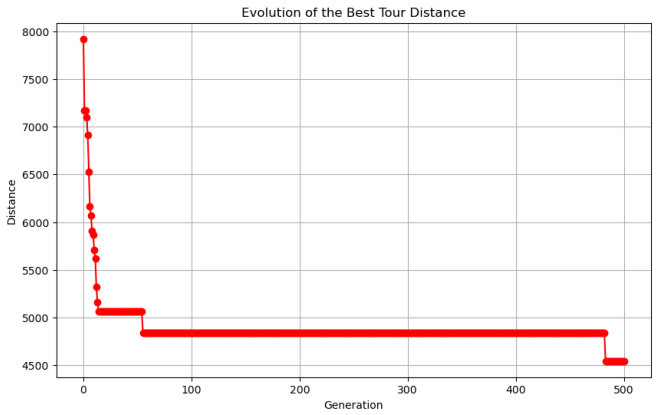


Fig. 3. Genetic Algorithm Progression: Total Distance Reduction over Iterations

### *D. Performance Evaluation*

TABLE I
COMPARISON OF ALGORITHM PERFORMANCE ON TSP

| Algorithm | Optimal Route | Optimal Value |
|---|---|---|
| Simulated Annealing | [0, 13, 12, 11, 15, 3, 4, 1, 7, 5, 8, 6, 2, 10, 16, 14, 9, 0] | 4544 |
| Genetic Algorithm | [0, 13, 12, 11, 15, 3, 4, 1, 7, 5, 8, 6, 2, 10, 16, 14, 9, 0] | 4544 |
| Google OR-Tools | [0, 9, 5, 8, 6, 2, 10, 16, 14, 13, 12, 11, 15, 3, 4, 1, 7, 0] | 4384 |

Table I presents the comparative results of three distinct methodologies applied to solve the Traveling Salesman Problem (TSP) using a comprehensive distance matrix. The Simulated Annealing and Genetic Algorithm techniques produced identical results over multiple trials, underscoring the consistency and reliability of these heuristic methods. However, these results were slightly inferior to those derived from Google OR-Tools, which utilizes more sophisticated, industrial-standard optimization algorithms. Despite the slight discrepancy in results, the divergence between the heuristic solutions and the Google OR-Tools outcome is within an acceptable range, considering the complexity and scale of the distance matrix involved. This variance can be attributed to the inherent differences in how heuristic methods and more deterministic, advanced algorithms like those employed in OR-Tools approach optimization problems. Up until now, we have explored the performance of two heuristic methods, and it seems that both share the limelight. We will continuously dig it by increasing the total number of vehicles to 4.

## III. Four Vehicle Routing Problem

The VRP can be perceived as a variation of the TSP, which adopts more constraints that increase the number of vehicles or salesman within the field of the traditional VRP. The object of the VRP problem is finding the summation of the shortest possible route for all vehicles. Every location (or customer) is visited exactly once by precisely one vehicle. Therefore, we must modify our original TSP problem to suit the VRP.

### A. Optimization Problem Formulation

Based on the previous established formulation, we add a new set $\mathcal{K} = \{1, 2, 3, 4\}$ represent there are total of four avaliable vehciles and for each element $k, k \in [1, 4]$ represent a particular vehicle. A binary decision boundary $x_{ij}^k$, $i, j \in \mathcal{N}, k \in \mathcal{K}$, is: 1 if and only if vehicle $k$ goes from customer $i$ to $j$; 0 otherwise.

**Objective Function:**

$$\text{Minimize} \quad \sum_{k=1}^{4} \sum_{i=0}^{16} \sum_{j=0}^{16} c_{ij} x_{ij}^k \tag{6}$$

**Constraints:**

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} x_{ij}^k = 1 \quad \forall j \in \mathcal{N} \setminus \{0\} \tag{7}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{N}} x_{ij}^k = 1 \quad \forall i \in \mathcal{N} \setminus \{0\} \tag{8}$$

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N} \setminus \{0\}} x_{i0}^k = \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{N} \setminus \{0\}} x_{0j}^k \leq 4 \tag{9}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall i, j \in N \tag{10}$$

### B. Algorithm Implementation

For both the Simulated Annealing and Genetic Algorithm, we set the hyperparameters the same as the previous implementation; however, due to the increase in the complexity, we modify the code to fit the routes of four vehicles of the crossover in the Genetic Algorithm and the mutate route to a different version with the consideration of the different route that each vehicle can take to meet the requirement of the constrain. Fig.4 and Fig.5 show the relationship between the sum of the distance and the number of iteration for Simulated Annealing and the Genetic Algorithm.

TABLE II
Comparison of Algorithm Performance on VRP

| Algorithm | Optimal Route | Optimal Value |
|---|---|---|
| Simulated Annealing | Vehicle 1 Route: [0, 7, 3, 8, 6, 12, 0]<br>Vehicle 2 Route: [0, 9, 5, 0]<br>Vehicle 3 Route: [0, 16, 4, 15, 11, 14, 0]<br>Vehicle 4 Route: [0, 10, 2, 1, 13, 0] | 8536 |
| Genetic Algorithm | Vehicle 1 Route: [0, 1, 3, 4, 7, 0]<br>Vehicle 2 Route: [0, 14, 15, 11, 12, 0]<br>Vehicle 3 Route: [0, 13, 16, 10, 6, 0]<br>Vehicle 4 Route: [0, 9, 5, 2, 8, 0] | 7372 |
| Google OR-Tools | Vehicle 1 Route: [0, 9, 10, 2, 6, 5]<br>Vehicle 2 Route: [0, 16, 14, 8, 0]<br>Vehicle 3 Route: [0, 7, 1, 4, 3, 0]<br>Vehicle 4 Route: [0, 13, 15, 11, 12, 0] | 6300 |

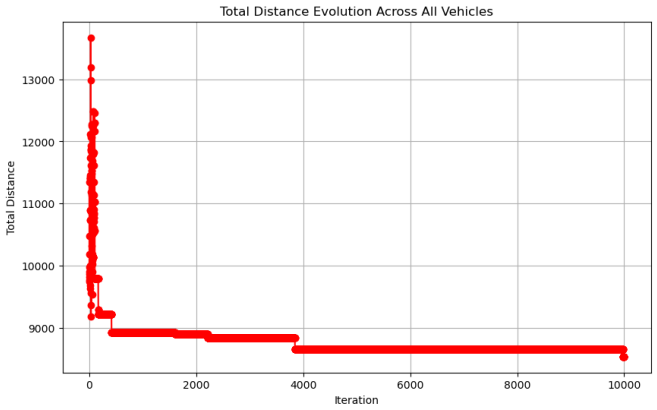


Fig. 4. Genetic Algorithm Progression: The Sum of Total Distance Reduction over Iterations

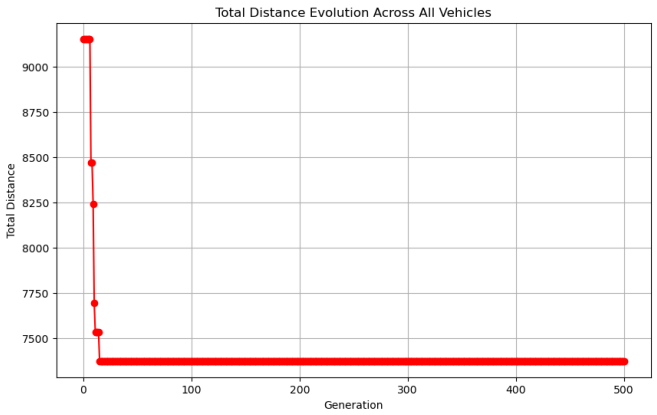


Fig. 5. Genetic Algorithm Progression: The Sum of Total Distance Reduction over Iterations

### C. Performance Evaluation

Table II shows the result we give precisely after running the code several times; we can observe the vast difference between these three methods that the distinction is relatively huge compared to the two heuristic solutions and the Google OR-Tools, which not only because of the more robustness of the advanced algorithm that Google OR-Tools utilized but also because the dynamic programming requirement while tracking the previous solution and the current solution by considering the different size of the route array significant enhance the effectiveness and efficient of the industrial-standard solution. Our implementation is highly restricted by the initialization array, which might make the result invariant regarding the crossover and requires careful consideration for route management and fitness evaluation. The need for dynamic adjustment in programming to cater to different route sizes can significantly increase the complexity of coding and testing these algorithms. On the other hand, through this particular experiment, the result from the Genetic Algorithm is slightly better than Simulated Annealing due to the implementation of the mutation and crossover inside the Genetic Algorithm, which significantly mitigates the effectiveness of the initialization.

Therefore, we will implement the Genetic Algorithm for the CVRP problem.

## IV. Capacity Vehicle Routing Problem

The CVRP(Capacity Vehicle Routing Problem) is an extension of the VRP, which involves more constraints than just increasing the number of vehicles compared with the TSP problem, which is the capacity of the vehicles and the demand required for each customer. Therefore, we constantly track the loading stage. [Mohammed et al.(2012)Mohammed, Ahmad, and Mostafa]At the same time, the vehicle passes through a specific customer to ensure that the current load does not exceed the vehicle's capacity and minimize the sum of the total cost for each vehicle. For the sake of simplicity, we consist of four vehicles and assume each vehicle has the same capacity. The demand matrix is acquired from the Google OR-Tools documentation to promptly ensure the optimization is solvable.

### A. Optimization Problem Formulation

Based on the previous optimziation formulation, we incline to add two more parameters $Q$ to represent the capacity of the vehicles and $q_i$ represent the demand of each nodes. Therefore we add one more constrain:

**Constraints:**

$$\sum_{i=1}^{16}\sum_{j=2}^{16} q_j x_{ij}^k \leq Q \quad \forall k \in \mathcal{K} \tag{11}$$

### B. Performance Evaluation

Table III shows the result we get after running the code several times and the reliable result we could acquire using the external libraries.

TABLE III
Comparison of Algorithm Performance on VRP

| Algorithm | Optimal Route | Optimal Value |
|---|---|---|
| Simulated Annealing | Vehicle 1 Route: [0, 5, 6, 2, 8, 0]<br>Vehicle 2 Route: [0, 12, 11, 15, 13, 0]<br>Vehicle 3 Route: [0, 7, 1, 4, 3, 0]<br>Vehicle 4 Route: [0, 14, 16, 10, 9, 0] | 6208 |
| Google OR-Tools | Vehicle 1 Route: [0, 4, 3, 1, 7, 0]<br>Vehicle 2 Route: [0, 14, 16, 10, 9, 0]<br>Vehicle 3 Route: [0, 12, 11, 15, 13, 0]<br>Vehicle 4 Route: [0, 8, 2, 6, 5, 0] | 6208 |

## V. Limitation and Future Improvment

Even though our algorithm achieves the same result as the Google OR-Tools, we test our code multiple times and fix tremendous bugs to mitigate the problem of missing the customers and passing through the same customers multiple times. Eventually, we made some processes to solve the problem, but the patch still needs to be the perfect solution, and the situation above might still occur while running the code chunk. By missing the ability to track the size of the routes and increasing the constraints to the capacity, we are unable to make the size of the generation and the size of the population similar to the original setting, which significantly reduces the further scalability of our Genetic Algorithm, whereas the Google OR-Tools; however, always correctly locate the global optimal point within the time manner. For possible future improvement, we could refine the existing algorithm to make it more robust while including the method of dynamic programming to further divide a giant optimization problem into several tangible tasks to solve the problem. Also, we will attempt to solve the vehicle routing problem with a time window and merge the constraints of time windows and vehicle capacity in the future. Throughout this project, we also reinforce our knowledge about the complexity of tackling the NP-hard problem, which demands clear management skills in handling constraints and variables from the previous solution to the current solution. Furthermore, we also understand why many research papers about VRP indicate a constraint that prevents sub-tour solutions, which is vital to constrain while writing and understanding the optimization problem.

## VI. Conclusion

In this project, we first explore the solution of the traditional TSP by using the Simulated Annealing and the Genetic Algorithm; the result shows that both algorithms are highly suitable for solving TSP. Furthermore, we continuously investigate the usage of two algorithms by slightly increasing the difficulty of TSP in evolving as a VRP problem in which the Simulated Annealing algorithm did not generate the appropriate result as the Genetic Algorithm. Finally, we attempt to solve a CVRP problem using our Genetic Algorithm, where we encounter the problem of numeric stability and the difficulty of tracking multiple constraints simultaneously. For each of our solutions, we contrast with the external library and examine the distinction. We each did a third of the project.

## References


[Dantzig and Ramser(1959)] George Bernard Dantzig and John Hubert Ramser. The truck dispatching problem. *Management Science*, 6(1): 80–91, October 1959. doi: 10.1287/mnsc.6.1.80.

[Mohammed et al.(2012)Mohammed, Ahmad, and Mostafa] Mazin Abed Mohammed, Mohd Sharifuddin Ahmad, and Salama A. Mostafa. Using genetic algorithm in implementing capacitated vehicle routing problem. In *2012 International Conference on Computer Information Science (ICCIS)*, volume 1, pages 257–262, 2012. doi: 10.1109/ICCISci.2012.6297250.

[REN(2012)] Chunyu REN. Applying genetic algorithm for capacitated vehicle routing problem. In *Proceedings of the 2nd International Conference on Electronic Mechanical Engineering and Information Technology (EMEIT 2012)*, page page numbers if available, Location of the Conference, 2012. Atlantis Press. doi: 10.2991/emeit.2012.107. URL `https://www.atlantis-press.com/proceedings/emeit-12/3325`.