Capacitated Electric Vehicle Routing Problem: Granting green deliveries

Bosse Bandowski
Universidad Eafit
Colombia
bbandowsk@eafit.edu.co

Manuela Zapata
Universidad Eafit
Colombia
mzapatag1@eafit.edu.co

Luis Bernardo Zuluaga
Universidad Eafit
Colombia
lbzuluagag@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

ABSTRACT

The present paper seeks to present two algorithms designed to come up with the best way to carry out a route through nodes, given real-life factors such as vehicles' fuel and a budget–in this case, time. In the first algorithm, the goal is to reach a feasible or naïve solution, considering the previously mentioned limitations and making sure that each route is a Hamiltonian Cycle. The second algorithm is the optimized version of the first, still considering all limitations but delivering a better solution that will benefit said budget. The algorithms we used are greedy algorithm (Closest Neighbor) and VND algorithm (Variable Neighborhood Descent) respectively. The results show that the VND algorithm approach is much faster, but has a higher memory consumption. However, given that time is the top priority, we have concluded that VND is the best algorithm to solve this problem.

Keywords: Vertices, Hamiltonian Cycle, Optimization, greedy algorithm, VND algorithm, complexity.

ACM Classification Keywords: CCS → Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → **Shortest paths**

1. INTRODUCTION

The origin of this problem can be tracked back to the 1800s, when Irish mathematician W.R. Hamilton and British mathematician Thomas Kirkman first formulated the Travelling Salesman Problem (TSP). However, it wasn't properly studied by mathematicians until the 1930s, when a group of scholars in Vienna and a professor from Harvard began to actually look for the different possible algorithms to solve said problem. The first solution, suggested by Karl Menger, was the obvious brute force algorithm, but he then observed the low optimality of the closest neighbor heuristics. As the years passed, the TSP's popularity increased, along with the possible solutions given by several mathematicians. Ever since then, the TSP has been widely discussed and researched. Nowadays, retail companies that offer delivery services often face this issue; how can the delivery man pan out his route in a way that is more efficient time-wise and fuel-wise, while ensuring that he visits every customer.

2. PROBLEM

The environmental crisis has overflowed into every industry, demanding environmentally friendly reforms that contribute to the rehabilitation of our planet, and technology has been our best ally in the search of upgrading all kinds of processes. In the case of the transport industry, that is one of the most harmful ones, there is a pretty feasible solution to minimize damage; electric vehicles. However, efficiency continues to be a priority, and electric vehicles tend to run slower and need to recharge more often, which means more expenses. How will a company with delivery services plan how its vehicles will visit clients in the most beneficial way?

Given a set of vertices (clients), we must find a way to visit each vertex once, and only once, in the most efficient way with our electric vehicles. Additional to the classic Hamiltonian Cycle per each route, there are several factors that simulate real-life situations, such charging stations for the traveling vehicles and a given budget on which we must run on. The problem is to design an algorithm that will deliver the best possible solution, taking care of our budget and our environment.

3. RELATED WORK

3.1 Ant Colony Optimization Algorithm

In the natural world, ants wander randomly, and upon finding food they return to their colony while laying down a pheromone trail. When the other ants find such path they are likely to travel through that path, however, if the path is too long, the pheromone trails start to evaporate. The longer the path, the more likely it is to be forgotten. On the other hand, a short path gets used more often, and thus the pheromone density becomes higher and the longer paths are forgotten. One solution to this problem is the ant colony system, which consists of traveling through the graph and leaving "pheromones" on each node, the amount of "pheromones" in each node is inversely proportional to the length of the route, at the end, the route with the most pheromones is the shortest.

3.2 Traveling Salesman Problem (TSP) or nearest neighbor

Given a list of cities and the distance between each pair of cities, what is the shortest possible route that visits each city and returns to the origin? There is no exact solution to this problem because the problem grows depending on the given

graph, its importance lies in the fact that many problems can be crafted like this one and if one efficient solution is found it can be applied to many problems. One solution to this problem is Simulated annealing, which considers some neighboring state s* of the current states, and probabilistically decides between moving the system to state s* or staying in-state s. These probabilities ultimately lead the system to move to states that imply lower energy loss. Typically, this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.

3.3 Branch and Bound Algorithm

The branch and bound algorithm consists on a systematic enumeration of candidate solutions, or a rooted tree, which expands as the algorithm runs through every possible path given by the graph's structure. Instead of generating all the possibilities (or branches) like in brute force, it checks if the current path has a heavier (more expensive) route than the "best" path so far. If yes, it stops expanding the branch and carries on to the following path. If not, the "best" path becomes the current path and it carries on to consider the remaining paths.

3.4 Genetic Algorithm

The Genetic algorithm is a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. It usually starts with randomly generated individuals, called a generation, then every individual is evaluated and mutated again to form a new generation. This iteration lasts until the maximum population has been produced or when it gets to a satisfactory result has been reached.

4. CLOSEST NEIGHBOR ALGORITHM

In this section we will explain the closest neighbor algorithm.

4.1 Data Structure

We used an ArrayList of ArrayList for the distance matrix, its length is the amount of nodes given in the data set and read by the program. The distance matrix is created in order to make a comparison of the distance to each node, decide which is closer and access the closest neighbor faster. The cells marked in red are the distance of a node to its self. We assigned that number as infinitely big so that the algorithm does not consider that as a possible transition.

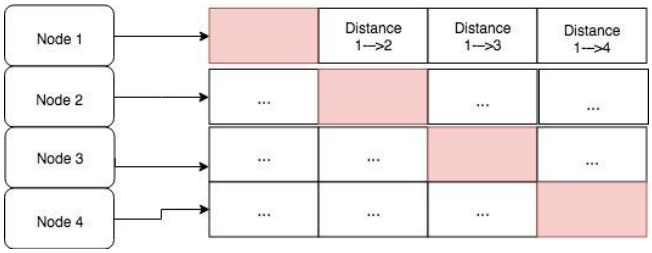

Figure 1: Data structure representation

4.2 Operations of the data structure

The data structure isn't meant to operate anything, it's just a storing mechanism to always have important data at hand.

4.3 Design criteria of the data structure

Given the structure of our project's development, we chose the distance matrix as the data structure for the main decision making in the algorithm because before we start with the VND algorithm, we need a starting point. This is where the closest neighbor algorithm comes into place; we must have an, although naïve, feasible solution before we can optimize it with the VND algorithm. Comparing each node with each other every time a vehicle needs to move is extremely expensive, so the distance matrix is meant to store all that data, making it so it only has to be calculated once.

4.4 Complexity analysis

Complexity for closest neighbor algorithm:

Algorithm	Part	Method	Complexity	Asymptotic Complexity
Closest Neighbor	setup	Run.checkInput	1	1
		Run.readInput	n+n	1
		Run.createDistanceMatrix	n^2	n^2
		Run.createStationMatrix	n^2s	n^2s
		Run.setImportantDistances	n	n
		Run.orderStationsByDistanceToBase	s^2	s^2
		Run.planReturnToBase	ns	ns
			1+(n+n)+n^2+n^2s+n+s^2+ns	n^2s
	planRoutes	Run.findGlobalClosestNeighbour	v	v
		Vehicle.canGoToCNInTime	1	1
		Vehicle.returnToBase	s	s
		Vehicle.canGoToCNWithFuel	1	1
		Vehicle.moveTo	1	1
		Vehicle.findClosestClient	n	n
		Run.WrapUpRoutes	ns	ns
			c(v+1+s+1+1+(vn)+ns)	cn(v+s)

Table 1: Table to report complexity analysis

4.5 Execution time

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Setup	0.2495s	0.2595s	0.31562s	0.3145s
CN	0.0214s	0.0272s	0.0309s	0.0367s

Table 2: Execution time for CN algorithm

4.6 Memory consumption

	Best case	Dataset 2	Dataset 3	Dataset 4
Setup	15 MB	15 MB	17 MB	17 MB
CN	111 MB	111 MB	108 MB	108 MB

Table 3: Memory consumption for CN algorithm

5. VARIABLE NEIGHBORHOOD DESCENT

In this section we will explain the variable closest neighborhood descent algorithm.

5.1 Data structure

Since the two algorithms (CN and VND) are joined together, they share the same data structure. Go to **4.1** for the complete explanation.

5.2 Operations of the data structure

Since the two algorithms (CN and VND) are joined together, they share the same data structure. Go to **4.2** for the complete explanation.

5.3 Design criteria of the data structure

Since the two algorithms (CN and VND) are joined together, they share the same data structure. Go to **4.3** for the complete explanation.

5.4 Complexity analysis

Complexity analysis for VND algorithm:

Algorithm	Part	Method	Complexity	Asymptotic Complexity
VND	setup	changeRouteRepresentation	n+v	n+v
			n+v	n+v
	optimize	calculateTimeBetweenTwoNodes	1	1
		sFeasibleSolution	n+v	n+v
		calculateObjectiveFunction	n+v	n+v
		move	1	1
		swap	1	1
		reconnect	n+v+n+v	n+v
			(n+v)(n+v)(n+3(n+v))	(n+v)^3

Table 4: Table to report complexity analysis

Total complexity of the program:

n^2s+cn(v+s)+n+v+((n+v)^3)*i

5.5 Execution time

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Setup	0.2495s	0.2595s	0.31562s	0.3145s
VND	0.0047s	0.0047s	0.0068s	0.0057s

Table 5: Execution time for VND algorithm

Memory consumption

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Setup	15 MB	15 MB	17 MB	17 MB
CN	111 MB	111 MB	108 MB	108 MB
VND	17 MB	17 MB	19 MB	19 MB

Table 6: Memory consumption for VND algorithm

6. CONCLUSIONS

The results show that the VND algorithm approach is much faster, but has a higher memory consumption. However, given that time is the top priority, we have concluded that VND is the best algorithm to solve this problem.

REFERENCES

1. Wikipedia, Travelling Salesman Problem history. https://en.wikipedia.org/wiki/Travelling_salesman_problem#History .
2. Gilbert Laporte.
   The Vehicle Routing Problem: An overview of exact and approximating algorithms
   Centre Recherche sur les Transportes, Montreal, 1991.
3. Allan Larsen, Oli B.G Madsen.
   The dynamic vehicle routing problem.
   Technical University of Denmark, Denmark, 2000.
4. Christian Nilsson.
   Heuristics for the Traveling Salesman Problem.
   Linköping University, Sweden.
5. Natallia Kokash.
   An introduction to heuristic algorithms.
   University of Trento, Italy.
6. Onne Beek and Birger Raa.
   Efficient Local Search Methods for Vehicle Routing.
   Gent Universiteit, Gent, Netherlands, 2011.
7. J.Fosin, T.Caric and E.Ivanjko.
   Vehicle Routing Optimization Using Multiple Local Search Improvements.
   Online ISN 1848-3380, Croatia, 2013
8. P. Hansen, N. Mladenovic, J.A.M.Pérez.
   Variable neighbourhood search: methods and applications.
   Springer-Verlag, DOI 10.1007/s10288-008-0089-1, 2008.
9. P.C.Pop and A. Horvat-Marc.
   Local search heuristics for the generalized vehicle routing problem.
   IACSIT Press, IPCSIT vol. 23, Singapore, 2012