Comparative of algorithms for Solving the Capacity Vehicle Routing Problem

Jesus C. Carmona-Frausto[1], Adriana Mexicano-Santoyo*, Salvador Cervantes-Alvarez [2],
Pascual N. Montes-Dorantes[1], Francisco Arg¨uelles-Granados[1]
Technological Institute of City Victoria
Technological National of Mexico
Mexico

[2]University Center of the Valleys
University of Guadalajara, Mexico
*adriana.ms@cdvictoria.tecnm.mx

ABSTRACT: *This article shows a comparative study of five algorithms to solve the Capacity Vehicles Routing Problem (CVRP). The first two compared methods use the well-known k-Nearest Neighbor (kNN) algorithm; one of these searches for the Hamiltonian Cycle and then split the route into several subroutes according to the number of vehicles and customers, the other one assigns individual vehicles in order to obtain a route until the capacity of vehicle is exhausted and go back to the depot. The third of them is a Genetic algorithm with an improved cross-over operator to obtain better solutions. The four and five algorithms were Simulated Annealing and Tabu Search, respectively.The set of test instances used to compare the performance of the algorithms corresponds to the well-known set of instances used by the specialized community a proposed by Augerat in 1995. The genetic algorithm proves to find a shorter path and to be closer to the optimal values of the tested dataset, but on the contrary it takes a little longer. On the other hand, Tabu Search shows a similar behavior to Genetic algorithm but results were achieved in shortest time than the Genetic.*

Keywords: CVRP, kNN, Genetic-Algorithm, Simulated Annealing, Tabu Search

Received: 18 August 2022, Revised 30 November 2022, Accepted 7 December 2022

DOI: 10.6025/dspaial/2023/2/1/1-9

1. Introduction

Currently, the entire industry requires transporting and distributing the products it manufactures, which is why it requires finding solutions that ensure delivery by investing the minimum of resources, this problem is known as Vehicle Routing Problem (VRP)[7]. The formulation of VRP was proposed by Dantzig and Ramser in 1959 [9]. The VRP consists of finding the optimal route for one or more vehicles to deliver a product to a set of customers, considering the existence of one or more depots and several customers [1]. The VRP by itself does not consider the capacity of the vehicle [17], nonetheless, the Capacitated Vehicle Routing Problem variant incorporates a restriction directly related to the capacity of the vehicles, which plays a very important role in the distribution task. The CVRP consists of determining the routes that should be used to deliver customer orders, considering extracting the product from a depot and taking it to the location of each of the customers, for which the following restrictions must be considered a) the capacity of the vehicle, b) the demand of each customer is less than or equal to the capacity of the vehicle, since it is only visited once, c) the route begins and ends at the depot [7]. There are many applications of CVRP for industries that require logistics as a means for the supply, production and dispersion of products whose purpose is the reduction of distances, delivery times and consequently, the total costs associated with the distribution process [19]. The CVRP is an NP-hard problem, which means that the solution to the problem is complex and there are no exact algorithms to solve cases involving large numbers of customers [9]. Nowadays, it is known that exact methods obtain optimal solutions when are applied to solve small instances. Therefore, it is necessary the use of approximate paradigms as metaheuristics for solving real cases.

A plenty of methods have been developed in order to solve the CVRP. All the attempts trying to find the algorithm that offers the best results in all instances. Despite all the efforts, it is important to consider the advantages that the development of simple heuristics brings, against the complexity that the development of metaheuristics implies, which generally require more execution time.

This article shows the comparison of five methods to solve the CVRP problem where they are approximate algorithms based on the use of the kNN algorithm and the other three are the Genetic, Simulated Annealing and Tabu Search metaheuristic algorithms, which have shown to obtain good results. when solving optimization problems. The rest of the article is structured as follows: Section 2 presents the mathematical model that corresponds to the CVRP. Section 2 presents work related to solve the CVRP. Section 4 describes each one of the five evaluated algorithms. Section 5 describes the dataset used for experimentation. Section 6 presents the obtained experimental results. Finally, Section 6 shows the conclusion of this work.

2. Mathematical Model of CVRP

The CVRP belongs to the class of node routing problems, in which tasks are associated with nodes of the network. The CVRP can be formally defined as follows [20]: let $G = (V, E)$ be a complete undirected graph, where $V = \{v_0, v_1, \ldots, v_n\}$ corresponds to the vertex set. The depot is represented by vertex v0, which has m independent vehicles with identical capacity Q. Vertex $\{v_1, v_2, \ldots, v_n\}$ represents the n customers; $E = \{(v_i, v_j) \| i, j \in V, i = j\}$ is a edge set; a non negative distance matrix $C = c_{ij}$ between customers v_i to v_j is defined on E; each edge c_{ij} represents the distance between any two customers; d_i represents the demand of customer. In CVRP, vehicle $k(k \in M)$ starts and end at the depot. The total demand of any route cannot exceed the vehicle capacity limit Q, each customer is served by only one vehicle. The goal of CVRP is to determine a set of maximum m routes and keeps the total route to a minimum cost, considering the following:

Decision variable:

$$x^k_{ij} = \begin{cases} 1, \text{if vehicle k from customer } i \text{ to } j \\ 0, \text{Otherwise} \end{cases} \qquad (1)$$

Objective function:
1. minimizing the total distance

Constraints:
1. Each route starts and ends at the depot

2. The total demand of any route does not exceed the total vehicle capacity

3. Each customer is visited once by only one vehicle

According to the variables and parameters previously expressed, the problem can be expressed as follows:

$$min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=0}^{m} c_{ij} x_{ij}^{k} \qquad (2)$$

$$s.t. \sum_{k=1}^{m}\sum_{i=0}^{n} x_{ij}^{k} = 1, \forall j \in \{1, 2, \ldots, n\} \tag{3}$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ij}^{k} = 1, \forall i \in \{1, 2, \ldots, n\} \tag{4}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ij}^{k} d_i \leq Q, \forall k \in \{1, 2, \ldots, m\} \tag{5}$$

$$\sum_{j=1}^{n} x_{ij}^{k} = \sum_{j=1}^{n} x_{ji}^{k} \leq 1, i = 0, \forall k \in \{1, 2, \ldots, m\} \tag{6}$$

3. Related work

In this section some work devoted to solve the CVRP where the algorithms compared in this work were used for experimentation. In 2019 Rabbouch [14] proposed an algorithm based in the Simulated Annealing algorithm which has shown an effective technique for solving this problem. Nonetheless, he found that the Simulated Annealing algorithm is slow to converge and reach a global optimal value. Li et al. [10] proposed the generation of a hybrid algorithm mixing the benefits of the Simulated Annealing and Tabu Search algorithms, considering that the algorithms find good solutions in optimization cases. Wang and Lu [18] developed an algorithm based in the Genetic algorithm because it has proven to found good solutions for VRP. Wang an Lu affirm that one of the disadvantages of the Genetic algorithm is that it depends on the initial solution, to solve the disadvantage they apply heuristic techniques to explore in the space of solutions and obtain a good initial solution. Caballero et al. [4] proposed the development of an algorithm based on the Tabu Search, which is used to generate a set of initial solutions and through the application of the genetic algorithm it is sought that the solutions are improved.

Nazif and Lee [12] developed an algorithm based in the Genetic, which incorporates an improved crossover operator consisting of a complete bipartite graph. Perwira et al. [13] proposed a Simulated Annealing algorithm which was compared against the Nearest Neighbor finding approximated differences amongst 5 % between reported values. The author concludes mentioning that Simulated Annealing outperforms the Nearest Neighbor. Masudin et al. [11] proposed the comparison between the Nearest Neighbor and Tabu Search algorithms, finding that Tabu Search outperforms on 10 % the travel in comparison to Nearest Neighbor. Fitriani et al. [6] applied the Nearest Neighbor algorithm to solve a real CVRP case and compared it against other two heuristics. The results showed a better performance of Nearest Neighbor than the Sequential Insertion algorithm and a worse performance than the Saving Matrix algorithm. Kulkarni et al. [8] presented a comparative between the NN algorithm against the NN with capacity restrictions where is observed that the algorithm NN with capacity restrictions obtains a reduction in distance in comparison to the NN algorithm. Ramirez et. al [15] compared the Nearest Neighbor against Genetic algorithm when solving the CVRP, they found a better performance for the Nearest Neighbor because only two of ten cases were better solved by the Genetic and the performance was longer than the Nearest Neighbor.

4. Comparison of Methods for solving CVRP

In this work, five strategies have been implemented to solve the CVRP problem in order to evaluated the performance for findings solutions and the execution time required to find the solution. The first apply the wellknown k-Nearest Neighbor algorithm, the second corresponds to the k-Nearest Neighbor algorithm incorporating constrains of capacity, the third one is the Genetic algorithm, the four was Simulated Annealing algorithm, and the five was the Tabu Search. The representation of the information, detail of the algorithms and representation of the solutions is shown following.

4.1. K-Nearest Neighbor algorithm

In this method the k-Nearest Neighbor algorithm is applied to find the route, where the node nearest will be the next served node. The algorithm looks for the Hamiltonian Cycle and then split the route into several subroutes according to the number of vehicles and customers. The obtained route corresponds to the path conformed by a set of vehicles that satisfy the customer demands in several places (nodes). The first vehicle starts from the depot (0), serves the nodes according to the Hamiltonian cycle route until the capacity of vehicle is exhausted. After that the vehicle comes back to depot (0) and starts serving the balance demand of last served node and proceeds on route until all the nodes are served[16]. The algorithm considers the use of one vehicle at a time.

The algorithm of k-Nearest Neighbor is summarized as the following

1. Input the transportation matrix, demand vector, and capacity of vehicles.

2. Find the Hamiltonian circuit using nearest neighbor, starts at the Depot, visits all nodes, and returns to the Depot.

3. Computes the cost of the Hamiltonian circuit.

4. Computes the total demand

5. Computes the number of vehicles required by satisfy the total demand.

6. While (total demand > 0)

• Starts the path marked by the Hamiltonian circuit from the Depot until the capacity of vehicle is exhausted.

• The vehicle returns to the Depot.

• Computes the total cost incurred from start node (Depot) to end (Depot again).

• Accumulate the value of the cost of the trip with the previous.

• Update the route without considering the served nodes.

7. Output the results

4.2 k-Nearest Neighbor + Capacity constraint algorithm
This method uses Capacity constraint and Nearest Neighbor algorithms simultaneously. First, a vehicle leaves the depot (0) with its full capacity and distributes to the customers (nodes) that are closest to the depot until the product is finished. Subsequently, the route is recalculated with the nodes not served or partially served and the process is repeated until all clients are served [16].

The algorithm is summarized as following:

1. Input the transportation matrix, demand vector, capacity of vehicles.

2. While (total demand > 0)

• Starts the route with a vehicle using nearest neighbor, starts at the Depot, visits nodes until capacity exhausted, and returns to the Depot.

• The vehicle returns to the Depot.

• Computes the total cost incurred from start node (Depot) to end (Depot again).

• Accumulate the value of the cost of the trip with the previous.

• Update total demand with unserved nodes.

3. Output the results.

4.3 Genetic algorithm

The genetic algorithm bases its search on recombining a solution. Its name is born in imitation of the process of evolution and uses the idea of natural selection [19].

This algorithm works with a population of individuals that represent a feasible solution for a determined problem, likewise, each individual is assigned a value related to the goodness of said solution. If an individual is better suited to the problem, it will have a better chance of being selected for breeding by crossing with another individual that was selected in the same way to produce better offspring. Otherwise, if an individual fails to adapt to the problem, it will have a lower chance of being selected for reproduction. In this way, a new population of individuals is created (possible solutions) that replace the previous solutions where the propagation of the best solutions is favored during subsequent generations. In general the algorithm can be described as follows:

1. For the initial population, 500 individuals are considered to have a more extensive search for better solutions, based on the initial solution obtained with the nearest neighbor algorithm. The population of individuals is randomly generated.

2. A cross of individuals with a probability of 0.90 was used to generate offspring. For each pair of individuals selected to cross (parents) two children are generated with the elements of the first parent plus the elements found in the second parent. For this work, the simple point cross was implemented.

3. To avoid stagnation in local optimums, the mutation was applied with a probability of 0.1 for each of the elements (genes) of each of the individuals. The mutation was made by exchanging a pair of positions in the same individual.

4. After the crossover and mutation stages, each individual of the new generated population is evaluated and the individual with the best fitness is incorporated into the next generation (elitism).

5. Because the selective factor is important to reach a better solution [10], a roulette type selection is applied. With this selection, the set of individuals that will have the possibility of interbreeding to form better solutions was formed.

6. This process is repeated for a finite number of iterations in order to gradually improve the solutions. In this implementation the algorithm stops after 1000 generations.

4.4. Simulated Annealing algorithm

This algorithm simulates the evolution of an unstable physical system from thermodynamic equilibrium to a fixed temperature. In each cycle a new solution (x') is randomly selected from the neighborhoods of the current solution (x). This algorithm accepts new solutions according to two criteria: i) the value of the objective function of the new solution is better and ii) the value of the objective function of the new solution is worse, and a random value generated between zero and one is less than the difference between the current solution and the new solution divided by the system temperature T. The system temperature is set to a value T_0 at the start of the algorithm. This value decreases every n cycles proportionally to a cooling factor [5].

The initial temperature (T_i), final temperature (T_f) and cooling (F_f) constants were established as follows:

$T_i = 100$, $T_f = 0.01$ and $F_f = 0.999$. For each temperature, the number of iterations in the Metropolis cycle [55] was set to 100. The following steps represents the algorithm.

1. The initial solution was obtained by applying the kNN algorithm.

2. In each iteration of the Metropolis cycle, a new solution is generated by changing the order of the elements of the initial solution as mentioned in [54], selecting 2 elements at random and using the reversal, exchange or insertion operators.

3. If the new solution is better than the initial solution, this new solution will now be the initial solution in the next iteration. But if the new solution is worse, it is also taken into account, since the Simulated Annealing algorithm allows escaping local optima

by accepting worse solutions to expand its search in a global range, not just a local one [14]. The worst solutions are accepted as long as the Boltzmann probability $P = e^{-(x'-x)/T_i}$ is greater than a random value generated between zero and one. If the condition is not met, the initial solution remains unchanged and the full loop is started again for 100 iterations.

4. Once the 100 iterations of the Metropolis cycle have finished, the temperature is decreased and another cycle is started with the best solution found.

5. The algorithm stops until $(T_i) \geq (T_f)$

4.5. Tabu Search Algorithm

The Tabu Search algorithm explores the solution space by repeatedly moving from one solution to the best of its neighbors trying to avoid local optima. The main attributes of each visited solution are stored in a tabu list for a certain number of iterations to prevent these solutions from being revisited, that is, to avoid cycles in the search by environments [15]. The following steps represents the implementation.

1. The initial solution was obtained by applying the kNN algorithm.

2. For the generation of the neighborhood, the exchange operator is used, for which two elements are selected at random and later they are exchanged considering only the feasible solutions.

3. The best solution is selected and the best move is added to the tabu list for 2 iterations.

Dataset	n	K	Q	UB	Opt
A-n32-k5	31	5	100	784	yes
A-n33-k5	32	5	100	661	yes
A-n33-k6	32	6	100	742	yes
A-n34-k5	33	5	100	778	yes
A-n36-k5	35	5	100	799	yes
A-n37-k5	36	5	100	669	yes
A-n37-k6	36	6	100	949	yes
A-n38-k5	37	5	100	730	yes
A-n39-k5	38	5	100	822	yes
A-n39-k6	38	6	100	831	yes
A-n44-k6	43	6	100	937	yes
A-n45-k6	44	6	100	944	yes
A-n45-k7	44	7	100	1146	yes
A-n46-k7	45	7	100	914	yes
A-n48-k7	47	7	100	1073	yes
A-n53-k7	52	7	100	1010	yes
A-n54-k7	53	7	100	1167	yes
A-n55-k9	54	9	100	1073	yes
A-n60-k9	59	9	100	1354	yes
A-n61-k9	60	9	100	1034	yes
A-n62-k8	61	8	100	1288	yes
A-n63-k9	62	9	100	1616	yes
A-n63-k10	62	10	100	1314	yes
A-n64-k9	63	9	100	1401	yes
A-n65-k9	64	9	100	1174	yes
A-n69-k9	68	9	100	1159	yes
A-n80-k10	79	10	100	1763	yes

Table 1. Characteristics of the dataset

4. For this work, the objective aspiration criterion is considered, and it indicates that a tabu movement is admitted if the movement produces a better solution than the best solution obtained so far [3].

Based on the experiments carried out in the algorithm, on this aspiration criterion it is maintained that if a movement that is considered taboo improves 10% of the current solution, the restriction is eliminated so that this movement improves the current solution.

5. Dataset

The group of instances used to evaluate the performance of the algorithms consists of 27 instances, which have from 31 to 79 nodes and from 5 to 10 delivery vehicles. The capacities of the vehicles are fixed at 10 units. The Table 1 describes the characteristics of each of the 27 instances. The "Dataset" column refers to the name of the dataset, the n column refers to the number of nodes including the depot, the K column refers to the number of available vehicles, the Q column refers to the capacity of each one of the vehicles, the "UB" column is the best distance value found, and finally, the "Opt" column tells us if the UB found is the best of the set. This dataset is the one used by Augerat [2].

6. Experimental Results

All the algorithms were implemented in Matlab and were executed using a computer with Intel (core) i7 processor, 2.6 GHz, 16 GB RAM and the Windows 10 operative system. The Table 2 shows the obtained distances from the five algorithms after solving the 27 instances. The No. column corresponds to the number of instance, Dataset column refers to the name of the instance. The k - NN, k - NN + CC, GA, SA, and TS columns shows the distance obtained by the algorithms k-Nearest Neighbor, k-Nearest Neighbor + Capacity constraint, Genetic, Simulated Annealing and Tabu Search, respectively. For simplicity the values were truncated at the decimal point. In Table 2 is observed that the Genetic algorithm found the largest number of best solutions (in bold) in comparison against the other four. The second best algorithm was the Tabu Search reaching 10 best solutions of the 27 instances. It is important to stress that any of the algorithms found the optimal solution. The k-NN+CC algorithm was the algorithm that in general achieves the worst solutions (underlined) for the 27 tested instances.

Simulated Annealing was the metaheuristic that showed the worst performance regarding Genetic and Tabu Search algorithms.

According to the results shown in Table 3, we can see that the fastest algorithm is k-NN+CC since it takes the shortest time to choose a route, followed by the kNN, nonetheless, are the algorithms that obtain the largest distances in all the conducted. On the other hand the metaheuristics, which achieves better solutions spent large times, but they are reasonable times. In the case of the Genetic algorithm, the highest value achieved is slightly greater than 15 seconds and for Tabu Search, the largest spent time was of 4 seconds. Only the Simulated Annealing algorithm required very long times to converge, taking more than 200 seconds in some cases. Regarding four of the tested algorithms we found that time is not very relevant since in none of the compared methods the time is not exceed of 15 seconds to obtain a solution. However, we can see that as the number of nodes increases in the instances, the time spent grows mainly in metaheuristics, but always achieving a better route in comparison to the heuristics.

7. Conclusion

In this work, a comparison between five different algorithms that solve the CVRP problem, are presented.

The kNN algorithm has the advantage that from the beginning it knows the route to be followed until the end of the journey, as well as the number of vehicles to occupy. The k-NN+CC algorithm discovers the next node as soon as it attends the node in turn, making the discovery process gradually. However, the solutions found by the heuristic algorithms based on kNN were far from those found by the metaheuristics. On the other hand, the Genetic algorithm makes use of a heuristic to find a better solution through evolution by previous generations. The Simulated Annealing algorithm, despite trying not to fall into local optima, requires much more time to reach good solutions. Finally, it is observed that the Tabu Search easily manages to improve the solutions, in short times. Results shows that Genetic algorithm, in general offers a better performance, nonetheless, if the priority is the speed, Tabu Search was found the best solution amongst the compared algorithms in this work. In general, it is important to highlight that although metahuristic algorithms require a longer amount of time, they manage to reach better solutions than heuristics. According to the experiments carried out in this work, the time required by the Simulated Annealing and Genetic algorithms is relatively small since they do not exceed 20 seconds in any case, which reaffirms that they find solutions in short times. As future work, it is proposed to test other data sets with a greater number of clients, demands and vehicles with different capacities.

No.	Dataset	k-NN	k-NN+CC	GA	SA	TS
1	A-n32-k5	970	1212	**812**	813	848
2	A-n33-k5	742	837	**666**	715	703
3	A-n33-k6	742	929	**666**	822	804
4	A-n34-k5	971	918	**796**	801	841
5	A-n36-k5	952	1110	833	**823**	913
6	A-n37-k5	862	1042	**734**	823	813
7	A-n37-k6	1123	1135	1019	1009	**989**
8	A-n38-k5	851	875	815	**731**	814
9	A-n39-k5	1022	1074	887	**825**	875
10	A-n39-k6	972	1190	**850**	968	961
11	A-n44-k6	1121	1253	1057	**986**	1023
12	A-n45-k6	1160	1194	1013	1051	**937**
13	A-n45-k7	1790	1429	**1247**	1306	1271
14	A-n46-k7	1169	1379	**1011**	1052	1050
15	A-n48-k7	1418	1542	1152	1238	**1222**
16	A-n53-k7	1361	1306	1120	1150	**852**
17	A-n54-k7	1574	1379	1253	**1230**	1287
18	A-n55-k9	1224	1381	1124	1237	**1119**
19	A-n60-k9	1820	1796	1524	1516	**1492**
20	A-n61-k9	1832	1374	1161	**1087**	1112
21	A-n62-k8	1506	1767	**1406**	1453	1436
22	A-n63-k10	1561	1736	**1414**	1548	1424
23	A-n63-k9	1995	1974	1721	1679	**1605**
24	A-n64-k9	1789	1831	**1497**	1526	1629
25	A-n65-k9	1586	1427	1325	1351	**1253**
26	A-n69-k9	1846	1568	1299	1381	**1258**
27	A-n80-k10	2752	2268	2011	2069	**1990**

Table 2. Distance obtained of each of the evaluated algorithm for each tested instance

References

[1] Alesiani, F., Ermis, G., and Gkiotsalitis, K. *Constrained clustering for the capacitated vehicle routing problem (cc-cvrp). Applied Artificial Intelligence* 31, 1 (2022), 1–25.

[2] Augerat, P. Approche poly`edrale du probl`eme de tourn´ees de v´ehicules. Theses, Institut National Polytechnique de Grenoble - INPG, June 1995.

[3] Bozorg-Haddad, O., Solgi, M., and A., L. H. *Meta-heuristic and Evolutionary Algorithms for Engineering Optimization*. 2017.

[4] Caballero-Morales, S.-O., Mart´inez- Flores, J.-L., and S´anchez-Partida, D. An evolutive tabu-search metaheuristic approach for the capacitated vehicle routing problem. *In Management and Industrial Engineering* (2018), pp. 477–495.

[5] Escobar, J. W., and Rodrigo, L. Un algoritmo metaheur´istico basado en recocido simulado con espacio de b´usqueda

No.	Dataset	k-NN	k-NN+CC	GA	SA	TS
1	A-n32-k5	133.32	**2.75**	1275.56	67120.45	3070.08
2	A-n33-k5	138.10	**2.79**	1715.12	61741.78	4125.12
3	A-n33-k6	142.98	**2.82**	2133.43	70671.56	2945.14
4	A-n34-k5	147.82	**2.85**	2540.45	64744.45	3226.12
5	A-n36-k5	152.06	**2.90**	2960.82	63975.48	3680.78
6	A-n37-k5	155.85	**2.93**	3394.81	78350.56	3314.71
7	A-n37-k6	158.93	**2.96**	3830.20	74374.21	2976.35
8	A-n38-k5	164.14	**3.01**	4273.87	73774.45	3854.15
9	A-n39-k5	169.76	**3.04**	4719.96	75651.52	3614.46
10	A-n39-k6	173.75	**3.09**	5167.01	71280.14	2400.63
11	A-n44-k6	180.06	**3.12**	5615.42	112650.78	3470.41
12	A-n45-k6	187.32	**3.15**	6116.00	76186.45	4400.23
13	A-n45-k7	194.87	**3.18**	6651.21	85174.32	2612.32
14	A-n46-k7	197.23	**3.21**	7191.14	85850.87	2850.89
15	A-n48-k7	201.01	**3.25**	7732.76	94392.65	2891.64
16	A-n53-k7	206.20	**3.28**	8282.76	82876.45	3431.15
17	A-n54-k7	210.96	**3.31**	8865.95	106200.85	2932.78
18	A-n55-k9	215.12	**3.34**	9457.79	121334.56	4746.71
19	A-n60-k9	222.96	**3.37**	10065.78	129425.16	3116.33
20	A-n61-k9	227.56	**3.4**	10699.43	113220.70	3015.65
21	A-n62-k8	230.82	**3.45**	11338.35	124935.10	3480.55
22	A-n63-k10	238.28	**3.48**	1982.31	202470.78	3444.30
23	A-n63-k9	243.51	**3.51**	12638.96	189730.87	3671.64
24	A-n64-k9	251.31	**3.54**	13278.37	204864.69	3544.11
25	A-n65-k9	257.21	**3.57**	13874.04	121284.15	3192.84
26	A-n69-k9	261.12	**3.62**	14477.84	186870.24	3563.77
27	A-n80-k10	266.50	**3.65**	15175.37	173217.10	3988.12

Table 3. Execution time of each algorithm when each of tested instance is solved. Execution time in milliseconds

granular para el problema de localizaci´on y ruteo con restricciones de capacidad. Rev. Ing. Univ. Medell´in 11, 21 (2012), 139–150.

[6] Fitriani, N. A., Pratama, R. A., Zahro, S., Utomo, P. H., and Sri, M. T. Solving capacitated vehicle routing problem using saving matrix, sequential insertion, and nearest neighbor of producto x in grobogan district. In AIP Conference Proceedings 2326 (2021), pp. 0200071–1–0200071– 7.

[7] Jinsi, C., Peng, W., Siqing, S., and Huachao, D. A dynamic space reduction ant colony optimization for capacitated vehicle routing problem. *Soft Computing* 26, 1 (2022), 8745–8756.

[8] Kulkarni, S., Sohani, N., and Sehta, N. Capacitated vehicle routing using nearest neighbor algorithm in supply chain. *International Journal of Engineering Research & Technology* 3, 5 (2014), 1331–1334.

[9] Ilhan, l. A population based simulated annealing algorithm for capacitated vehicle routing problem. *Turkish Journal of Electrical Engineering & Computer Sciences* 28, 1 (2020), 1217–1235.

[10] Lin, S.-W., Lee, Z.-J., Yin, K.-C., and Lee, C.-Y. Applying hybrid metaheuristics for capacitated vehicle routing problem. *Expert. Syst. Appl.* 36, 1 (2009), 1505–1512.

[11] Masudin, I., Sadiyah, Risma F. andUtama, D. M., Restuputri, D. P., and Jie, F. Capacitated vehicle routing problems: Nearest neighbour vs. tabu search. *International Journal of Computer Theory and Engineering* 11, 4 (2019), 76–79.

[12] Nazif, H., and Lee, L. S. Optimised crossover genetic algorithm for capacitated vehicle routing problem. *Applied Mathematical Modelling* 36, 5 (2012), 2110–2117.

[13] Perwira Redi, A., Rohmatul Maula, F., Kumari, F., Utami Syaveyenda, N., Ruswandi, N., Uswatun Khasanah, A., and Chandra Kurniawan, A. Simulated annealing algorithm for solving the capacitated vehicle routing problem: a case study of pharmaceutical distribution. *Jurnal Sistem dan Manajemen Industri* 4, 1 (2020), 41–49.

[14] Rabbouch, B., Saˆadaoui, F., and Rafaa, M. Empirical-type simulated annealing for solving the capacitated vehicle routing problem. *J. Exp. Theor. Artif. Intell*. 31, 1 (2019), 1–16.

[15] Ram´irez-Cortes, I., Hern´andez-Aguilar, J. A., ´Angel Ruiz, M., Cruz-Chavez, M. A., and Arroyo-Figueroa, G. Design and implementation of a cvrp simulator using genetic algorithms. *Research in Computing Science* 147, 2 (2018), 35–47.

[16] Sanjay, K., Nagendra, S., and Neha, S. Capacitated vehicle routing using nearest neighbor algorithm in supply chain. *International Journal of Engineering Research & Technology* 3, 5 (2014).

[17] Santillan, J. H., Tapucar, S., Manliguez, C., and Calag, V. Cuckoo search via levy flights for the capacitated vehicle routing problem. *J Ind Eng Int* 14, 1 (2018), 293–304.

[18] Wang, C.-H., and Lu, J.-Z. A hybrid genetic algorithm that optimizes capacitated vehicle routing problems. *Expert. Syst. Appl.* 36, 2 (2009), 2921–2936.

[19] Yuanxiao, W., and Xiwe, L. A tight approximation algorithm for multi-vehicle cvrp with unsplittable demands on a line. *J Syst Sci Complex* 35, 1 (2022), 1902–1909.

[20] Yuelin, G., Hongguang, W., and Wanting, W. A hybrid ant colony optimization with fireworks algorithm to solve capacitated vehicle routing problem. *Applied Intelligence* (2022), 1–17.