A data-driven approach for solving route & fleet optimization problems

TECHNICAL WHITE PAPER

KARVY ANALYTICS

EXCELLERATING ANALYTICS

Authors

Charles Florin, PhD
Senior Director & Chief Data Scientist, Karvy Analytics Ltd.

Akhil Sakhardande
Senior Data Scientist, Karvy Analytics Ltd.

Shubham Sharma
Data Scientist, Karvy Analytics Ltd.

Vivek Jha
Data Science Intern- Undergraduate IIT Kharagpur, Karvy Analytics Ltd.

Rachit Jain
Data Science Intern - Undergraduate IIT Guwahati, Karvy Analytics Ltd.

Contents

1 Introduction to route and fleet optimization problems.....3
2 Challenges involved in traditional approaches..... 3
3 Our approaches.....4
3.1. Ant Colony Optimization (ACO) Algorithm.....5
3.1.1 Algorithm..... 5
3.1.2 Phases involved in the algorithm..... 7
3.2. Genetic Algorithm based approach 8
3.2.1 Making subsets using binary numbers 8
3.2.2 Application of Genetic Algorithm to Vehicle Routing Problem..... 8
3.2.3 The tree structure.....10
3.2.4 Objective Function..... 11
4 Inference.....11
4.1 Performance Comparison 11
4.2 Benefits.....11
4.3 Illustration.....11
5 Industrial applications.....13
6 References.....13

1. Introduction to route and fleet optimization problems

Organizations across all industries are facing the problem of route and fleet optimization to reduce their operational costs. The objective of this paper is to present algorithms that can be used to plan distance optimized routes to be followed by a delivery fleet. The fleet undertakes different kinds of works either at intra-city level or inter-city level following certain constraints. We assume that the fleet starts from a single starting node (the central office/industry) and returns back to the same place after visiting a set of nodes (house-holds/offices/warehouses), each of which is visited only by one of the fleet member.

The problem at hand can be understood by two well-known industrial applications, which are:
1. Multi-Vehicle Routing Problem
2. Multi-Travelling Salesman Problem

We study both the problems using different algorithms and make a fair comparison of their performance, running time and their ability to cater to constraints.

2. Challenges involved in traditional approaches

Route and fleet optimization problem is a NP-Hard problem in combinatorial optimization. The worst case running time for any algorithm to solve such a problem increases super-polynomially or rather say exponentially with the increase in number of nodes to be catered.The most direct solution using Brute Force Search will have a running time of O(n!) where n is the number of nodes under consideration while Held-Karp Algorithm (an application of Dynamic Programming) solves the problem in O(n^2.2^n). Similarly, other algorithms including 'branch-and-bound algorithms' and 'linear programming algorithms' exist but none of them is powerful enough to solve the problem in finite time for a large problem size.

3 Our Approaches

The problem at hand is similar to a constrained multi-Travelling Salesmen Problem (mTSP). It is a NP-hard problem and can't be solved using exact iterative methods in finite time as the problem size increases. There are several heuristics and meta-heuristics described in
literature that can be used to obtain approximate solutions to Travelling Salesman Problem (TSP) but only a few of them can be easily scaled up to be applied to constrained mTSP. In this paper we implement the following two meta-heuristic algorithms on different variations of route and fleet optimization problems, to find approximate but near optimal solutions to it.

1. Ant Colony Optimization to solve multi-Travelling Salesman problem
Ant Colony Optimization is a probabilistic technique that searches for an optimal path in the graph, based on the behavior of ants seeking a path between their colony and source of food [1].

2. Genetic Algorithm to solve multi-Vehicular Routing problem
Genetic Algorithm emulates the mechanics of natural selection by a process of randomized data exchange. The fact that they are able to search in a randomized, yet directed manner, allows them to reproduce some of the innovative capabilities of natural systems.

Both the above-mentioned techniques produce competent results, not the best, in terms of solution quality and time to reach the optimal solution.[6] But due to its modularity, multi-objective support and ease of applicability and scalability, we have chosen them to solve the constrained mTSP. These algorithms also use probabilistic selection rules, not deterministic and hence they're very efficient with inherently parallel distribution. We discuss both the algorithms in detail with respect to their use cases. Then we study the parameters used in both the algorithms followed by a performance comparison between both the algorithms. Finally, we list out the industrial applications of these algorithms.

We have implemented both the algorithms using the statistical tool R (open source) with RStudio as the Integrated Development Environment. The choice of language is not restricted and the same algorithms can also be implemented in other languages including Python and Matlab.

3.1 Ant Colony Optimization (ACO) Algorithm

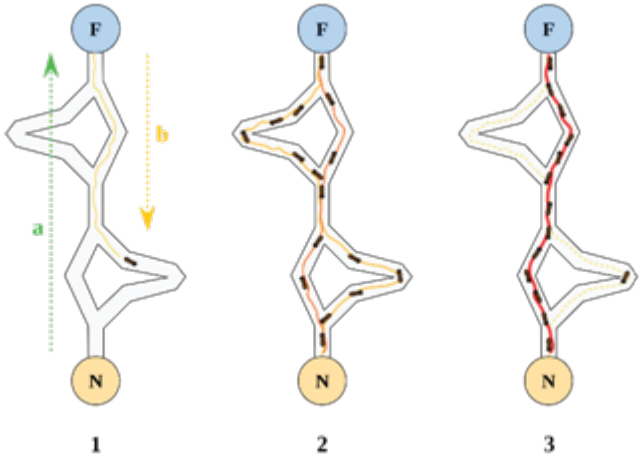


Figure 1

We know that ants wander randomly. But when they find a food source they walk back to their colonies leaving pheromones (markers) on the path, which shows that the path leads to a food source. When other ants come across the markers they are likely to follow these paths with certain probability. If they do, they then populate the path with their own markers as they bring the food back. As more ants find the path, it gets tronger until there are a couple streams of ants traveling to various food sources near the colony. Because the ants drop pheromones every time they bring food, shorter paths are more likely to be stronger, thus optimizing the "solution". In the meantime, some ants are still randomly scouting for closer food sources.[2][3][4] (See figure 1)

Algorithm

In ACO, artificial ants build a solution to a combinatorial optimization problem by traversing a fully connected construction graph shown in Figure 2 and defined as follows:

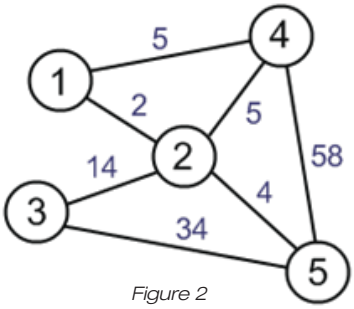


Figure 2

Consider a set of n nodes, the locations at which various jobs are to be done. These nodes can be represented in a form of weighted graphs as shown in figure 2, where each weight c_ij represents the distance between nodes i and j. Also assume that we have a delivery fleet of size m. Since only one of them visits a particular node we introduce m-1 artificial nodes whose location is same as node 1, the central/starting node (See figure 3). Additionally, assume that the pheromone level on each edge (i,j) is τ_ij.[7]

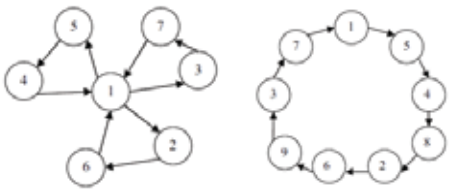
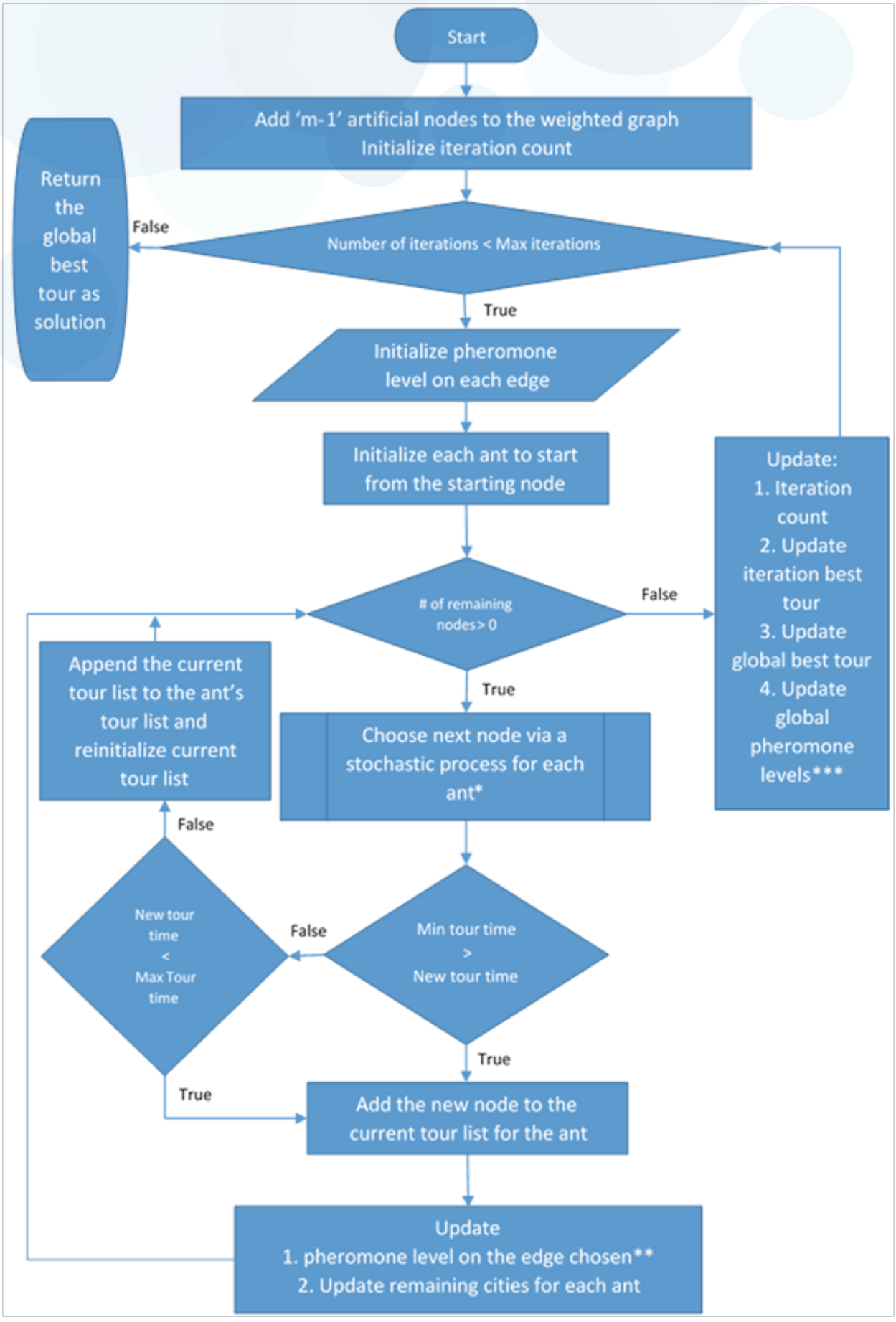


Figure 3

This converts our problem into a standard travelling salesman problem, which can be further solved using Ant Colony Optimization Technique.

The complete Ant Colony Optimization Algorithm is explained in Figure 4 using a flowchart and its certain phases are explained in detail on the next page.[5]


Start
Add 'm-1' artificial nodes to the weighted graph
Initialize iteration count
Number of iterations < Max iterations
False
Return the global best tour as solution
True
Initialize pheromone level on each edge
Initialize each ant to start from the starting node
# of remaining nodes > 0
False
Update:
1. Iteration count
2. Update iteration best tour
3. Update global best tour
4. Update global pheromone levels***
True
Choose next node via a stochastic process for each ant*
Append the current tour list to the ant's tour list and reinitialize current tour list
False
New tour time < Max Tour time
False
Min tour time > New tour time
True
True
Add the new node to the current tour list for the ant
Update
1. pheromone level on the edge chosen**
2. Update remaining cities for each ant

Phases involved in the algorithm

Construction of Ant Solution/ Selection of next node:
Ants in ACO use a pseudo-random proportional rule: the probability of an ant to move from node i to node j depends on a random variable q uniformly distributed over [0,1] and a parameter q_0.

If q<q_0 then amongst the feasible components, the component that maximizes the product α is chosen. This means that it is preferred to select an edge with higher pheromone level and shorter path length. Otherwise, we make a probabilistic decision using roulette wheel selection procedure, which defines probability of an ant moving from current node i to node j for each remaining city as

$$p(c_{ij}) = \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum \tau_{il}^{\alpha}.\eta_{il}^{\beta}} \quad ,for\ j\ \epsilon\ remainingcities$$

where η_{ij} is the inverse of distance between points i and j.

Local Pheromone Update:
The above-explained greedy approach favors the exploitation of pheromone information. It is counterbalanced by the introduction of local pheromone update which acts like a diversifying component to encourage subsequent ants to choose other edges to produce different solutions in one iteration. It is performed after each step taken by an ant on the last edge (i,j) traversed:

$$\tau_{ij} = (i - \varphi).\tau_{ij} + \varphi.\tau_0$$

Where (0,1] is the pheromone decay coefficient, and τ_0 is the initial value of pheromone which is the minimum level of pheromone present on each edge.

Global Pheromone Update:
After all ants have completed their tours in one iteration, we update the pheromone levels on all the paths using the following equation which signifies pheromone evaporation on all the paths:

$$\tau_{ij} = (1 - \rho).\tau_{ij},$$

whereρ is the global evaporation rate.

We also update the pheromone levels on the path followed by the best ant (the ant corresponding to the iteration best tour) using the formula:

$$\tau_{ij} = \rho.\Delta\tau_{ij}^{best}$$

where $\tau_{ij}^{best} = 1/L_{best}$ is the best ant used edge (i,j) in its tour and L_{best} is the length of the best tour found in the current iteration. This is done to ensure that the best solution of the current iteration is not lost while searching for better solutions.

3.2 Genetic Algorithm based approach

Before applying Genetic Algorithm to identify optimum routes, we slice the problem by forming smaller clusters. This technique helps us to reduce the complexity by approximately n times, n being the number of weighted clusters. We propose a new method of creating groups, by using binary numbers. Genetic Algorithm is then utilized to optimize the routes within each group [10][11]

Approach

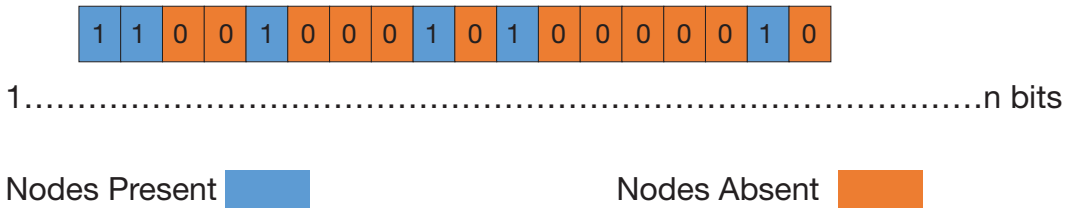
- Form groups within the given delivery nodes using binary number method (explained below). Consider the required constraint while forming groups.
  * Use genetic algorithm to find the optimal route within each group
  * For each group remove the nodes already catered.
  * Again, form groups using binary representation with reduced number of nodes. Find optimal outes for each group using genetic algorithm.
- Repeat until all nodes are catered. This gives a tree structure to the entire problem.
- Finally optimize the main objective function that contains cost and distance.
- Use Google Maps API to plot the routes to be taken.

One of the major advantages of using this method is that additional constraints can easily be added to this approach, we simply have to add the constraints while forming the groups and modify our objective function.

3.2.1 Making subsets using binary numbers

Suppose there are n nodes to be catered. All the integers between 1 to 2^n-1 are encoded into binary numbers, so as to get all the possible combinations of 0's and 1's of n bit length. Each bit corresponds to a node. Each n-bit binary representation is the route to be followed by a vehicle. The vehicle will serve the nodes corresponding to the bits having 1, and will skip the nodes with 0. Let the capacity of vehicle be c. For each binary representation, the demand at the nodes corresponding to bits having 1 is added. The binary representations whose sum of demands fulfill the constraint c _1<c<c_2 are taken and the rest are discarded (interval (c_1,c_2) allows for a margin in capacity of the vehicle). This gives us all the groups satisfying a constraint.

Binary representation



3.2.2 Application of Genetic Algorithm to Vehicle Routing Problem

GA works by generating a population of numeric vectors called chromosomes, each representing a possible solution to the problem. The individual components within a chromosome are called genes [12][16]

Chromosome encoding: This is the most basic step of GA. Here the problem variables are encoded into chromosomes. Permutation encodingis used for this problem. In permutation encoding, every chromosome is considered to be an array of numbers representing an ordered sequence.

Chromosome A:

3	5	7	1	4	6	8	2	9

The basic process has the following phases [13]

1. **Initialization** - Create an initial population. This population is usually randomly generated and can be any desired size, from only a few individuals to thousands.
2. **Evaluation** - Each chromosome of the population is then evaluated for a fitness value. This fitness value is a numerical representation of how well a chromosome fits with our desired requirements.The more the fitness value, better the chromosome fits our desired requirements. The fitness value in this problem is the inverse of distance travelled.
3. **Selection** - We want to be constantly improving our populations overall fitness. The basic idea is to make it more likely for the fitter individuals to be selected for our next generation. Roulette wheel selection method is used for this problem.
4. **Crossover** - During crossover, we create new individuals by combining aspects of our selected individuals. The expectation is that by combining certain traits from two or more individuals we will create an even 'fitter' offspring, which will inherit the best traits from each of its parents. Ordered crossover is used for this problem. In this crossover method we select a subset from the first parent, and then add that subset to the offspring. Any missing values are added to the offspring from the second parent in order that they are found.

   To make this explanation a little clearer, consider the following example.

1	2	3	4	5	6	7	8	9

9	8	7	6	5	4	3	2	1

Figure 5 - Parents

					6	7	8	

9	5	4	3	2	6	7	8	1

Figure 6 - Off-springs

   Here a subset of the route is taken from the first parent (6,7,8) and added to the offspring's route. Next, the missing route locations are adding in order from the second parent. The first location in the second parent's route is 9, which isn't in the offspring's route so it's added in the first available position. The next position in the parents' route is 8, which is in the offspring's route so it's skipped. This process continues until the offspring has no remaining empty values. If implemented correctly the end result should be a route which contains all of the positions, its parents did with no positions missing or duplicated.

5. **Mutation -** We need to add a little bit randomness into our populations' genetics otherwise every combination of solutions we can create would be in our initial population. Mutation typically works by making very small changes at random to an individual genome. We use swap mutation for this problem.

1	2	3	4	5	6	7	8	9

1	2	8	4	5	6	7	3	9

Figure 7 - Above gene shows the parent and the lower one shows the mutated offspring

With swap mutation, two locations in the route are selected at random then their positions are simply swapped. For example, if we apply swap mutation to the following list [1,2,3,4,5] we might end up with [1,2,5,4,3]. Here, positions 3 and 5 were switched creating a new list with exactly the same values, with a different order. Because swap mutation is only swapping pre-existing values, it will never create a list which has missing or duplicate values when compared to the original, and that's exactly what we want for the traveling salesman problem.

6. And repeat! - Now we have our next generation we can start again from step two until we reach a termination condition.

7. Termination - The GA search process typically continues until a pre-specified fitness value is reached, a set amount computing time passes or until no significant improvement occurs in the population for a given number of iterations. The key to find a good solution using a GA lies in developing a good chromosome representation of solutions to the problem.

3.2.3 The tree structure

After the route optimization is carried out for each subset, we arrange these subsets in ascending order of the distance covered. We then choose the best five subsets (it's taken to simplify the complexity of tree structure). The nodes in these five subsets are removed from the set containing all the nodes and the algorithm is repeated, considering only the non-catered nodes. This process is continued till every node is accommodated. (See Figure 8)

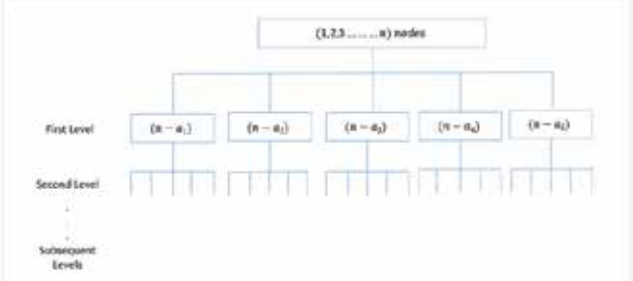


Figure 8 – Representation of tree structure formed

3.2.3 The tree structure

Let there be m vehicles with C_i = Cost/km for i^{th} vehicle.
The final objective function is:

$$Minimize, Z = C_1.d(a_1) + C_2.d(a_2) + C_3.d(a_3) + \ldots\ldots + C_m.d(a_m)$$

4 Inference

4.1 Performance Comparison

Ant Colony Optimization technique is able to generate optimal and near optimal solutions for difficult optimization problems in finite time, thereby adhering to the underlying constraints. On the other hand, Genetic Algorithm can be used to generate such solutions adhering to a lot more constraints, since it is more robust but takes more time as compared to Ant Colony Optimization. Refer to the performance table (Table 1). (Actual performance time of both the algorithms depends on various other parameters and constraints that have been used)

Table 1 - Performance comparison

Number of Nodes	Execution time of Ant Colony Optimization in seconds			Execution time of Genetic Algorithm based approach in seconds (*)		
	User	System	Elapsed	User	System	Elapsed
10	2.53	0.45	3.73	5.67	0.52	6.78
20	5.47	0.68	7.25	976.52	42.54	1098.46
50	14.46	1.39	18.57	-	-	-
100	31.15	1.88	37.45	-	-	-
150	64.17	2.45	72.84	-	-	-

*:Grouping takes around 95% of the total running time.

4.2 Benefits

Both the algorithms applied are inherently parallel and possess multi-objective problem capabilities, which can be used to find efficient solutions for TSP and related problems. On one hand Ant Colony Optimization technique uses 'positive feedback' for rapid discovery of good solutions, while on the other hand Genetic Algorithm has an ability to avoid being trapped in local optimal solution and produce solutions which get better with time. Both the algorithms are good for optimization in noisy environments and always produce an answer. These algorithms when implemented can increase efficiency by 25%, which can potentially decrease the expenditure of a firm by significant amounts.

4.3 Illustration

The following image (Figure 9) is a screenshot of one of the test cases. We used the R-Shiny application to develop a user interface and performed a test to calculate the optimized root for 60 delivery nodes across a city. Also see (Figure 10) to see the paths followed by the salesmen.

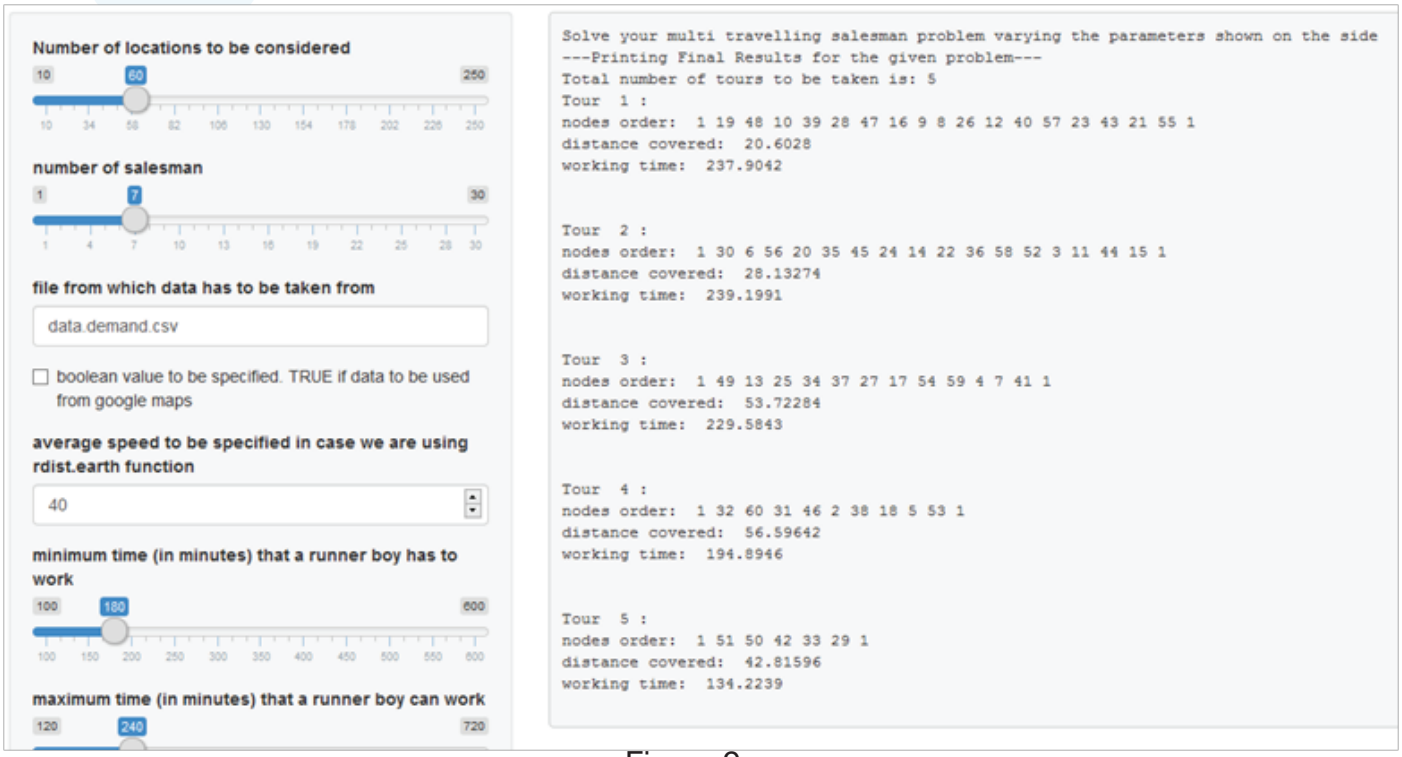


Figure 9

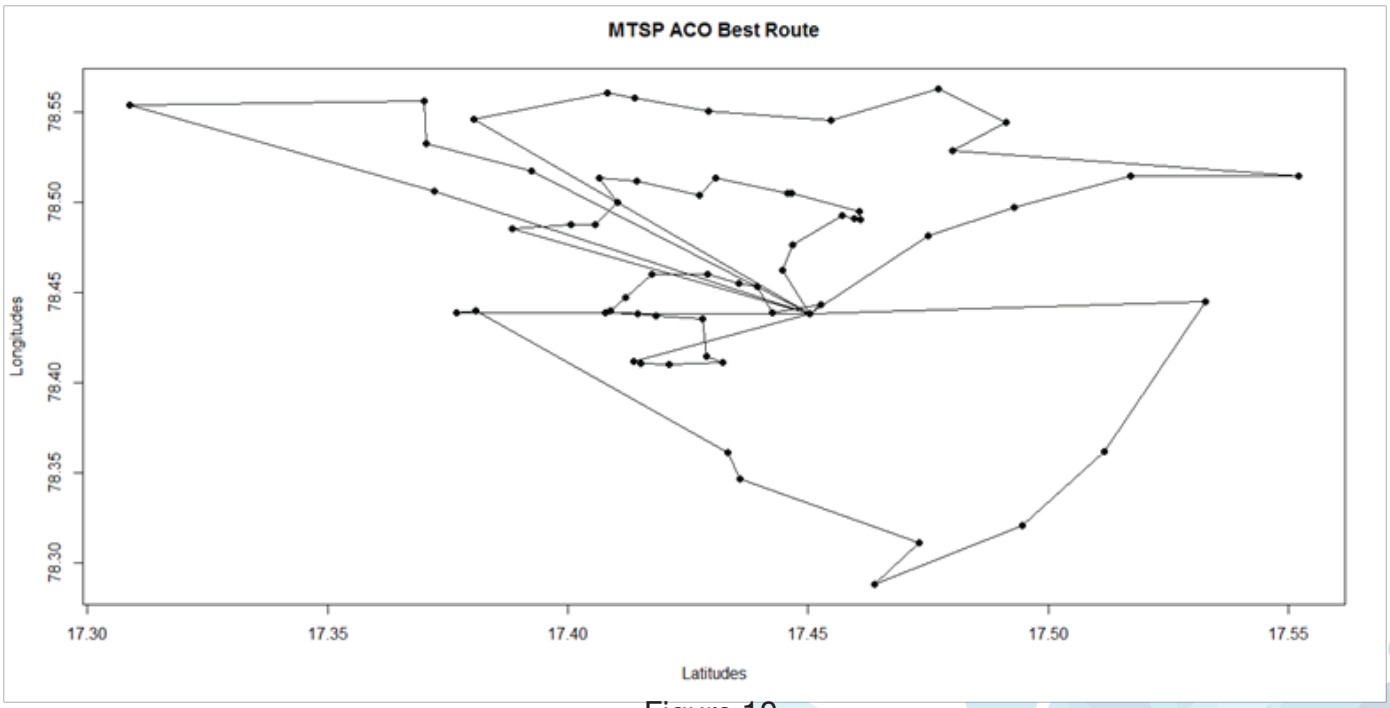


Figure 10

5. Industrial applications

ACO and Genetic Algorithm can be applied to almost all types of routing and scheduling problems, by just varying the constraints at hand. This is mainly because solutions to these problems are based on finding a local maxima or minima without actually iterating through all possible combinations. Some of the applications of the algorithms are:

1. Travelling Salesman Problem and its variations
2. Vehicle Routing Problem and its variations
3. Field engineer job-shop scheduling problem [14] [15]
4. Open-shop scheduling problem [16]
5. Permutation flow shop problem [17][18]
6. Single machine total tardiness problem [19][20]
7. Quadratic assignment problem [21][22]

For an online home services organization connecting home service professionals to customers an efficient solution to a constrained mTSP can be very useful to schedule visit of home service professionals to their customers. The entire process can be handled in a time and cost efficient way, resulting in effective management and cost cutting.

Ant Colony Optimization can further be used to schedule the servicemen, which can be treated similar to travelling salesmen. Servicemen can be divided based on what all-different jobs they undertake so as to schedule the right serviceman for the right problem. Using a fair estimate of how much time a serviceman takes at a customers' place (this parameter can be improved with time), we can calculate the optimal routes to be followed by servicemen. We can even use company data to forecast future demand volumes that can help the organization be prepared with a larger fleet of servicemen when need arises.

6. References

1. Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), 53-66.

2. Di Caro, G., &Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 9, 317-365.

3.Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1), 29-41.

4. Gutjahr, W. J. (2000). A graph-based ant system and its convergence. Future generation computer systems, 16(8), 873-888.

5. Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega, 34(3), 209-219.

6. Antosiewicz, M., Koloch, G., &Kamiński, B. (2013). Choice of best possible metaheuristic algorithm for the travelling salesman problem with limited computational time: quality, uncertainty and speed. Journal of Theoretical and Applied Computer Science, 7(1), 46-55.

7. Junjie, P., &Dingwei, W. (2006, August). An ant colony optimization algorithm for multiple travelling salesman problem. In First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06) (Vol. 1, pp. 210-213). IEEE.

8. Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., &Parthiban, P. (2010). Optimization of multiple vehicle routing problems using approximation algorithms. arXiv preprint arXiv:1001.4197.

9. Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

10. Ahmed, Z. H. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. International Journal of Biometrics & Bioinformatics (IJBB), 3(6), 96.

11. Potvin, J. Y. (1996). Genetic algorithms for the traveling salesman problem. Annals of Operations Research, 63(3), 337-370.

12. Vaira, G., DZEMYDA, G., AUGUTIS, J., BARAUSKAS, R., KAKLAUSKAS, A., ŽILINSKAS, J., ... & Kurasova, O. (2014). Genetic Algorithm for Vehicle Routing Problem (Doctoral dissertation, Vilniaus universities).

13. Applying a genetic algorithm to travelling salesman problem (http://www.theprojectspot.com/-tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5).

14. Cheng, R., Gen, M., &Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. Computers& industrial engineering, 30(4), 983-997.

15. Merkle, D., Middendorf, M., &Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. IEEE transactions on evolutionary computation, 6 (4), 333-346.

16. Liaw, C. F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research, 124(1), 28-42.

17. Reeves, C. R. (1995). A genetic algorithm for flow shop sequencing. Computers & operations research, 22(1), 5-13.

18. Stützle, T. (1998, September). An ant approach to the flow shop problem. In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98) (Vol. 3, pp. 1560-1564).

19. Bauer, A., Bullnheimer, B., Hartl, R. F., & Strauss, C. (1999). An ant colony optimization approach for the single machine total tardiness problem. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 2). IEEE.

20. França, P. M., Mendes, A., &Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. European Journal of Operational Research, 132(1), 224-242.

21. Tate, D. M., & Smith, A. E. (1995). A genetic approach to the quadratic assignment problem. Computers & Operations Research, 22(1), 73-83.

22. Gambardella, L. M., Taillard, É. D., & Dorigo, M. (1999). Ant colonies for the quadratic assignment problem. Journal of the operational research society, 50 (2), 167-176.