Ant Colony Optimization for Autonomous Delivery Robots (2021)

<Name Redacted>, <Name Redacted>, and Matthew J. Zaksek, *The University of Texas at Dallas*

***Abstract*— Autonomous robots are widely used for the delivery of products to consumers, notably for food deliveries at The University of Texas at Dallas, as well as other universities across the nation. However, as with any delivery system, inefficient transportation routes lead to unnecessarily increases in the time and costs associated with operating such systems. Therefore, autonomous delivery systems can reduce costs and make faster deliveries simply by using more optimal routes. This paper will present a method for delivery route optimization using ant colony optimization with the goal of minimizing costs associated with delivering goods.**

***Index Terms*—Ant Colony Algorithm, Autonomous Delivery Robots, Travelling Salesman Problem, Vehicle Routing**

I. INTRODUCTION

There has been increasing interest in deploying small, largely autonomous delivery robots for short deliveries to consumers. Some applications under study include the possibility of using such bots within hospitals to deliver drugs to patients [1]. Many students and faculty at The University of Texas at Dallas would likely be more familiar with the autonomous robots that currently function on campus, delivering food, drinks and snacks [2].

Regardless of the particular application, most autonomous delivery bots are powered by batteries with limited energy supply. Therefore, in order to minimize travel distance, travel time and energy costs, it is critical that bots choose the most efficient routes to make their deliveries. For bots making single deliveries, the primary focus is minimizing the route on a grid in order to minimize the travel distance from the starting point to the destination [3]. However, for bots that must make multiple stops in a single trip before returning to the starting point, another key consideration is which order the bot will visit each destination in order to minimize the overall travel route, as visiting the destinations in varying orders will result in varying overall travel distances. It is this second consideration that will be the primary focus of this paper.

II. LITERATURE REVIEW

The ant colony algorithm is a heuristic approach for solving optimization problems. It was introduced by Marco Dorigo in 1992 [7]. The ant colony algorithm is based on ants' ability to find the shortest route from their colony to a food source and back [4]. This is achieved by using pheromone trails, the ants mark the paths taken by depositing pheromones on the ground as they move. At the initial stage the ants move at random in search of food, but subsequent ants can detect the pheromones left by the previous ants and follow it reinforcing the trail by depositing its own pheromones on the same path [6]. The shorter paths will be reinforced as more ants can cover the distance in the same time compared to other paths. The other paths are neglected, and the pheromones will evaporate leaving a pheromone trail only on the shortest path.

The ant colony optimization algorithm uses "artificial ants" to search for the optimal solution for the problem being solved [6]. These artificial ants have memory, relying on information from the previous iteration [5]. The ants decide on the next location to move to using a probability function that dictates the possible paths that can be taken from its current location. At every iteration t, an ant k, can move into any available direction, in this case from a starting position x, towards a new position y. The probability function is

$$P_{xy} = \frac{\tau_{xy}^{\alpha} \cdot \eta_{xy}^{\beta}}{\sum_{z \in alllowed} \tau_{xz}^{\alpha} \cdot \eta_{xz}^{\beta}} \quad (1)$$

The probability function relies on are the pheromone function τ, and the attractiveness η. The attractiveness is given by the inverse of the distance, so as the distance from one neighboring point to the next is increased the attractiveness function value decreases and vice versa. This implies that the closer the next possible location the more likely the ant would travel in that direction. α and β are constant parameters, α controls the weight of the pheromone function and β controls the weight of the attractiveness.

$$\eta_{xy} = \frac{1}{distance_{xy}} \quad (2)$$

The pheromone function is meant to model the pheromones left by each "ant" on each iteration. After each individual iteration an updating function revises the value of the

This paper was submitted as part of a final course project for Engineering Optimization at The University of Texas at Dallas on December 6, 2021.

<Name Redacted> is affiliated with The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: <Email Redacted>).

<Name Redacted> is affiliated with The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: <Email Redacted>).

Matthew J. Zaksek is affiliated with The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: matthew.zaksek@utdallas.edu).

The entire MATLAB code mentioned in this paper has been submitted to Blackboard alongside this document according to the course requirements.

pheromone function, this function is modeled as

$$\tau_{xy} = (1-\rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^{k} \qquad (3)$$

Where ρ is the evaporation rate of the pheromones. So for every iteration the formula will decrease the old pheromones and increase the amount of pheromones left by the next "ant". The last term in the equation is given by

$$\Delta\tau_{xy}^{k} = \begin{cases} Q/L_k & if\ ant\ k\ used\ xy\ in\ its\ path \\ 0 & otherwise \end{cases} \qquad (4)$$

Where Q is a constant and L is the distance of the path taken by ant k. These formulations above can be applied into simulation software to model specific cases.

III. Problem Formulation

The particular problem is based off the standard Travelling Salesman Problem (TSP). Typically, problems of this type require the input of coordinates for multiple locations, the construction of a distance matrix, and several additional constraints, namely that the salesman must visit each location once and only once, and that the salesman must return to the starting position. The objective is to minimize the total travel distance.

In this case, nine points on the campus of The University of Texas at Dallas were selected for the delivery bot to travel to:

1. Canyon Creek Heights South
2. Engineering and Computer Science West
3. Naveen Jindal School of Management
4. Residence Hall South
5. Eugene McDermott Library
6. Sciences Building
7. SPN
8. Waterview Science and Technology Center
9. North Lab

The coordinates for these locations were approximated using an official campus map provided by The University of Texas at Dallas, shown in Fig. 1. Beginning from the bottom-left of the map, x-coordinates were provided using the horizontal axis at the top of the map, and y-coordinates were assigned numbers based on reverse alphabetical order, starting with the letter "N", alongside the vertical axis at the left of the map. So, for example, campus locations at row "N" would have a y-coordinate of 1, locations in row "M" would have y-coordinates of 2, and so on.

Fig. 1. Blank Campus Map of The University of Texas at Dallas

The distance matrix is a key component of TSPs, which provides the travelling distance from each location to all other locations. Unlike many conventional TSPs, the distance matrix for this case was not computed automatically using the distance formula for linear distances. Instead, values for the distance matrix were estimated using Directions from Google Maps. For example, to estimate the travel distance between SPN and North Lab, the address SPN would be entered as the starting point, and North Lab's address would be entered as the destination point. It is assumed that the travel distance in the opposite direction is equivalent. The resulting distance matrix is shown in Table 1, with distances in kilometers.

A primary advantage of this method is that the resulting distances are more suitable for real world application than purely linear distances. Practically, any sort of travelling bot is not going to be able to move in a perfectly straight line in order to move from one location to another, as there are almost always obstacles that must be navigated around and designated walkways that may wind and bend in nonlinear fashion. Google Maps, when computing walking directions, takes obstacles into account and only sends the bot on established walkways and sidewalks, which is precisely what a bot would do in practical settings.

TABLE I
DISTANCE MATRIX (KM)

Location	1	2	3	4	5	6	7	8	9
1	0	1.1	1.1	1.5	1.4	1.4	1.9	1.4	1.7
2	1.1	0	0.5	0.65	0.5	0.45	1.1	1.8	0.75
3	1.1	0.5	0	1.1	0.35	0.65	1.5	2.2	0.7
4	1.5	0.65	1.1	0	0.85	0.4	0.75	1.8	0.4
5	1.4	0.5	0.35	0.85	0	0.5	1.2	2.1	0.45
6	1.4	0.45	0.65	0.4	0.5	0	0.8	1.8	0.35
7	1.9	1.1	1.5	0.75	1.2	0.8	0	2.2	0.75
8	1.4	1.8	2.2	1.8	2.1	1.8	2.2	0	2.1
9	1.7	0.75	0.7	0.4	0.45	0.35	0.75	2.1	0

IV. MATLAB IMPLEMENTATION

This problem was implemented and solved using MATLAB. The code begins by requiring the user to configure initial parameters, beginning with entering coordinates for each of the locations for the bot to travel to. The code also gives the user the option to specify a maximum distance that the bot is able to travel. If the optimal distance is greater than this specified maximum, the code will run as normal and display the optimal route, but will also return an error message stating that the optimal route exceeds the maximum allowable distance, suggesting the user consider removing locations for that particular trip.

The program then proceeds to compute the distance matrix under the assumption that the delivery bot will travel from location to location in a straight line, as is typical for TSPs. Alternatively, users can manually provide a distance matrix, as was done in this case as described in the previous section. The user can then specify numerous ant colony algorithm parameters, including the maximum number of iterations, the total number of ants, the initial pheromone levels, the relative weights of the heuristic and pheromone levels in the probability function, and the pheromone evaporation rate, among others.

Following the problem definition, the algorithm begins. For each iteration, each ant moves from one location to the next, choosing the next location based on the probability function which weighs the nearest unvisited location against the level of pheromone laid by ants on previous iterations. For each ant during each iteration, the distance travelled is computed from the sum of values contained in the distance matrix based on the ant's particular route. If the current ant's route has a shorter total distance than the previous shortest route, then that current ant's distance replaces the previous shortest distance.

After all the ants have moved in the previous iteration, the pheromones are updated. For each route, the pheromones that are deposited are inversely correlated with the total route distance. This means that if the ant travelled a relatively short route, then more pheromone is deposited. If a route is relatively long, relatively fewer pheromones are left behind. This mechanism is a key component leading to convergence towards the optimal solution. This means that, over time, shorter routes have higher levels of pheromone, making them probabilistically more attractive for future ants to follow. Afterwards, after each iteration, the pheromones partially evaporate at a rate defined in the problem definition. This means that pheromones laid by more recent iterations will generally be stronger than pheromones laid by earlier iterations.

V. RESULTS

Once all iterations have completed, MATLAB exports two plots. The first one, an example of which is shown in Fig. 2, is a rudimentary plot of the locations the bot stops, in addition to lines interconnecting them. The lines connecting the points indicate the sequence in which the bot would visit each location, though without explicitly specifying a starting point or a direction.

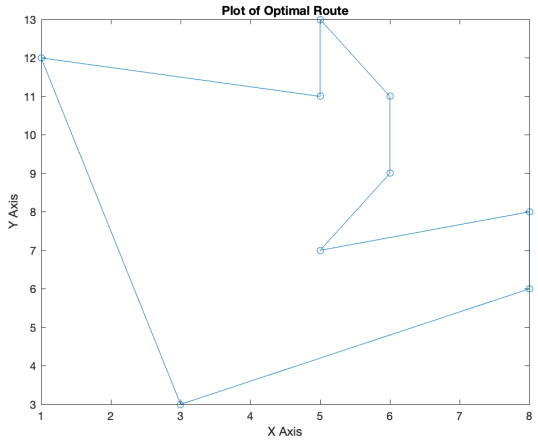

Fig. 2. Rudimentary Plot of Optimal Route

MATLAB also outputs a plot showing the minimum total distance computed up to each iteration, similar to that shown in Fig. 3. Based off this figure, the program took approximately 16 iterations to converge to the optimal route distance of 7.45 km. However, future instances of the program may produce somewhat different plots, as the ant colony algorithm is not a deterministic process. The procedure for each ant to select the next location to visit is based on a probabilistic function, and so it is expected that each time the code is run, the shape of the plot may vary, and it may take more or fewer iterations to converge to the optimal solution. Over several runs, this particular program typically, though not always converges

within approximately 20 iterations.

Using the optimal route data, the campus map was manually illustrated to provide more clarity about which locations would be visited in which order, as well as specifically which route the bot would take. This plot can be seen in Fig. 4. The route can originate at any point, and despite the arrows, also holds true for either direction, so long as the same direction is held throughout the entirety of the trip.

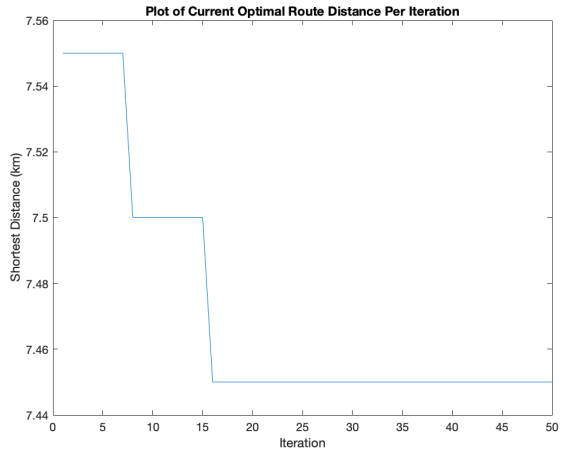

Fig. 3. Plot of Optimal Route Distance for Each Iteration

Fig. 4. Illustrated Campus Map with Optimal Route

VI. Conclusion

The ant colony optimization problem seen throughout this report illustrates the capability of this algorithm, as it is able to solve complex optimization problems quickly. With the average amount of iterations necessary to achieve an optimal solution being around 20, computing times and cost can remain minimal. As this algorithm matures future possibilities include real-world map application with the use of graph theory. This would allow for more precise route optimization for autonomous delivery robots everywhere. Another possible application would be to apply ant colony optimization to autonomous delivery vehicles like cars and vans, using simulation of Urban Mobility (SUMO) to simulate traffic, vehicle speed and other road conditions.

References

[1] "The Robots Have Landed", Auxiliary Services. [Online]. Available: https://services.utdallas.edu/dining/delivery/. [Accessed: 07-Dec-2021].

[2] Akka, K., & Khaber, F. (2018). Mobile robot path planning using an improved ant colony optimization.International Journal of Advanced Robotic Systems,15(3), 1729881418774673.

[3] Archetti, C., Speranza, M. G., & Savelsbergh, M. W. (2008). An optimization-based heuristic for the split delivery vehicle routing problem.Transportation Science,42(1), 22-31.

[4] Dorigo, M. (1996). The Ant System: Optimization by a colony of cooperating agents.IEEE Transactions on Systems, Man, and Cybernetics-Part B,26(1), 1-13.

[5] Dorigo, M., & Di Caro, G. (1999, July). Ant colony optimization: a new meta-heuristic. InProceedings of the 1999 congress onevolutionary computation-CEC99 (Cat. No. 99TH8406)(Vol. 2, pp. 1470-1477). IEEE.

[6] Dorigo, M., Maniezzo, V., & Colorni, A. (1991). The ant system: An autocatalytic optimizing process.

[7] Messac, A. (2015).Optimization in practice with MATLAB®: for engineering students and professionals. Cambridge University Press.