Solving Capacitated Vehicle Routing Problem Using a Simulated Annealing and a Immune System

AmirAbbas RezaSoltani
610399205

1 Problem

The Capacitated Vehicle Routing Problem (CVRP) involves determining a set of routes, starting and ending at the depot v_0 , that cover a set of customers. Each customer has a given demand and is visited exactly once by exactly one vehicle. All vehicles have the same capacity and carry a single kind of commodity. No vehicle can service more customers than its capacity C permits. The objective is to minimize the total distance. Thus, the CVRP is reduced to partitioning the graph into m simple circuits where each circuit corresponds to a vehicle route with a minimum cost such that: (1) each circuit includes the depot vertex; (2) each vertex is visited by exactly one circuit; (3) the sum of the demands of the vertices by a circuit does not exceed the vehicle capacity C.
Formally, the CVRP can be defined as follows: Given a complete undirected graph G = (V, E) where $V = v_0, v_1, \ldots, v_n$ is a vertex set and

$$E = (vi, vj); vi, vj \in V, i < j$$

is an edge set. Vertex v_0 denotes the depot, and it is from where m identical vehicles of capacity C must serve all the cities or customers, represented by the set of n vertices $v_1, \ldots, v_n$. Each edge is associated with a distance c_{ij} , between customers v_i and v_j. Each customer v_i has non-negative demand of goods q_i. Let $V_1, \ldots, V_m$ be a partition of V, a route R_i is a permutation of the customers in V_i specifying the order of visiting them, starting and finishing at the depot v_0. The cost of a given route $R_i = v_{i0}, \ldots, v_{ik+1}$, where $v_{ij} \in V$ and $v_{i0} = v_{ik+1} = v_0$ (v_0 denotes the depot), is given by:

$$Cost(R_i) = \sum_{j=0}^{k} c_{j,j+1} \tag{1}$$

2 Simulated Annealing

2.1 Configuration Representation

In order to solve the CVRP problem, we propose the configuration. The representation is based on a vector where each cell $m \in v_1, v_2, v_3, \ldots, v_n$ corresponds to a customer and cell v_0 denotes the depot. Each route starts and ends at the depot. For example in Table 1, rout $< v_4, v_1 >$ is passed by first vehicle and rout $< v_2, v_3 >$ is passed by second vehicle. Order of customers is important in this representation.

v_0	v_4	v_1	v_0	v_2	v_3	v_0

Table 1: Configuration Representation

2.2 Initial Configuration

The initial solution is generated deterministically using a greedy algorithm that iterates over the list of cities and constructs initial non-optimal feasible routes based on a first-fit approach. The algorithm, proceeds as follows. If there are n customers $1, 2, 3, ..., n$, the algorithm assigns the first customer in the list to a new route as long as its demand doesn't violate the capacity constraint of a vehicle and sets the status of the node to be visited. The algorithm proceeds to the next customer in the list. If it encounters a customer demand that violates the capacity constraint, the customer is skipped and the following customer in the list is processed. If the route capacity is exceeded, then a new route is allocated and the algorithm repeats until all customers have been assigned to routes.

2.3 Neighborhood Transformation

2.3.1 Move

The move transformation finds five pairs of customers $< v_i, v_{i+1} >$ that have the shortest distances closest to each other including the depot. This is done by computing all distances between each pair of customers on all generated routes, including the distances to the depot. The transformation next selects five random customers that exclude the depot and the customers at positions v_{i+1}. The random customers are removed from their routes, and deterministically inserted into random routes. The transformation selects a random route and inserts the random customers in the routes based on the capacity constraint. Thus, for every random customer, a random route is selected and if the customer demand does not violate the route capacity it will be inserted in the new route. This method is applied in 20% of iterations.

2.3.2 Replace Highest Average

The replace highest average transformation calculates the average distance of every pair of customers in the graph. Thus, for each vertex v_i in a route, the method computes $d_i = (d_{i-1,i} + d_{i,i+1})/2$ and selects the five vertices with the largest average distances and removes them from their routes. The transformation picks next five random routes and inserts the five selected customers in the route with the resulting minimum cost. This method is applied in 80% of iterations.

2.4 Cost Function

The objective of the algorithm is to minimize the cost of all routes. Thus, the objective is to minimize the following problem cost function ($Cost(R_i)$ is defined in equation 1):

$$F(s) = \sum_{i=1}^{m} Cost(R_i) \tag{2}$$

2.5 Cooling Schedule

The cooling schedule is the set of parameters controlling the initial temperature, the stopping criterion, the temperature decrement between successive stages, and the number of iterations for each temperature. The cooling schedule was empirically determined. Thus, the initial temperature, T_{init} , was set to 5000 while the temperature reduction multiplier, α, was set to 0.99. The number of iterations, M , was determined to be 5. The algorithm stops when the temperature, is below 0.1.

2.6 CVRP Annealing Algorithm

The algorithm, starts by selecting an initial configuration and then a sequence of iterations is performed. In each iteration a new configuration is generated in the neighborhood of the original configuration by replacing vertices that have the largest distances to other vertices as well as by moving the vertices with the highest average distance to their neighbors. The solution is always feasible as the neighborhood transformations use a constructive approach that generates feasible routes. The variation in the cost functions, ΔC , is computed and if negative then the transition from C_i to C_{i+1} is accepted. If the cost function increases, the transition is accepted with a probability based on the Boltzmann distribution. This probability function is:

$$T(\Delta C) = e^{\frac{-\Delta C}{T_{current}}} \tag{3}$$

where $T_{current}$ is current temperature. The temperature is gradually decreased from a sufficiently high starting value, $T_{init} = 5000$, where almost every proposed transition, positive or negative, is accepted to a freezing temperature, $T_f = 0.1$, where no further changes occur.

2.7 Experimental Results

E-n51-k5 and E-n101-k8 benchmarks are used to test the algorithm, 5 times. The naming convention is, starting with series of benchmark following by number of customers and number of vehicles. So E-n51-k5 consists of 51 customers and 5 vehicles (with capacity of 160) and E-n101-k8 consists of 101 customers and 8 vehicles (with capacity of 200); And results are as follows:

Data set	Max cost	Min cost	Mean cost	Mean time (min)
E-n51-k5	802.36	701.71	756.1	8
E-n101-k8	1338.22	1191.15	1255.39	12

Table 2: Experimental Results of the Simulated Annealing Algorithm

3 Immune System

3.1 Configuration Representation

It is resembled simulate annealing counterpart.

3.2 Initial Configuration

It is also similar to simulated annealing; But in this algorithm, 5 configurations is produced. So for 5 times, random permutation in cities is applied; Then with greedy algorithm, each configuration is produced similar to mentioned the simulated annealing algorithm.

3.3 Cost Function

It is resembled simulate annealing counterpart.

3.4 Proliferation

In this algorithm, number of configurations in population is constant 5. So first sorting configurations based on their cost, is applied in descending order. Then up to configuration rank between 1 to 5, copies of original configuration is added to population. For example the configuration, is copied 5 times, second best configuration is copied 5 times, ..., 5th best configuration is copied just 1 time.

3.5 Mutation

After proliferation, mutation is applied on each new copy that is added to population. To perform mutation on a configuration, first a customer is selected, it is removed and added to a rout that does not violate capacity constraint.

Mutation is applied on each copied configuration S, with probability of $P(S)$, which R_S is rank of original configuration from which S is copied and:

$$P(S) = e^{-\frac{1}{9}R_S} \tag{4}$$

3.6 Select

As mentioned, number of configurations in population, must remain constant 5. So after applying Mutation, from 8 best configurations based on their cost, 5 configurations are selected randomly, to makeup next generation.

3.7 CVRP with Immune System

The algorithm starts with initializing population for 5 configurations. The sequence of iterations is applied; In each iteration, new configurations are produced by proliferation and mutation operations as mentioned earlier. The algorithm applies select operation as mentioned, to makeup new generation. Every produced configuration is feasible since Mutation checks if the generated configuration is feasible. This process stops when overall best answer doesn't change in 2000 consecutive iterations.

3.8 Experimental Results

E-n51-k5 and E-n101-k8 benchmarks are used to test the algorithm, 5 times, and results are as follows:

Data set	**Max cost**	**Min cost**	**Mean cost**	**Mean time (min)**
E-n51-k5	841.5	714.78	778.87	4
E-n101-k8	1220.08	918.53	1063.71	10

Table 3: Experimental Results of the Immune System Algorithm

4 Conclusion

To solve Capacitated Vehicle Routing Problem (CVRP), 2 algorithms was represented; The simulated annealing (propsed in attached article) and the immune system. From sections 2.7 and 3.8, the immune system algorithm has better result in cost and time in E-n101-k8 benchmark than the simulated annealing algorithm (implemented by article). But in E-n51-k5 benchmark, the simulated annealing algorithm has better results than the immune system algorithm.