Proceedings of the 11th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'25)
Paris, France - August, 2025
Paper No. CIST 154
DOI: 10.11159/cist25.154

Solving the Vehicle Routing Problem via Distance-Aware Clustering and Simulated Annealing

Yusuf Akyol[1], Abdül Halim Zaim[2], Birnur Elif Cırık[3]
[1]Istanbul Technical University, Informatics Institute
Sarıyer, Istanbul, Turkey
akyoly15@itu.edu.tr; azaim@itu.edu.tr
[2] Istanbul Technical University, Department of Computer Engineering
Sarıyer, Istanbul, Turkey
[3] Istanbul Technical University, Department of Electronics and Communication Engineering
Sarıyer, Istanbul, Turkey
cirik24@itu.edu.tr

Abstract **–** The problem of routing a given number of vehicles leaving a depot to serve customers is known as the *Vehicle Routing Problem* (VRP). VRP is used in various fields such as logistics, supply chain and distribution. To solve VRP, in this study, we propose a solution which uses a heuristic algorithm that we developed to distribute customers to vehicles and then optimizes the route of each vehicle using Simulated Annealing technique. Our solution aims to solve VRP by generating routes of similar length for each vehicle in a short enough time to be used in real-time applications when no capacity value is given for the vehicles. To measure the performance of our solution, we compared it with OR-Tools, an open source VRP library, using problem instances that we have created by generating synthetic data. We found that in most cases it performed better and was able to create shorter routes. Thus, we consider it as an effective and performant solution for classical VRP. Since we offer a direction-oriented solution, we think that it produces useful routes in real-life problems, especially in distribution-based real-life problems.

Keywords**:** Vehicle Routing Problem, Optimisation, Simulated Annealing

1. Introduction

The vehicle routing problem is a general name for the problems including a set of customers and a set of vehicles that will serve them. The problem emerged from the practical need for supplying goods to the customers by the vehicles departing from a depot. In its basic form, the goal is to assign each vehicle a route as a possible shortest way that starts from the depot and visits a subset of customers and returns to the depot such that each customer will be visited once only by a vehicle [1].

Historically, the problem is formulated as a generalization of *Travelling Salesman Problem* (TSP) with the name of Truck Dispatching Problem in 1959 by Dantzig and Ramser. Considering the TSP as the problem of finding the possible shortest route among n given points by passing through the points only once, the vehicle routing problem is generalised by adding some additional constraints to it [2]. After the problem was proposed in 1959, a vast number of variations of the problem were proposed: capacitated VRP (CVRP), VRP with pickup and delivery, dynamic VRP (DVRP), VRP with time windows [3]. The increase in the computing power of computers in the last decades has allowed an improvement in the solutions for complex VRP problems, which has led to a rapid increase in the number of solution techniques for VRP presented in the literature in the last few decades [4].

VRP is an NP-hard problem. The simple solution is to count all possible routes one by one and find the shortest route. However, this process becomes almost impossible to compute when the size of the problem and the number of customers increase slightly. Various heuristic methods have been used to overcome this problem in literature, and acceptable results have been obtained at low cost, which are not guaranteed to be the best [5]. In this paper, a solution using a heuristic algorithm will be presented.

In cases where vehicle capacities are not given or are infinite, we observe that there are few VRP solutions where routes of close length are created and those that exist are not efficient in terms of solution time. The main contribution of this paper is to propose a direction-oriented optimization algorithm to fill this gap in the literature, which produces good results in short

periods of time. The paper first gives a mathematical description of the classical VRP in Section 2. Then in Section 3, the proposed solution is explained in detail and expressed mathematically. Section 4 shows the results of the algorithm on various problem instances and a benchmark comparison with the OR-tools library. Finally, in Section 5 we discuss how the algorithm can be improved in the future for different VRP variations and its advantages in real-life problems.

2. The Problem

Consider a set of customers C = {1...n}, each having a positive demand q_i. And consider a set of K vehicles responsible for supplying the demands for the customers from a depot. To meet the demands of the customers, we must map a route for each vehicle leaving the depot in such a way that no customer is left unvisited. Each vehicle must depart from the depot, visit a subset of customers, and return to the depot. No customer should be left unvisited, and each customer should be visited at most once. The goal is to minimize the sum of travelled distance (or the time that it takes) of routes.

We can define the problem using a complete graph $G(N, \mathcal{E})$ where N is the set of nodes representing the customers and $\mathcal{E}$ is the set of arcs for each pair of nodes i, j. The cost of going from node i to j, or the cost of passing through the arc $\mathcal{E}_{ij}$ is denoted by c_{ij}. For the classical VRP we assume the cost of going from node i to j is the same as going from node j to i. By defining the binary decision variable x_{ij}, which has the value 1 if and only if the arc from node i to node j is included in a route, for $i, j \in N$, the objective function can be given as below with the constraints.

$$min \sum_{i \in V} \sum_{i \in V} c_{ij} x_{ij} \tag{1}$$

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \backslash \{0\} \tag{2}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \backslash \{0\} \tag{3}$$

$$\sum_{i \in V \backslash \{0\}} x_{i0} = K \tag{4}$$

$$\sum_{j \in V \backslash \{0\}} x_{0j} = K \tag{5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \forall S \subseteq V \backslash \{0\}, S \neq \emptyset \tag{6}$$

$$x_{ij} \in \{0,1\} \; \forall i, j \in V \tag{7}$$

The objective function is stated in (1) subject to the rest constraints. The objective is to minimize the total cost of the routes with the given constraints. K is the number of vehicles and $r(S)$ is the minimum number of vehicles needed to serve S where $S \subseteq V$. We assume that 0 is the depot. Constraints 2 and 3 state that exactly one arc will be used to enter a node and exactly one arc will be used to leave the node associated with the customer. Constraint 4 depicts that the number of vehicles returning to the depot is K, the total number of vehicles. Constraint 5 states that the number of vehicles leaving the depot is the same as the previous one, K. Constraint 6 ensures that routes are connected and constraint 7 shows x is a binary integer variable that can be either 0 or 1.

The first and simplest solution to the problem is counting all possible routes and finding the shortest one. However, this job gets practically impossible after a small number of nodes and vehicles. Thus, the VRP is an NP-hard problem where exact algorithms can solve only small instances of problems. Heuristics remain the only solution method to the practical VRP examples [6].

3. Proposed Solution

A heuristic algorithm is proposed that reformulates the original VRP as a sequence of subproblems, each corresponding to a Travelling Salesman Problem (TSP) instance assigned to a single vehicle. This transformation is achieved by clustering customers into as many groups as there are vehicles. Each resulting TSP subproblem is then solved independently using the Simulated Annealing metaheuristic.

The core assumption underlying this approach is that the total travel distance can be reduced by partitioning customers such that each group is maximally distant—based on the given cost metric (e.g., distance or time)—from the others.

Accordingly, our algorithm consists of two main stages:

1. **Clustering The Customers** – partitioning the customers into groups based on the distance criteria,
2. **Route Optimization** – finding the shortest possible route within each group.

3.1. Clustering The Customers

During the clustering phase, to ensure that each group is as spatially distant from the others as possible, we first select K leaf nodes—where K is the number of vehicles. These leaf nodes are chosen such that they are maximally distant from both the depot and from each other. The selected nodes serve as the initial members of their respective groups. Subsequently, all remaining nodes (excluding the depot) are assigned to the group of the leaf node with which they are most directionally aligned. The algorithm for finding the leaf nodes and grouping the remaining nodes can be summarized step by step as in below.

Suppose, L is the set of nodes that will be selected as leaf nodes, $L \subset V$. Initially $L = \emptyset$ while $i \in V \backslash \{0\}$ is the candidate to be the leaf node.

$$i^* = arg c_{0i} \tag{3.1.1}$$

As the first step, we find the very first leaf node with the equation 3.1.1 that is the farthest from the depot. Then, to find the next leaf node, we calculate a heuristic distance H_{dist} value for each node that tells us how far it is from the previously selected leaf nodes. The node with the highest H_{dist}value becomes the next leaf node.

$$H_{dist}(i) = c_{0i} + c_{ij^*} \quad \text{where} \quad j^* = arg\ min_{j \in L} c_{ij} \tag{3.1.2}$$

With 3.1.2 the H_{dist} value for the i^{th} node can be calculated, where j^* is the nearest node among the previously selected leaf nodes to the i^{th} node. In brief, heuristic distance is measured as the sum of the distance of the node to the depot and the distance of the same node to the closest of the previously selected leaf nodes.

$$i^* = arg H_{dist}(i) \tag{3.1.3}$$

3.1.3 depicts the mathematical equation for selecting the next leaf node. After the selection, set of leaf nodes is update as $L \leftarrow L \cup \{i^*\}$. This operation, selecting the next leaf, is repeated until $|L| = K$.

$$i \in G_{j^*} \quad \text{where} \quad j^* = arg(c_{0i} + c_{ij} - c_{j0}) \tag{3.1.4}$$

As stated in 3.1.4, we assign the remaining nodes one by one except leaf nodes and depot to the same group as the leaf node they are most directionally aligned. The method we use to calculate this closeness is the sum of the distance from the node to the depot and the distance from the node to the corresponding leaf node minus the distance from the depot to the corresponding leaf node. With this step, all the nodes are assigned to a group. The basic idea behind this formula is to find approximately how far a node is to a direct line drawn from depot to the leaf node.

3.2. Route Optimization

After we complete clustering the customers into K (the number of vehicles) groups, we now have as many TSP problems as the number of vehicles. Here our approach is to use the *Simulated Annealing* optimization method to find the shortest route for each sub-TSP problem.

Simulated annealing (SA) is a general global minimum finding optimization method for solving combinatorial or other types of problems, based on the analogy of shaping iron by bringing it to a certain temperature and then cooling it [7].

SA has been shown to be one of the successful solution techniques in solving TSP in previous studies and comparisons with other solution techniques have been made by other researchers [8]. Therefore, we will use SA in this part of our solution. With SA, we will try to find the best solution by replacing the current solution with a neighbour solution with a certain probability in each iteration. We will use 0.99 cooling rate and initial temperature 1000.

4. Results

To test our solution, we created a python program that generates a synthetic data set of different sizes. To measure the success of the solution, we compared the results on the same data set with OR-Tools, an open source VRP library. For OR-Tools, we used the sample python code given in the documentation. To make the vehicle capacity variable insignificant, a high number was given for the vehicle maximum travel distance parameter, 100000, which is even higher than the total travel distance. By calling the *SetGlobalSpanCostCoefficient* method with a value of 100, we asked OR-Tools to minimize the longest route. This will make OR-Tools comparable with our solution.

The total distance of all routes will be mostly in our favour. While this is a success, we should keep in mind that OR-tools tries to minimize both the global total distance and the longest route. Therefore, the metric that we consider to be the most ambitious and successful is the time required for the solution. We will show how the solution time of OR-Tools increases dramatically as the problem size increases, making it inefficient for use in real-time applications. The trademark of our solution is that it generates very acceptable solutions in a very short time.

4.1 Generating Dataset

We used python programming language and NumPy library to create the dataset synthetically. We create a map of size x*y with a variable input value and create n points with random x y coordinates on this map with a second input value. After generating n random points, we create a distance matrix that holds the distances of these points between each other and to the depot.

In figure 4.1.1 we see the distribution of the locations of 8 randomly generated customers on the map. The green dot in the centre is the depot.

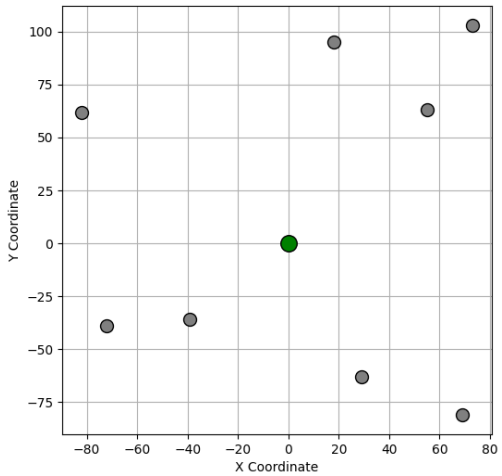

Fig. 4.1.1: Randomly generated 8 customer nodes.

4.2 Benchmark

For the random customers we created, we run both our solution and OR-Tools with a script we wrote in Python and plot the results with matplotlib. You can see a sample result for 32 customers and 4 vehicles in figure 4.2.1.

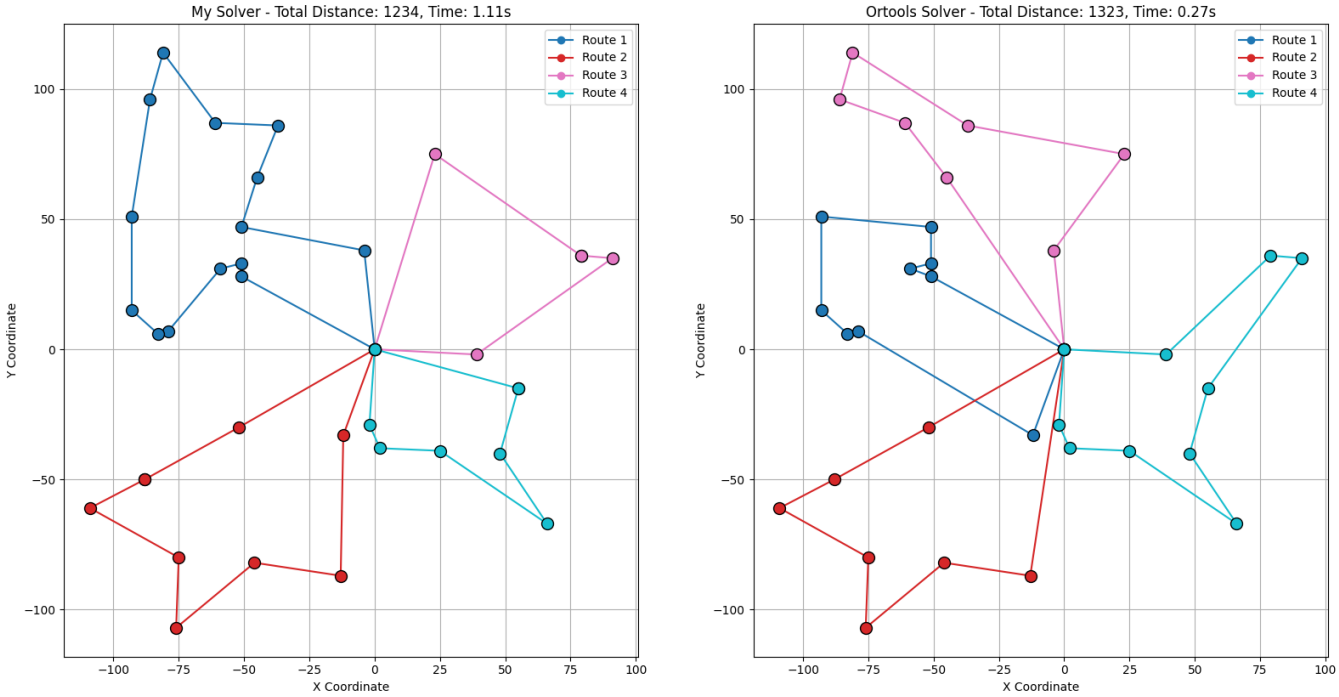

Figure 4.2.1: Results for 32 customers and 4 vehicles.

In addition to single tests, we ran both models 100 times each for different vehicle and customer numbers and listed the results. An important point here is that OR-Tools can produce solutions that do not use all vehicles, especially in problems with more than 6 and 7 vehicles. For the sake of comparison, we have omitted the tests in this case. Our solution is using all the vehicles provided. See discussions and future work section for a version of our solution that also optimises without using all vehicles.

Table 4.2.1: Results of 100 random tests for 8 customers.

Number of vehicles	Number of Customers	Average OR-Tools Finish Time (s)	Average My Solver Finish Time (s)	OR-Tools Win Count	My Solver Win Count	Deal Count
3	8	0.0094	0.3449	12	46	42
4	8	0.0106	0.3454	15	37	48
5	8	0.120	0.3463	15	34	51
6	8	0.0127	0.3540	19	26	55

As shown in Table 4.2.1, for 100 times, a dataset with 8 customers is generated and tested with a different number of vehicles between 3 and 6. In these tests, our solution performs noticeably better.

Table 4.2.2: Results of 100 random tests for 32 customers

Number of vehicles	Number of Customers	Average OR-Tools Finish Time (s)	Average My Solver Finish Time (s)	OR-Tools Win Count	My Solver Win Count	Deal Count
3	32	0.2100	0.5990	26	74	0
4	32	0.2464	0.7375	29	69	2
5	32	0.2687	0.8638	24	74	2
6	32	0.2996	1.0135	22	75	3
7	32	0.3284	1.1216	40	59	1
8	32	0.3897	1.2859	31	69	0

When we look at table 4.2.2, the first thing that stands out is again the dramatic win rate of our solution. The decrease in the draw rate is related to the increase in the number of customers.

Table 4.2.3: Results of 100 random tests for 64 customers

Number of vehicles	Number of Customers	Average OR-Tools Finish Time (s)	Average My Solver Finish Time (s)	OR-Tools Win Count	My Solver Win Count	Deal Count
3	64	2.6953	0.5990	26	74	0
4	64	3.1477	0.7375	29	69	2
5	64	3.3249	0.8638	24	74	2
6	64	3.0961	1.0135	22	75	3
7	64	2.4680	1.1216	40	59	1
8	64	2.6708	1.2859	31	69	0

Table 4.2.3 marks the first instance in which our solution achieves shorter computational times. Additionally, it continues to outperform the benchmark in terms of success rate in most cases. Notably, unlike previous tests, the cooling rate was adjusted to 0.997 in this set of experiments.

Table 4.2.4: Results of 10 random tests for 64 customers

Number of vehicles	Number of Customers	Average OR-Tools Finish Time (s)	Average My Solver Finish Time (s)	OR-Tools Win Count	My Solver Win Count	Deal Count
4	128	18.2591	2.1001	5	5	0
6	128	19.1405	2.0061	6	4	0
8	128	19.7737	2.2681	7	3	0

In table 4.2.4, we see that even with the high number of customers, our solution gives good results in almost one tenth of the time. To reveal the success of the solution in such a short time compared to the OR-Tools, we can look at the difference in the total travel distance result between the solutions. With the test where there are 4 vehicles, the average total distance result is 2456 for OR-Tools solution and 2471 for our solution, which makes the difference only %0.6. The difference percentage is %1.42. Our solution produces quite good routes in very short time, which makes it useful for real world problems where the time is crucial, especially for real-time applications. In addition to that, by tuning parameters in SA step,

we can still have more optimized routes in larger but still less time than OR-Tools. Table 4.25 illustrates the comparison when we increase the scale of the SA step.

Table 4.2.5: Results of 100 random tests for 128 customers

Number of vehicles	Number of Customers	Average OR-Tools Finish Time (s)	Average My Solver Finish Time (s)	OR-Tools Win Count	My Solver Win Count	Deal Count
4	128	20.1424	10.2873	13	87	0
6	128	20.9120	8.8524	33	67	0
8	128	21.3374	7.5373	54	45	1

5. Discussions and Future Work

For larger problems with 64 or more customers, we found that our solution performs well in terms of time and performs well in finding optimal routes. Although we are behind in time for small problems with fewer customers, we can improve the time by playing with parameters such as initial temperature, cooling rate and number of iterations in the simulated annealing step. Because in problems with a small number of customers, there is often no point in too many iterations.

Our problem is designed to solve the classical VRP where all vehicles will be used. However, it may be an interesting study for the future to create a version that does not need to use all the vehicles, can find a better result with fewer vehicles, and can work with constraints. In the same way, CVRP solutions can be produced by working on a version where the vehicle capacity parameter is added.

In addition, in the SA step we only performed an optimisation within the cluster. A future research topic for inter-cluster intermediate optimisation could be to see how SA works. Also, as an alternative to SA, Tabu Search, Genetic Algorithms or Ant Colonisation solutions can be tested. Because our solution has a modular structure that makes this kind of change possible.

5. Conclusion

An efficient direction-oriented method for solving the traditional Vehicle Routing Problem is presented in this research. The suggested approach performed competitively when compared to OR-Tools, a well-known open-source VRP package. The method provides significant practical benefits in real-world routing settings by focusing each vehicle on a unique subset of clients that are spatially segregated from those assigned to other vehicles. The approach is a suitable substitute for applications needing almost instantaneous route generation since it reliably generates high-quality routes in incredibly short calculation durations. One such instance is the planning of courier routes for metropolitan regions with constricted roadway networks and limited infrastructure, where directional clustering can be used to assign routes and improve transportation efficiency. Future research and real-world applications are made possible by the existing solution's modular design, which addresses the standard VRP formulation while facilitating adaption and extension to more specialized VRP versions.

Acknowledgements

This work was supported by the Scientific Research Projects Department of Istanbul Technical University. Project ID Number : 46803 and Project Code : MYL-2025-46803

References

[1] M. Fisher, “Chapter 1 Vehicle routing,” in Handbooks in operations research and management science, 1995, pp. 1–33.
[2] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” Management Science, vol. 6, no. 1, pp. 80–91, Oct. 1959.
[3] S. N. Kumar and R. Panneerselvam, “A survey on the vehicle routing problem and its variants,” Intelligent Information Management, vol. 04, no. 03, pp. 66–74, Jan. 2012.

[4] S.-Y. Tan and W.-C. Yeh, “The Vehicle Routing Problem: State-of-the-Art Classification and Review,” Applied Sciences, vol. 11, no. 21, p. 10295, Nov. 2021.
[5] F. Liu, C. Lu, L. Gui, Q. Zhang, X. Tong, and M. Yuan, “Heuristics for Vehicle Routing Problem: A survey and Recent advances,” arXiv (Cornell University), Jan. 2023.
[6] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, “New heuristics for the vehicle routing problem,” in Springer eBooks, 2005, pp. 279–297.
[7] F. Busetti, ‘Simulated annealing overview’, World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf, vol. 4, 2003.
[8] M. Panda, ‘Performance comparison of genetic algorithm, particle swarm optimization and simulated annealing applied to TSP’, International Journal of Applied Engineering Research, vol. 13, no. 9, pp. 6808–6816, 2018.