Applying Ant Colony Optimization and Genetic Algorithms for Multi-Objective Vehicle Routing Problem

Abdullah Siddiqui
Dhanani School of Science and Engineering
Habib University
Karachi, Pakistan
ms03586@st.habib.edu.pk

Kuldeep Dileep Lohana
Dhanani School of Science and Engineering
Habib University
Karachi, Pakistan
kl04008@st.habib.edu.pk

***Abstract*—The capacitated vehicle routing problem (CVRP) is an NP-hard combinatorial optimization problem and a sub category of the larger array of vehicle routing problems (VRPs). In this paper, a genetic algorithm and an ant colony optimization algorithm are proposed to solve this problem. The proposed algorithms are tested on a series of benchmark instances and compared to the optimal recorded values as well as to each other to determine their performance. The results indicate that the algorithms have a satisfactory performance, especially for a small number of nodes. Visualization of the vehicle routing problem with a time window constraint is also explored in this paper to better allow comprehension of the problem and observe the effects of optimization.**

***Index Terms*—Capacitated vehicle routing problem, time window, genetic algorithm, ant colony optimization, visualization.**

I. Introduction

This paper aims to explore the use of genetic algorithms (GA) and ant colony optimization (ACO) algorithms in order to tackle the vehicle routing problem. In particular, we focus on the vehicle routing problem with the goal of minimizing total distance and number of vehicles/journeys required, while considering the capacity of the vehicles and a pre-specified time window to deliver the goods in, as our problem constraints.

The Vehicle Routing Problem (VRP) is a combinatorial optimization problem of the NP-hard category. It is a broad generalization of the better known Traveling Salesman Problem (TSP). There is a huge concern in the logistics and transportation industry where the goal of a given company is to transport goods and services to their customers in the most efficient way. This efficiency can be measured by a wide variety of metrics for example minimising the total distance taken, the time taken for a journey, the fuel costs, the number of riders or maximizing the amount of customers serviced by one rider etc.

There are many variants of the vehicle routing problem where different objective functions are optimized based on particular needs. For instance, one form of the VRP is Capacitated VRP (CVRP) where the constraint is applied on the capacity of the vehicle while making sure that maximum customers are served the capacity they require at minimum cost in terms of the distance traversed. Similarly, in Distance Constraint VRP (DCVRP), there is a constraint on only the distance that can be covered by the vehicle. Another category is VRP with Time Window (VRP-TW) where only a limited time is allowed to make the delivery and return back to the depot.

The focus of this paper is the capacitated vehicle routing problem, together with the time window constraint. The CVRP can be formally defined as a problem involving designing a set of routes for a fleet of identical vehicles with overall minimum route cost which service all the demands such that (1) all vehicles should begin and terminate at the central depot, (2) each customer should be visited exactly once, by exactly one route, (3) the total demand of the route does not exceed the capacity of the vehicle, and (4) split deliveries are not allowed [1].

One important point to note here is that the CVRP can be defined as either one vehicle making several trips sequentially to deliver all the goods, or multiple identical vehicles making single trips simultaneously to deliver all the goods. The optimal distance remains the same in either case. However, the single vehicle approach understandably takes more time to deliver all the goods but saves up on vehicular maintenance and fuel costs, while the multiple vehicle approach can deliver all the goods in a much faster fashion but at a higher operational cost. Both these approaches are very close together logically in terms of the algorithm implemented. For the initial testing, the focus of the problem formulation in this paper has been on using one vehicle to traverse all the customer nodes. Later on, when we are visualizing the problem and considering the time window constraint, multiple vehicles are introduced into the problem.

In this paper, we will discuss the ant colony optimization and genetic algorithms in section II of the paper and conduct a short literature review before moving on to our implementation of both the algorithms in section III. Sections IV and V detail the conducted experiments and obtained

results while section VI helps visualize the improved routing by our proposed algorithms. Finally, section VII concludes the paper with commenting on the obtained results and discussing how to move forward.

II. Background and Related Work

A. Genetic Algorithm

Genetic algorithms are a common optimization technique used to find the optimal solution to a computational problem by modelling natural evolution processes. A population of solutions is maintained over several generations and is gradually improved using selection protocols and the genetic processes of mutation and crossover. New individuals are created by combining members of the population via crossover to perform knowledge sharing. The resultant off-springs may be mutated further to enhance the exploitative nature of the algorithm. Survivors are then selected from the total population to form the next generation, and the process is repeated until optimal candidate solutions can be achieved and the termination criteria is met [2].

Extensive work has been carried out in literature to solve the capacitated vehicle routing problem using genetic algorithms such as in [1], [2], [3], [4], and [5]. Authors in [3] have proposed a GA based solution to solve VRP wth pickup and delivery of goods to and from customers while aiming to minimize travelled distance, number of drivers, time and total cost. In [4], a modified GA is proposed to solve VRP while employing parallelism techniques such as a local scatter search method to partially decompose the problem. [2] proposes a novel mutation method while [5] focuses on an improved route crossover method.

This paper takes inspiration from multiple papers in order to design a meta-heuristic approach to optimize the given problem. The mutation operation is inspired from the inversion and swap sequence specified in [2]. The crossover operator isn't inspired from any particular paper, but was designed after perusing multiple papers and designed to suit our particular genotype representation.

B. Ant Colony Optimization Algorithm

Ant colony optimization (ACO) mimics the behavior of ants foraging for food. Each ant in a colony essentially represents a candidate solution. In real life, ants exhibit several interesting behaviours which are copied in the ACO algorithm. For example, while ants are out looking for food, they lay down a chemical trail called pheromones. The concentration of pheromones on a path signifies its promise to reach a food source. A high concentration of pheromone trail will enhance the probability of future ants taking that route. If we consider the shortest path to the food source, we note that the ants return home the quickest from the food source via this path so the pheromone trail is strongest there. If more ants take the route, they lay down more pheromones. Hence, as time goes on, more and more ants will converge to find the shortest route to the food source. Another important aspect to the ACO is the pheromone evaporation rate i.e. the pheromone disappearing with time. The laying down of pheromone as well as its evaporation form a delicate balance of exploration and exploitation that combine to find the optimal solution for a given problem.

Extensive work has been carried out in literature to solve the capacitated vehicle routing problem using ant colony optimization such as in [6], [7], [8], [9], and [10]. Authors in [6], propose a hybrid ACO alogrithm named SACO, to solve CVRP, which is a combination of ACO and Simulated Annealing (SA), where SA prevents ACO from getting stuck at a local optima. Moreover, in order to improve the performance of their SACO and to reduce its computational time, the authors further extended their algorithm into a decomposed version called DSACO which uses a unique pheromone perturbation strategy to avoid pheromone stagnation. Authors in [7] propose a novel algorithm to solve CVRP which is a combination of hybrid ACO and a stochastic local search algorithm i.e. Pareto Local Search (PLS). Authors in [8] propose a two-stage hybrid Ant Colony System (ACS) algorithm to solve CVRP. The first stage of the proposed algorithm minimizes the number of vehicles by using a revised ACS algorithm with a randomized algorithm (RA), while the second stage minimizes the travel cost using Iterated Local Search (ILS). They also propose a unique way to update the pheromones by utilizing certain elitist ants, which has inspired the pheromone trail update in our ACO algorithm as well. In [9], authors have addressed a two-dimensional loading vehicle routing problem which involves loading of freight into the vehicles and the routing of vehicles such that the customer demands are met. They claim to have proposed a modified ACO algorithm that successfully combines two completely different objectives i.e. loading and routing into one. [10] proposes a robust multiple ant colony system in which multiple ant colonies work in parallel to generate multiple solutions to the VRP problem by incorporating various uncertainties such as weather, traffic jam etc.

In this work, we have utilized the conventional ACO algorithm as well as a sequential ACO technique in order to solve CVRP such that a shortest path is found to meet all customers' demands while also ensuring that the constraints on the capacity of the vehicle are not exceeded. The time window constraint is also explored towards the end of the paper.

III. Implementation

A. Genetic Algorithm

This section details the implementation of a genetic algorithm to obtain an optimal solution.

1) Setting Up the Problem: The first step taken by the algorithm is to calculate the minimum possible trips a vehicle has to make in order to service the customers while fulfilling

capacity requirements. This is simply obtained using the following formula:

$$n = \begin{cases} floor(\frac{c_t}{c_v}) + 1, & \textit{if } c_t \neq kc_v \textit{ for } k \ \in \mathbb{R} \\ \frac{c_t}{c_v}, & \text{otherwise} \end{cases}$$

where, n is the number of trips, c_t is the total capacity of the goods that need to be delivered, and c_v is the capacity of one vehicle.

2) Chromosome Representation: The genetic algorithm proposed utilizes a conventional representation of an individual candidate solution. Each individual is represented as an array of arrays where each sub-array represents the route taken by a vehicle during one trip out from, and back to the depot. One example of such a chromosome is as follows:

$$[[1, 3, 7, 9], [4, 6, 5, 2], [10, 8, 12, 11]] \tag{1}$$

The above is denoted as the condensed representation, which represents the following route:

$$[0, 1, 3, 7, 9, 0, 4, 6, 5, 2, 0, 10, 8, 12, 11, 0] \tag{2}$$

Here, we have 12 customer nodes that are being visited in 3 trips. 0 represents the depot that the vehicle starts from, and returns to in each trip.

3) Fitness Function: For the CVRP, the fitness of each candidate solution is determined by simply taking a sum of the euclidean distances between concurrent nodes that need to be travelled in a complete journey out from, and back to the depot. The reciprocal of this value is fed to the algorithm since the GA used is designed to maximize the fitness value.

For the CVRP with time window, the fitness function also processes an additional constraint of time as well as distance. The calculated fitness value is weighted to represent both the distance (70% weightage) and time (30% weightage) before the reciprocal is taken, such that maximizing the fitness minimizes both distance and time with greater focus being on distance minimization.

4) Crossover: Two parents are selected for reproduction and knowledge sharing according to the parent selection criteria. The crossover operation first converts the chromosome from the condensed state shown in (1) to the expanded state in (2). Then the two-point crossover operation is applied on the resultant sequence of nodes while ensuring that two depot nodes do not appear consecutively in the chromosome. The two-point crossover can easily be understood as follows. For two parents:

$$P = [1, 2, 3, \mathbf{4, 5, 6, 7}, 8, 9]$$
$$Q = [9, 3, 7, 8, 2, 6, 5, 1, 4]$$

when the crossover operation is applied between points 4 and 7 for parent P, we obtain the following off-spring:

$$R = [3, 8, 2, \mathbf{4, 5, 6, 7}, 1, 9]$$

Furthermore, safeguard conditions were implemented to ensure that the capacity and time taken for each route were within the specified constraints for the generated off-springs. If the constraints were not met, the off-spring was discarded and crossover was repeated stochastically to ensure that they were met.

5) Mutation: The mutation operation is applied to an individual stochastically if a randomly generated number meets the mutation rate criterion. During mutation, either the swapping or the inversion operation is conducted, again introducing stochasticity into the process. The swap operation consists of randomly selecting two sub-strings or single nodes of customers and exchanging them. The inversion operator reverses the visiting order of the customers between two randomly selected points [2].

Furthermore, safeguard conditions were implemented to ensure that no depot nodes appeared consecutively in the chromosome, and that the capacity and time taken for each route were within the specified constraints for the mutated candidate. If the constraints were not met, the mutated candidate was rejected and mutation was repeated stochastically to ensure that they were met.

6) Parent and Survivor Selection: Several different selection schemes were explored while testing the proposed algorithm, namely truncation, binary tournament, rank based selection, and fitness proportionate selection schemes. The best results for the proposed algorithm were obtained by using binary tournament selection for parent selection and truncation for survivor selection.

B. Ant Colony Optimization

An ACO algorithm is implemented where the ants in a colony represent alternative solutions and are actively searching the solution space for the optimal value.

1) Construction of Vehicle Routes: In each iteration, every ant in the colony behaves as a vehicle and traverses all the nodes keeping to the given constraints. Initially, each ant is assigned a random customer node to travel to from the depot because initially there is no pheromone trail. Subsequently, ants continue to search the solution space based on specific transition probabilities mentioned in the next subsection. Each ant in the ACO is represented in a very similar fashion to the chromosome representation for the genetic algorithm, as shown in (1) i.e. an array of sub-arrays, where each sub-array represents one trip out from the depot and back to the depot, and the entire array represents the complete journey to traverse all the nodes.

2) Transition Probability: An ant k at node i will travel to the node j within its feasible neighborhood N_i^k with probability P_{ij}^k given by the equation below [11]:

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k}[\tau_{ik}]^{\alpha}[\eta_{ik}]^{\beta}}$$

Where, η_{ij} is defined as desirability or visibility and is the inverse of the distance between node i and j, τ_{ij} denotes the pheromone concentration on the path connecting node i and j, and α and β are constants which control the influence of pheromone concentration and visibility on the probability of an ant selecting a path from node i to j, respectively. The ant will always return to the depot whenever the next node to be visited is not in its feasible neighborhood i.e. the capacity of the vehicle has been exceeded.

3) Fitness Function: For the CVRP using ACO, the fitness of each candidate solution stored in each ant is determined by simply taking a sum of the euclidean distances between concurrent nodes that need to be travelled in a complete journey out from, and back to the depot. This value is then used in the pheromone update equation to bias the algorithm towards routes with smaller cumulative distances. The goal of the ACO algorithm is to minimize the fitness value.

4) Pheromone Trail Update: Once all the ants have traversed through all the nodes and have formulated the possible solutions, the pheromone concentration on each trail is updated to share the knowledge with the ants of the next iterations. The pheromone is laid down by each ant on every traversed edge depending upon the objective value L_k and can be obtained as:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

where m is the total number of ants. The quantity of pheromone trail laid per unit length, $\Delta\tau_{ij}^{k}$, on the edge (i, j) by the k^{th} ant between time t and $t + n$ is given by:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \textit{if } k^{th} \textit{ ant uses edge (i,j) in tour} \\ 0, & \text{otherwise} \end{cases}$$

where Q is a constant and L_k is the tour length of the k^{th} ant.

We also consider an additional factor by allowing the best recorded solution to influence the pheromone update phenomenon. This is done in order to obtain faster convergence to the optimal solution. All the edges that are travelled in the best recorded solution so far, with a fitness value of L^*, are further emphasized as if σ ants (elitist ants) had used them [8]. One such elitist ant increases the trail intensity by an amount $\Delta\tau_{ij}^{*}$ such that:

$$\Delta\tau_{ij}^{*} = \begin{cases} \frac{1}{L^*}, & \textit{if edge (i,j) belongs to best solution} \\ 0, & \text{otherwise} \end{cases}$$

As discussed earlier, part of the pheromone laid down evaporates by a factor of $(1 - \rho)$ in order to allow the algorithm to continue to explore the solution search space. Mathematically, the update in the pheromone trail between edge i and j for each iteration can be given by:

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij} + \sigma\tau_{ij}^{*}$$

Where, ρ defines the amount of pheromone that persists on the trail such that $(1-\rho)$ is the pheromone evaporation rate.

5) Sequential Ant Colony Optimization: Sequential ACO algorithm is a variation on the common ACO technique to design the solution by traversing all the nodes within the given constraints as discussed in the previous subsections [12]. The main way it differs from the traditional ACO algorithm is that each ant helps to calculate the complete route by implementing only one trip out from, and back to the depot. Another ant then designs the second trip of the route. And so on till the entire route is complete. Each ant starts off with a different customer node as its first stop once it leaves the depot. Across different iterations, each trip forming the complete route is gradually improved to reach the optimal solution. Apart from this key difference in the nature of each ant and what they represent, the techniques to calculate transition probabilities and pheromone trail updates remain the same. In addition to the conventional ACO algorithm, we also implemented the sequential ACO algorithm and tested the benchmark instances against it.

IV. Experiments

The GA and ACO were tested using the Augerat A dataset as a benchmark. This dataset is a part of the three Augerat dataset proposed by Augerat et al. in [13] which has been a popular benchmark instance for VRPs since its introduction in 1995. The dataset can be accessed here. The proposed algorithms were tested for different benchmark instances where the total number of nodes and the minimum vehicles/journeys required varied from 32 to 80 and 5 to 10 respectively. The results obtained are shown in section V.

The genetic algorithm was tested on the aforementioned benchmarks with different configurations. A population of 1500 individuals with 850 off-springs being generated per generation, for a total of 300 generations was used to test for the first 6 datasets. A population of 9000 individuals with 5100 off-springs being generated per generation, for a total of 500 generations was used to test for the rest of the datasets. These configuration were found to be a suitable compromise between the results obtained and the time taken to generate results.

The ACO algorithm was tested on the aforementioned benchmarks to evaluate its performance where the number of ants was kept the same as the number of nodes, allowing each ant to start from a different node. The remaining hyperparameters i.e. α, β, ρ and Q were decided based on the values that produced the optimal fitness value. Figure 1 depicts the trend of α against the fitness value, from which we can infer that the minimum distance is achieved when α is about

2. Similarly, from figure 2 we can conclude that the optimal value of beta is between 9 and 10. Through experimenting on different configurations, the values of rho and Q were similarly determined to be 0.9 and 30 respectively.

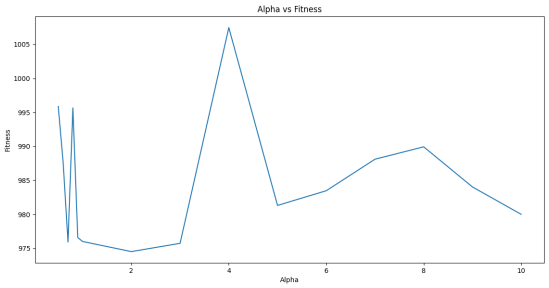

Fig. 1. Fitness versus Alpha

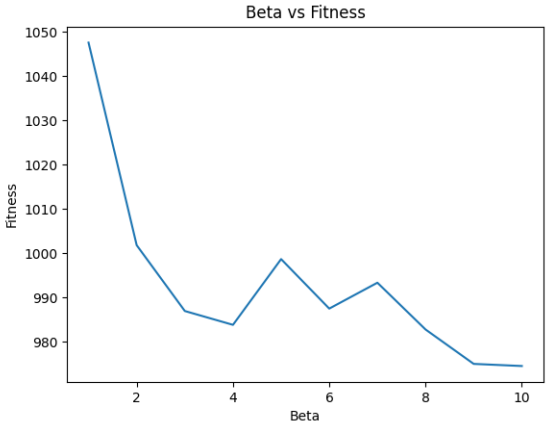

Fig. 2. Fitness versus Beta

V. Results and Discussion

A. Genetic Algorithm

The results obtained by testing the genetic algorithm on the Augerat A dataset can be viewed in table I. Note that as the number of nodes and trips increase from 32 & 5 to 80 & 10 respectively, the deviation from the optimal recorded value increases in a non-linear fashion. The deviation or the error is calculated by:

$$Deviation = \frac{\text{GA Result - Optimal Result}}{\text{Optimal Result}} \times 100$$

Note that for smaller number of nodes i.e. $(30-37)$ nodes, the error value obtained is less than or about 4%. The maximum recorded error was 24.8% for 80 nodes and 10 trips while the average deviation was 7.56% from the benchmark value.

This increasing error trend is understandable since the problem complexity grows i.e. number of nodes and trips increases. It has been tested and confirmed that a larger population will also successfully reduce the large error magnitudes found for the bigger datasets, but the time taken for those algorithm to process on our personal computers was too long to process the results for each and every dataset for a large population.

Figure 3 and figure 4 show the graphical evolution of the fitness values as generations progress in the genetic algorithm for the $A-n37-k5$ and $A-n69-k9$ data instances respectively. The orange curve represents the average fitness so far (ASF) and the blue curve represents the best fitness so far (BFS). The BFS and ASF curves tend to follow a monotonically decreasing pattern until reaching convergence. It can be observed that both curves tend to follow each other and eventually the average fitness and the best fitness converge onto a single value, even though best fitness is achieved earlier.

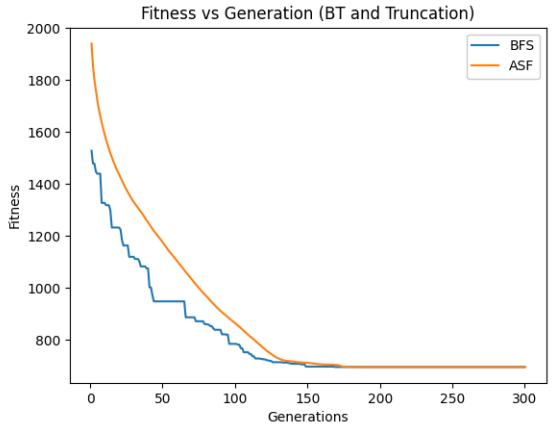

Fig. 3. Fitness versus Time for 37 nodes and 5 trips via GA

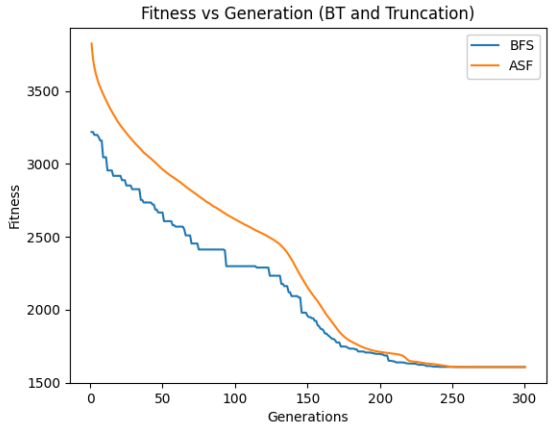

Fig. 4. Fitness versus Time for 69 nodes and 9 trips via GA

B. Ant Colony Optimization

The results for the ant colony optimization algorithm as tested on the Augerat A dataset can be observed in table I. As mentioned earlier, two varieties of ACO were implemented and tested. The first was the conventional ACO technique, and the second was a sequential approach hereafter referred to as Sequential ACO or Seq-ACO. The deviation or the error of

TABLE I
COMPUTATIONAL RESULTS FOR GA, CONVENTIONAL ACO AND SEQUENTIAL ACO TESTED ON THE AUGURAT A DATASET

Dataset	Benchmark Value	GA Result	Seq-ACO Result	Conventional ACO Result	GA Deviation	Seq-ACO Deviation	Conventional ACO Deviation
A-n32-k5	**784**	807	808	801	2.93%	3.06%	2.17%
A-n33-k5	**661**	681	756	677	2.98%	14.4%	2.42%
A-n34-k5	**778**	819	868	811	4.26%	5.98%	4.24%
A-n36-k5	**799**	827	994	826	3.52%	24.4%	3.38%
A-n37-k5	**669**	695	800	689	3.85%	19.6%	2.99%
A-n39-k5	**822**	870	899	861	5.80%	9.37%	4.74%
A-n44-k7	**937**	1001	1224	990	6.81%	30.6%	5.71%
A-n48-k7	**1073**	1158	1333	1140	7.92%	24.2%	6.24%
A-n55-k9	**1073**	1192	1379	1157	11.1%	28.5%	7.83%
A-n60-k9	**1408**	1447	1667	1429	2.80%	18.3%	1.49%
A-n69-k9	**1168**	1344	1604	1300	14.8%	37.3%	11.3%
A-n80-k10	**1764**	2201	2314	2033	24.8%	31.1%	15.2%

the value obtained from ACO from the benchmark value is calculated by:

$$Deviation = \frac{\text{ACO Result - Optimal Result}}{\text{Optimal Result}} \times 100$$

It appears that the conventional ACO outstrips the other two algorithms in terms of performance. Note that although the deviation from the optimal benchmark is still increases in a non-linear fashion as the number of nodes increases for the ACO, the deviation has reduced greatly when compared to the optimal values achieved by the genetic algorithm. The maximum deviation recorded is 15.2% for the largest dataset while the average deviation is about 5.64% from the optimal value.

Conversely, sequential ACO displays the poorest performance out of all three algorithms. It has a maximum deviation of 37.3% and an average deviation of 20.6% from the benchmark value. There is also no clear pattern of deviation increasing as dataset complexity increases since large deviations occurr at simpler datasets as well. This leads the authors to conclude that the sequential ACO method is quite unsuitable for implementation in a vehicle routing problem compared to the two other algorithms tested.

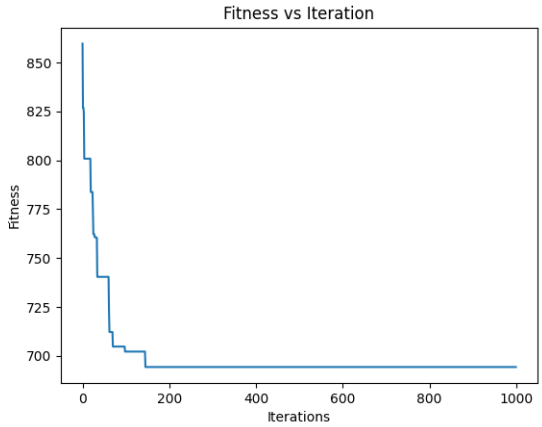

Fig. 5. Best Fitness versus Time for 37 nodes and 5 trips via ACO

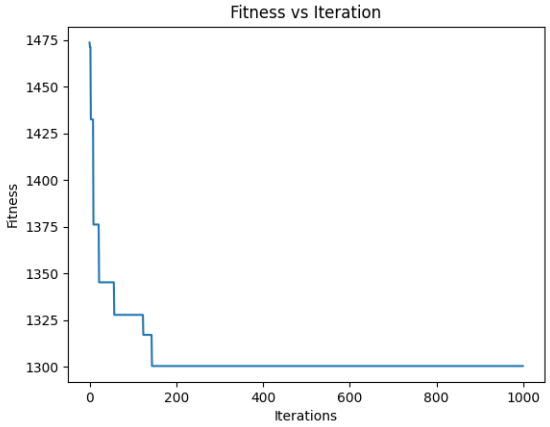

Fig. 6. Best Fitness versus Time for 69 nodes and 9 trips via ACO

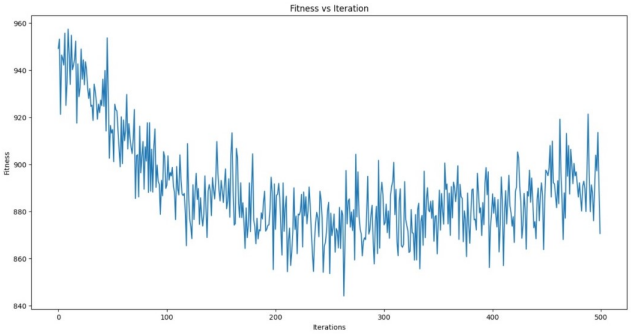

Fig. 7. Average Fitness versus Time for 37 nodes and 5 trips via ACO

Figure 5 and figure 6 show the graphical evolution of the fitness values as iterations progress in the conventional ant colony optimization algorithm for the $A-n37-k5$ and $A-n69-k9$ data instances respectively. The curve represents the best fitness values obtained so far. Observe that the algorithm reaches convergence at about 200 iterations even though it's allowed to run for 1000 iterations. The convergence is reached faster, and the final fitness value obtained is also better than

the counterpart results using the genetic algorithm shown in figure 3 and figure 4.

Figure 7 shows how the average fitness for the conventional ACO varies as iterations increase. It is important to note that although we do observe a decreasing pattern in the average fitness values, the curve obtained is very noisy and does not achieve the optimal best fitness value like its counter part for the GA. Over many different graphs, it was also noted that the average fitness curve tends to not show much improvement and moves sideways while the best fitness curve reaches the optimal value. This observation can be attributed to the fact that our pheromone update follows an elitist approach where the focus is on improving a subset of ants to obtain the optimal solution.

Figure 8 depicts the graphical evolution of the fitness value as iterations progress in the sequential ACO algorithm for the $A-n32-k5$ dataset. The curve shows the average and best fitness values obtained so far. Note that the average fitness curve does not achieve convergence and also takes significantly more time than its counterpart in the conventional ACO in order to minimize the distance value. This is undesirable behavior. The best fitness curve does appear to achieve a minimal value but the results achieved were nowhere as optimal when compared with the conventional ACO or even the GA.

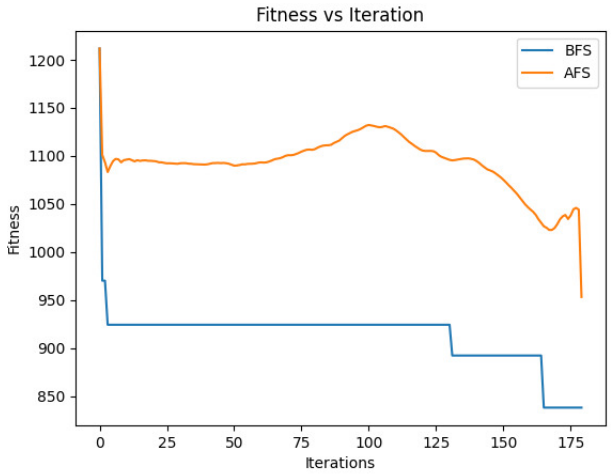

Fig. 8. Fitness versus Time for 32 nodes and 5 trips via Sequential ACO

VI. Visualization

One important part of optimization problems is the ability to be able to visualize the result, and the improvement that occurs due to optimization by your algorithm. In addition to working on improving the algorithms and fine tuning the hyperparameters to better match the benchmark instances, we also focused on being able to visualize the obtained improvement in the optimized distances and time values for the vehicle routing problem.

The Veroviz library available here was very helpful for this purpose. Veroviz utilizes the Folium library in Python to visualize geo-spatial data by plotting interactive maps using leaflets. It is also integrated with the Cesium library in JavaScript which can be used to view real-time animations on 3D maps of the globe with high precision.

One limitation faced with visualization of the improved routing by our algorithm was that we were unable to find adequate CVRP datasets that provided the longitude and latitude values that could be used for plotting the data onto a map. The attempts to convert the x and y coordinates to longitude and latitude values were unsuccessful as well since the data points obtained would either be very densely or very sparsely populated, and not necessarily next to any roads accessibly by a delivery truck.

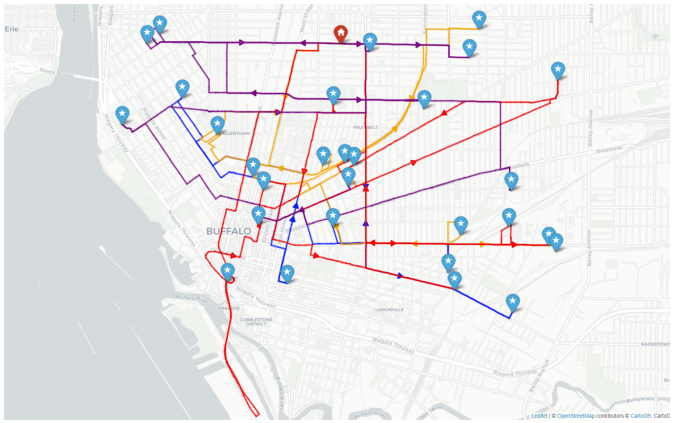

Fig. 9. Randomized Initial Route for the CVRP (Distance = $124km$)

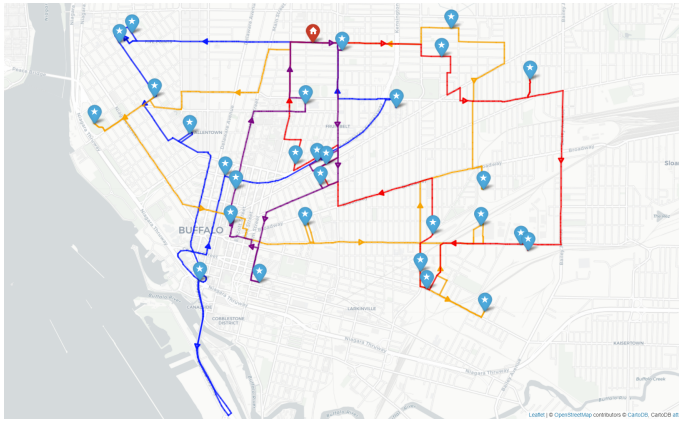

Fig. 10. Optimal Route Calculated for the CVRP by GA (Distance = $67.5km$)

As such, the visualization for the CVRP was carried out by randomly generating 29 customer nodes and 1 depot node in a specified area, next to roads accessible by a delivery truck. The capacity for each node and the vehicle was assigned using an Augerat A dataset of 29 nodes. The delivery truck was allowed to make 4 trips out from, and back to the depot in order to deliver goods to all the customers. The distance between all the nodes as well as time taken to cover said distance, incorporating the directions to be taken on real roads, were

obtained using the Veroviz library. The genetic algorithm was then used to determine the optimum sequence of nodes which minimized the total distance travelled by a vehicle.

Figure 9 shows a map displaying a route that was initialized randomly at the start of the optimization process. The total distance was noted to be $124km$ at this point. Figure 10 shows the same route after it has been optimized by the proposed genetic algorithm. The algorithm successfully decreases the total distance to almost half its initial value i.e. $67.5km$. The difference is also clearly visible in the diagram. Figure 9 shows a chaotic route with paths crisscrossing each other. Figure 10 shows a more ordered route with a tendency to visit all the nearest nodes before proceeding with the trip. In this way, we can clearly visualize the positive impact of our proposed algorithm on optimizing distances while fulfilling the capacity constraint.

Figure 11 shows an optimized route plotted using the aforementioned Cesium software. This representation displays an animation of the route on top of a 3D map of the specified coordinates. It can also produce a complete video which shows a delivery truck visiting all the customer nodes and depositing boxes i.e. 'goods' as required. This is very helpful in showcasing the time aspect of the problem. The complete video showing the animation of the process can be accessed here.

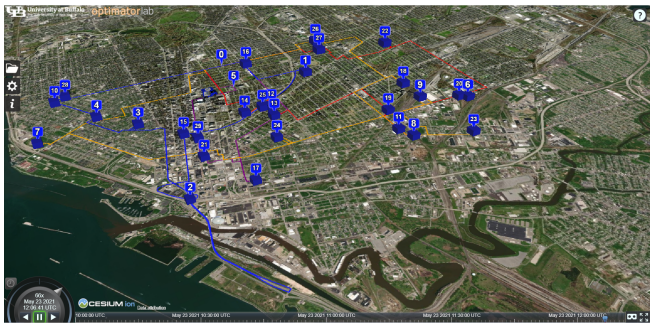

Fig. 11. Optimal Route 3D Visualization On a Map

As mentioned earlier, the capacitated vehicle routing problem was also explored with reference to a time window constraint. This was successfully implemented using 24 customer nodes and 4 vehicles.

First, distance was optimized over multiple iterations for 4 vehicles traversing the customer network and the average time taken was noted to be about 47 minutes, while the average minimum distance obtained was $61.57km$. The system was therefore required to optimize the time while keeping it below 40 minutes at all costs, as well as to minimize the distance as much as possible. The best value of time was obtained to be 37 minutes at a total distance covered of $71.6km$. Therefore, we obtained a reduction in time of 10 minutes while increasing the total distance covered by $10.03km$. Figure 12 shows a snapshot of the video recorded showing the side by side comparison of the CVRP problem on the L.H.S and the CVRP-TW problem on the R.H.S. The full video of the animation can be accessed here.

VII. Conclusion and Way Forward

The aim of this paper to explore the use of ant colony optimization and genetic algorithms for the vehicle routing problem was successful and we were able to obtain reasonable accuracy for two out of three algorithms upon testing them with the benchmark instances. We were also able to successfully visualize the vehicle routing problem on a map with actual routes instead of just relying on euclidean shortest-path distances. The use of Cesium software enabled us to showcase the resulting optimization in the time domain for a capacitated vehicle routing problem with time window as well.

However, there is plenty of work that can still be conducted in this domain. One interesting aspect we were unable to explore due to lack of time is to consider the problem in a local context for Pakistani food or grocery delivery services. Unlike their internationally renowned counterparts, Pakistani brands suffer from a disrepute on account of late deliveries of ordered goods. Testing out the algorithm on datasets made available locally would be very interesting and might help to narrow down the root cause of the delays.

There is plenty of work that can be used to further refine the ACO and GA proposed in this paper as well. The hyperparameters can always be tuned further, and the algorithm can be tested for larger populations and longer time periods to allow it to reach complete convergence. There are many different techniques for calculating transition probabilities & updating the pheromone trail for ACO algorithms and many different crossover and mutation operators for GAs available in literature that we were unable to adequately explore. Exploring and combining such techniques would help us better understand how to fine tune our algorithm so that it may be able to match or even beat the optimal recorded values on the benchmark data instances. An improved fitness function could be designed that would be even better suited to concisely represent the factors under consideration in CVRP-TW i.e. distance, time, and capacity.

The visualization of the data can also be further improved to allow for better comprehension, for example the animation using Cesium can be enhanced to a live-action display with a 3D model of the vehicle overlaid on top of the 3D map to make it more immersive.

References

[1] H. Awada, R. Elshaera, A. AbdElmo'ezab, and G. Nawaraa, "An effective genetic algorithm for capacitated vehicle routing problem," *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 2018.

[2] H. Nazir and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, vol. 6, 2012.

[3] G. Stacia, S. Nanang, and B. Syafri, "Vehicle routing problem with pick-up and deliveries using genetic algorithm in express delivery services," *AIP Conference Proceedings*, 2019.

[4] L. S. Ochi, D. S. Vianna, L. M. Drummond, and A. Victor, "A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet," *Future Generation Computer Systems*, vol. 14, 1998.

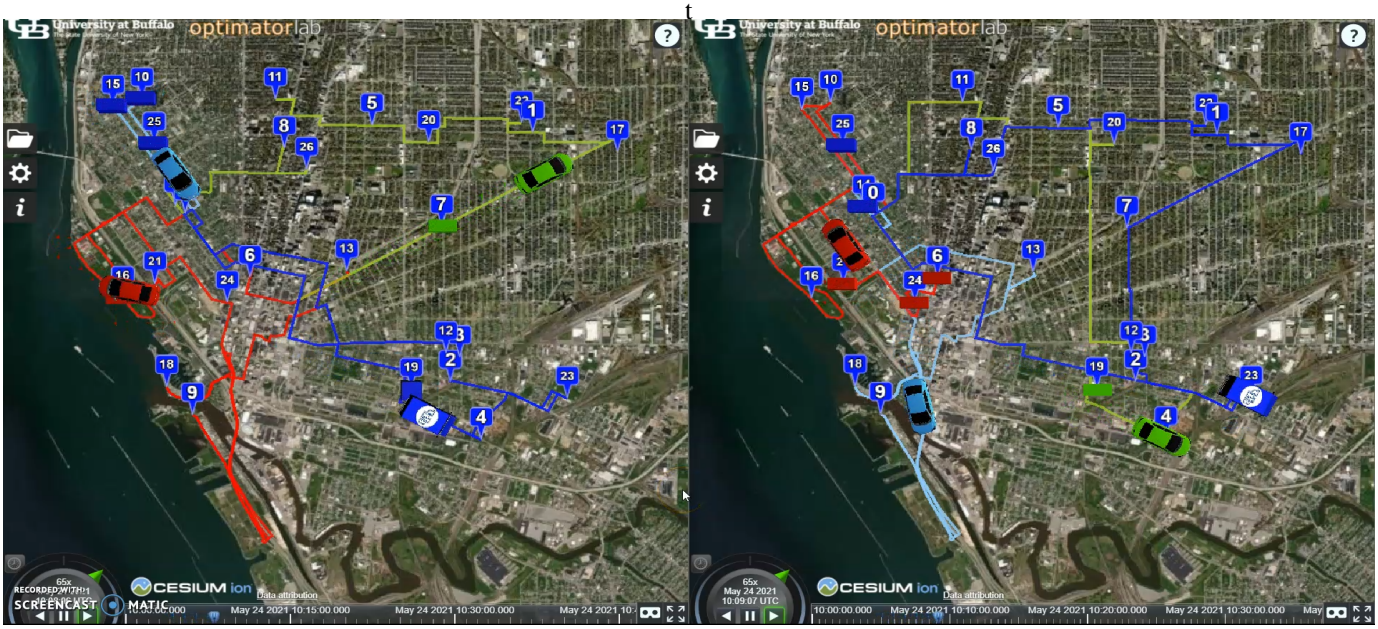

Fig. 12. Visualization of algorithm optimized for distance (L.H.S) and for both distance and time (R.H.S). Click here for Video.

[5] C. Ren, "Applying genetic algorithm for capacitated vehicle routing problem," *Proceedings of the Second International Conference on Electronic & Mechanical Engineering and Information Technology*, 2012.

[6] X. Sun, Y. Fu, and T. Liu, "A hybrid aco algorithm for capacitated vehicle routing problems," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp. 510–514, 2017.

[7] M. Xia, "A modified ant colony algorithm with local search for capacitated vehicle routing problem," in *2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA)*, vol. 2, pp. 84–87, 2009.

[8] C. Qi, S. Cui, and Y. Sun, "A two-stage hybrid ant colony algorithm for the cvrp," in *2008 International Conference on Computational Intelligence and Security*, vol. 2, pp. 215–219, 2008.

[9] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori, "Ant colony optimization for the two-dimensional loading vehicle routing problem," *Computers & Operations Research*, vol. 36, no. 3, pp. 655–673, 2009.

[10] N. Toklu, R. Montemanni, and L. Gambardella, "A robust multiple ant colony system for the capacitated vehicle routing problem," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1871–1876, 2013.

[11] W. Tan, L. S. Lee, Z. Abdul Majid, and H.-V. Seow, "Ant colony optimization for capacitated vehicle routing problem," *Journal of Computer Science*, vol. 8, pp. 846–852, 01 2012.

[12] S. Mazzeo and I. Loiseau, "An ant colony algorithm for the capacitated vehicle routing," *Electronic Notes in Discrete Mathematics*, vol. 18, pp. 181–186, 2004.

[13] P. Augerat, "Approche polyédrale du problème de tournées de véhicules," *PhD thesis, Institut National Polytechnique de Grenoble, France*, 1995.