Dynamic Path Finding using Ant Colony Optimization

Reshma M, Neena Thomas, Surekha Mariam Varghese

Abstract: ***Ant Colony Optimization (ACO) has been commonly applied in solving discrete optimization problems. This is an attempt to apply ACO in a dynamic environment for finding the optimal route. To create a dynamically changing scenario, in addition to distance, constraints such as air quality, congestion, user feedback, etc are also incorporated for deciding the optimal route. Max-Min Ant System (MMAS), an ACO algorithm is used to find the optimal path in this dynamic scenario. A local search parameter ε is also introduced in addition to ρ to improve the exploration of unused paths. Adaptability was studied by dynamically changing the costs associated with different parameters.***

Keywords: ACO, MMAS, BreezoMeter, Google Maps, Traffic Congestion

I. INTRODUCTION

Traveling has a vital role, and it gives an extraordinary experience to every person's life. People of all ages, from all countries, travel to different places for many reasons. Map services make it possible for travelers to look for the information around them and schedule journeys to his/her preferred destinations. In most major cities, traffic congestion is a major crisis. The best way to reduce traffic and manage traffic is to reroute vehicles. The vehicles must be rerouted to a less congested route. Traveling through a polluted route is one of the major causes of most of the health issues in human beings. Discovering the pollution - free route is essential in aforesaid circumstances. Connected and smart streets can collect information and distribute information and resources from and to other users that include traffic details, road blockages, road works, etc. In this paper, an ACO algorithm based on MMAS along with a local search operator ε in addition to ρ parameter is presented. To keep the balance between exploration and exploitation, the pheromone trail evaporation is done in two phases. The goal of this method is to find a path between source and destination with less congestion, pollution, and crowdsourcing input. It uses Google Map API to evaluate the details on the path between a source and a destination. Among the available range of roads, users would be recommended a path with reduced congestion and pollution. The remaining paper is structured in the following way. Section 2, reviews the related work. Section 3 presents the proposed algorithm. In Section 4, the results are discussed and in Section 5, a series of experiments are conducted and the performance of the proposed algorithm is evaluated. Finally, the conclusions are given in section 6.

Manuscript received on January, 2021.
Revised Manuscript received on January 15, 2021.
Manuscript published on January 30, 2021.
* Correspondence Author
Reshma M*, Computer Science and Engineering, Mar Athanasius College of Engineering, Kothamangalam, Ernakulam, India. Email: reshmamohandas96@gmail.com
Neena Thomas, Computer Science and Engineering, Mar Athanasius College of Engineering, Kothamangalam, Ernakulam, India.
Dr. Surekha Mariam Varghese, Computer Science and Engineering, Mar Athanasius College of Engineering, Kothamangalam, Ernakulam, India.

II. LITERATURE SURVEY

Almost all vehicle routing schemes use the local map data. Google Maps API can be used to receive the shortest path information [1]. Considering distance and congestion, ACO can be applied in determining the optimum path. The criteria considered in [2] included the average speed of automobiles on a road and the measurements of air quality on a path. There has been a growing interest in applying ant colonies for optimization problems. ACO algorithms have proven to be useful in discrete optimization problems. Many researchers have used ACO for dynamic optimization as well. Casper and Mark have been successful in solving the Dynamic Traveling Salesman Problem (DTSP) with ACO [3]. The secret to the successful improvement of ACO is to tackle the exploration/exploitation problem effectively. Epsilon greedy is an important policy - based exploration method for reinforcement learning and has also been used as the pseudo - stochastic process to boost the ACO Algorithms. In reinforcement learning, important exploration strategies include epsilon greedy (ε - greedy) policy as in [4]. The ε - greedy policy adopts the current best selection from candidates with the probability of ε, and a random selection with the probability of 1- ε. In swarm intelligence and evolutionary computation, the local search focuses on the feasible neighborhood space of the current solution and seeks better solutions. For a local search, there is no guarantee to obtain a global optimum. Global search tries to explore all feasible solutions in the solution space to achieve the global optimum; however, it is quite a time consuming if not impossible. Combing global and local search is an important research method for swarm intelligence and evolutionary computation [5 - 8]. The algorithm attempts to balance global and local searches by applying epsilon greedy in ACO so that better solutions can be found more efficiently.

III. PROPOSED METHODOLOGY

The system aims at finding a congestion - free, eco - friendly optimal route. The parameters considered for the optimal route are distance, congestion, air quality, and traveler recommendation. Among these parameters, only distance is static and all other parameters vary dynamically. N shortest routes between the given source S and destination D and k intermediate points in each of the route returned by Google map API are taken. The users will be led to the least congested and polluted road. The system's key modules involve obtaining path details from the Google Map

API, obtaining intermediate points on the road, detecting congestion using data obtained from Traffic APIs, detecting air quality in particular areas, collecting feedback from users, finding an optimal route.

A. Identification of routes via Google Map API

Google Map API will help the user to locate the path from source to destination. The vehicle routing system sends map requests through maps API, and map servers send the result back to the application. The users can provide the origin and destination. The application will communicate with the Google Maps API Directions Service. When specifying the origin or destination in a directions request, the user can specify a query string. The Directions Service returns directions using a series of waypoints. The path directions are shown as a polyline on a map. Travel time is the primary factor that is optimized, but other factors such as distance can also be taken into account. The routes obtained will have varying air quality and congestion. Thus it is better if the intermediate points along the route are determined. It will become easier to determine the air quality levels and congestion between the intermediate points. The direction steps are taken as the intermediate points. A single step describes a specific, single instruction on the journey, e.g., "Turn left at the fourth street". The step not only describes the instruction but also contains distance information relating to how this step relates to the following step.

B. Estimation of Congestion

Congestion on the road is induced by a change in vehicle volume. When probe vehicles using the app move with GPS location enabled, these coordinates are used to compute the speed of the vehicle. Reduced speed of vehicles in a particular segment signifies congestion in that segment. As more and more drivers use the app, congestion computation becomes more accurate.

The latitude and longitude of the intermediate points are used to find traffic information. This is achieved with the assistance of Traffic API. The different colors on the routes in the traffic layer show the current status of the traffic on these routes. Various colors are being used to denote live traffic status. They are green, orange, and red denoted with values 0, 1, and 2 respectively. Green means there are fewer traffic delays. Orange means there is a medium amount of traffic. Red means there is high traffic.

C. Detection of Air Quality

Although many initiatives and information sources are helpful to broadly inform on air pollution, they often lack key elements that are required to provide individuals with relevant information, namely real - time and location - based data. By the time the data is available, the conditions are likely to have changed and not relevant to the point of interest.

In contrast, BreezoMeter's air quality data is easily accessible through API calls and is real-time and location-based. Moreover, Breezometer verifies all its data sources and performs strict QA to ensure the highest level of accuracy. Thus, BreezoMeter API is the best choice to collect air quality information.

BreezoMeter accepts the latitude and longitude information of the intermediate points and responds in real-time with air quality values. The BreezoMeter's Air Quality Index (BAQI) scale ranges from 0 = "poor" to 100 = "excellent" with five categories, namely poor (0 - 19), low (20 - 39), moderate (40 - 59), fair (60 - 79) and excellent (80 - 100).

D. Collection of Feedback

User recommendations are very important in improving the performance of any system. A feedback system collects responses from the traveler as good, satisfactory, or poor for different aspects such as timeliness, eco-friendliness, etc about the route obtained. The users can enter their feedback based on the satisfaction level they obtained after traveling through a particular route. This can depend on whether the route is prone to accidents, road conditions, road jumps, jerking, etc. This feedback of the traveler is distributed among different segments of the route traveled by him. As more and more travelers use the different segments, their crowd sourced importance will vary making the value dynamic.

E. Implementation of ACO

There are well-known deterministic algorithms capable of solving shortest path problems in a directed graph with n vertices and m arcs. Dynamic variations of shortest path problems allow the weight function to be changed while determining the shortest paths, or after they've been calculated. This could occur in the shortest route navigation problems where the weight feature represents travel time that can be affected by traffic, pollution, etc. ACO algorithms can continue the optimization process even after the graph changes. When minor changes are made in the graph, ACO can achieve better performance than other algorithms that have to start from scratch whenever the graph is modified [9].

Ants move from source to destination in each iteration, passing through different sub - routes. The parameter of the pheromone constant is used to deposit pheromone on the map. After ant travels from one stage to another, it deposits a pheromone that reinforces the route to another ant trailing the road, the pheromone trail that decides the next node to follow: node i, an explorer ant m select probabilistically node j to follow next using probabilistic rule shown in (1).

$$P_{ij}^{k}(t) = \left\{ \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{i \in N_i^k}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}} \right\} \qquad (1)$$

where τ (i, j) is the strength of pheromone deposited by each ant on the path (i, j), α is the intensity control parameter. η (i, j) is the gain of the path (i, j) which is determined by η (i, j) = 1/Q(i, j), where Q(i, j) is the least distance of move from node point i to the destination node point j, β is the visibility control parameter. ACO performance can be improved by obtaining stronger exploitation of the best solutions found during the search. This will lead to the problem of stagnation of the search. Therefore, the key to achieving the best performance of ACO algorithms is to combine improved exploitation of the best solutions found during the search with an effective mechanism for avoiding early search stagnation. MMAS has been developed to satisfy these requirements. The local search exploration of unused paths is also improved by introducing a ε parameter in addition to ρ in MMAS.

The major steps in applying ACO include:

- Initializing ants: Artificial ants are located at each source node. The probability of selecting an edge between city x and y by the k^{th} ant is given by (2).

$$P_{xy}^{k} = \frac{\left(\nu_{xy}^{\mu}\right)\left(\tau_{xy}^{\alpha}\right)\left(\eta_{xy}^{\beta}\right)\left(\phi_{xy}^{\gamma}\right)\left(\delta_{xy}^{\lambda}\right)}{\sum_{z \in allowed_y}\left(\nu_{xy}^{\mu}\right)\left(\tau_{xy}^{\alpha}\right)\left(\eta_{xy}^{\beta}\right)\left(\phi_{xy}^{\gamma}\right)\left(\delta_{xy}^{\lambda}\right)} \quad (2)$$

Here, the ν parameter is the strength of pheromone deposited by each ant on the path. τ parameter is considered as the distance parameter of the route, η is used as the parameter for indicating the congestion rate of a particular route, φ is used for indicating air quality value of that route and δ is used as the parameter for crowdsourcing. μ, α, β, γ, and λ refers to the preference of the respective parameters.

- Pheromone update: In MMAS only one single ant is used to update the pheromone trails after each iteration. The modified pheromone trail update rule is given by (3).

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best} \quad (3)$$

where $\Delta\tau_{ij}^{best} = Q/L_{best}$. Here, Q is a constant and L_{best} denotes the solution cost of the global-best solution. The pheromone update is the most important means of search exploitation in MMAS. By this choice, solution elements which frequently occur in the best-found solutions get a large reinforcement. The pheromone evaporation takes place once the ants complete each iteration. It is given by (4).

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad (4)$$

where ρ is the pheromone evaporation coefficient. After pheromone update by global-best path, search stagnation may occur. This can occur if the pheromone trail is much higher for a choice than all others at every selection point. In this situation, due to the probabilistic choice governed by (2), an ant will prefer this solution component over all alternatives and further reinforcement will be given to the solution component in the pheromone trail update. In such a case, the ants often create the same solution and the search space exploration is stopped. Such a stagnation situation should be avoided. One way to do this is to affect the probability of selecting the next component of the solution. Yet by restricting the pheromone trails' effect one can easily prevent the relative differences between the pheromone trails from being too drastic during the algorithm's run. To achieve this goal, MMAS imposes explicit limits τ_{min} and τ_{max} on the minimum and maximum pheromone trails such that for all pheromone trails $\tau_{ij}(t)$, $\tau_{min} \leq \tau_{ij}(t) \leq \tau_{max}$. After each iteration one has to ensure that the pheromone trail respects the limits. The exploration of local search needed to be improved to obtain an optimal solution. Otherwise, there is a chance of getting trapped in sub-optimal solutions. This is implemented by introducing an additional parameter called local search parameter, ε in addition to ρ to explore the unused paths. The ε parameter evaporates the pheromone in the edges of the used paths other than that of the best path. It is given by (5).

$$\tau_{ij} = (1-\varepsilon)\tau_{ij} \quad (5)$$

where ε is the local search parameter.

- Stopping procedure: The ACO algorithm is concluded by achieving a predefined number of iterations while an ant is discarded before hitting its target by achieving a predefined maximum number of hops. The total number of iterations is100. The values of alpha, beta, gamma, and lambda can range from 0.1 to 1.0.

IV. RESULT

Various paths between source and destination have been obtained utilizing the Google Map API. Congestion and air quality index, as well as reviews from the users, are also collected. MMAS with local search parameter ε was implemented to achieve the path with less congestion and pollution. 100 iterations are done to achieve the desired outcome. Fig. 1 shows three of Google's routes from a source to a destination. The time and distance to travel are provided on the map. Fig. 2 indicates the intermediate points of the path. The points also get the latitude and longitude pair measurements. Fig. 3 provides information on the congestion of a specific latitude-longitude pair. Fig. 4 displays a road network for applying ACO. Fig. 5 depicts optimal path finding. Users will insert values ranging from 0.0 to 1.0 to alpha, beta, gamma, and lambda. This recommends the best routes depending on the preferences.

Fig. 1. Three best routes from source to destination

Fig. 2. Intermediate points of a route

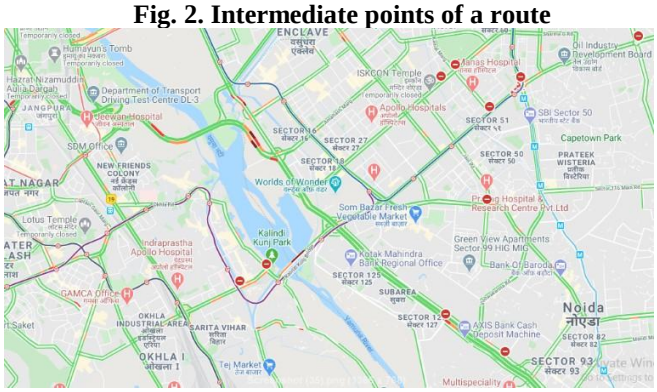

Fig. 3. Details of congestion of an intermediate point

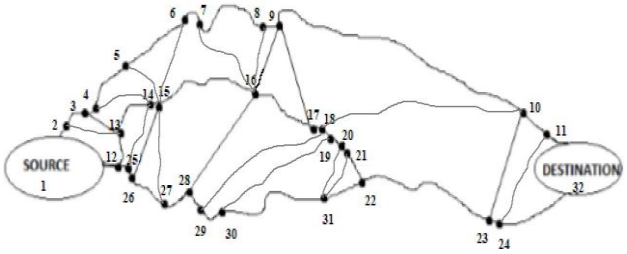

Fig. 4. Road network under consideration for applying ACO

V. PERFORMANCE ANALYSIS

The routing applications can improve their performance by incorporating user recommendations, congestion levels, and pollution levels of an area in the decision of selection of routes.

```
Iteration : 8
[[ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 ...
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]]
Iteration : 9
[[ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 ...
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]]
route of all the ants at the end :
[[ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 ...
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]
 [ 1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32]
 [ 1. 12. 25. 26. 27. 28. 29. 30. 31. 22. 23. 24. 32.]]
best path : [1. 2. 3. 13. 14. 15. 16. 8. 9. 17. 18. 19. 20. 21. 22. 23. 24. 32.]
```

Fig. 5. Optimal Path Finding

The experiments were carried out on various combinations of sources and destinations. Those were in Delhi and near it. It is a highly congested and polluted area in India. So this region is well suited for evaluation. Preference is given for distance, congestion, air quality, user feedback. The readings have been taken, and the percentage change by which the obtained route differs from the suggested route by Google is calculated. The experiments are conducted by varying one parameter from 0.1 to 1.0 and keeping the other three parameters constant. The fig. 6-9 displays a comparison of the collected results.

From fig. 6, it can be noted that as the distance parameter is considered, the percentage change from Google's suggested path changes only very slightly for higher values of α and varies greatly when the value of α is lower. Fig. 7 shows that as the congestion preference parameter (β) varies from 0.1 to 1.0, the percentage of change is higher.

Considering the air quality preference parameter (γ) and the satisfaction level preference parameter (λ), the percentage change is also higher and is slightly higher than the percentage change of β parameter. So it can be concluded from the observation that percentage change from Google's suggested route showed higher value for air quality and satisfaction level preference parameter.

Dynamism is introduced in the problem with an increase or decrease of edge costs, change in air quality, user feedback values, etc. The problem under consideration is highly dynamic as the values of each of the parameters can change instantly.

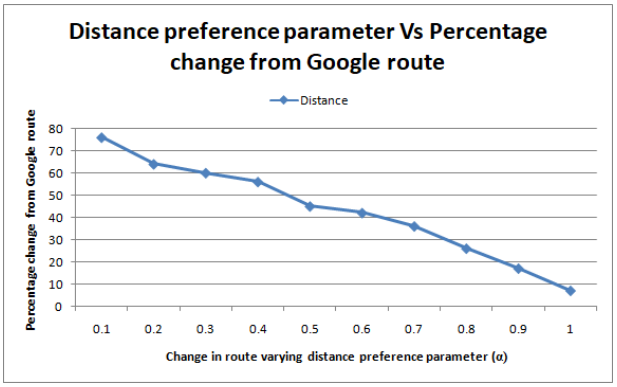

Fig. 6. Varying distance (α) parameter

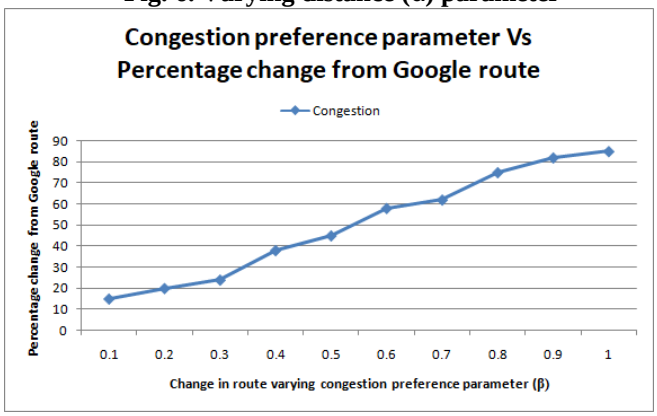

Fig. 7. Varying congestion (β) parameter

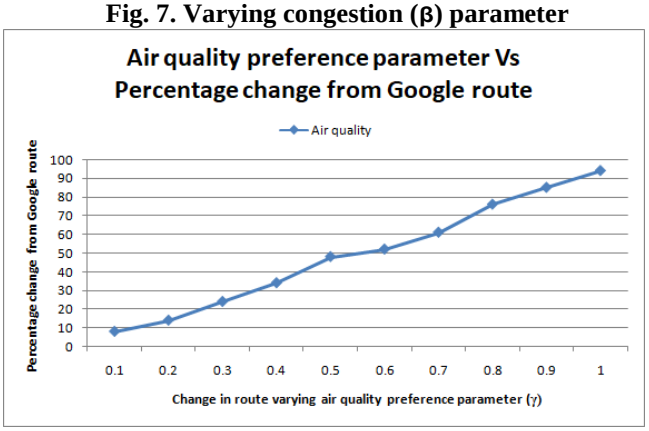

Fig. 8. Varying air quality (γ) parameter

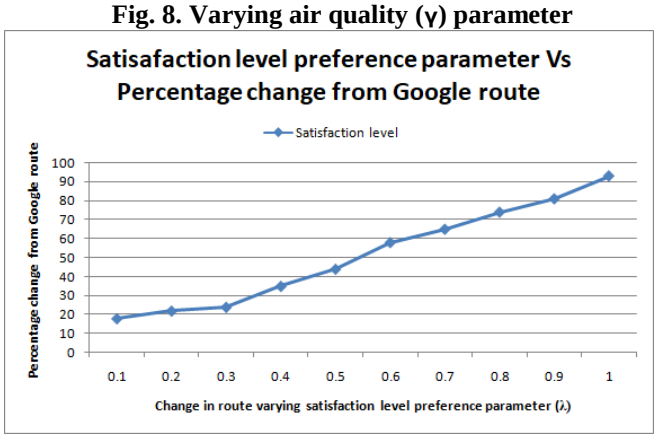

Fig. 9. Varying satisfaction level (λ) parameter

The adaptability of the algorithm to positive and negative changes is studied. A positive change is brought by introducing a good edge, i. e., one with a smaller edge cost, whereas, a negative change can be brought by introducing an edge with greater edge cost.

A fifty node graph is considered to study adaptability. The experiments were conducted by making both positive and negative changes in an edge. Let i be the iteration on which the change was introduced, and j be the number of iterations taken to reach the solution. The values of i and j for both positive and negative changes are noted. The experiment is then repeated several times. The average value of j is found for the corresponding i value for both positive and negative changes. Fig. 10 and 11 show the adaptability graph where the average value of j is plotted varying i for positive change and negative change respectively.

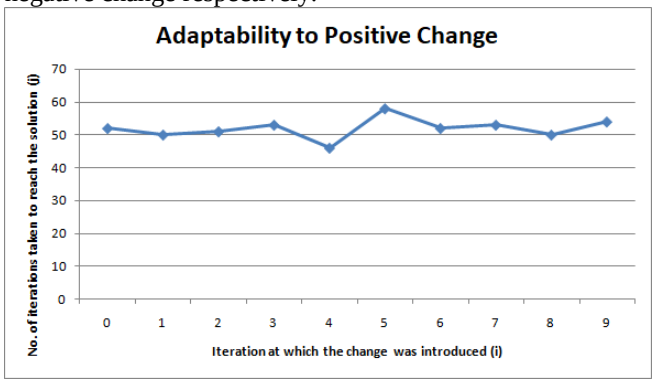

Fig. 10. Adaptability graph for positive change

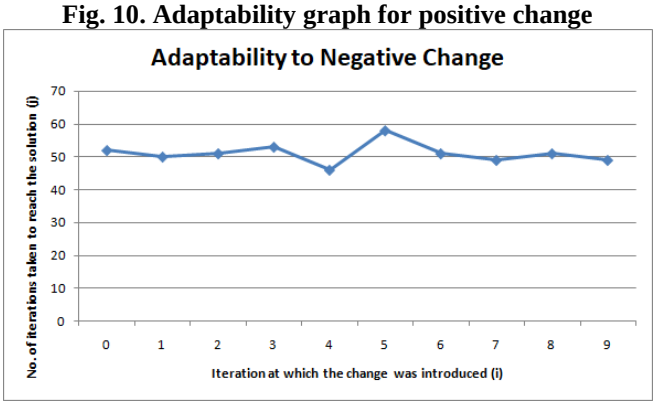

Fig. 11. Adaptability graph for negative change

VI. CONCLUSION

MMAS proves to be effective in solving dynamic optimization problems. From the observations, it could be concluded that as the preference of distance was higher, the optimal route differed only slightly from Google's suggested path. But while the parameters congestion preference, satisfaction level, and air quality preference were considered, this percentage of change was higher. Google's suggested route varied highly when the air quality preference value was considered. From the study conducted with a virtual graph of fifty nodes, it can be concluded that the MMAS algorithm with a local search parameter, ε is very much adaptable to both positive and negative dynamic changes. Thus the method adopted is very much effective in finding an optimal path in a dynamic environment.

REFERENCES

1. Aparnasree R, Ani Sunny, and Surekha Mariam Varghese, "Real time traffic congestion detection and optimal path selection using big data processing," in *International Journal of Control Theory and Applications*, 2016.
2. Neena Thomas, Athira Balagopal, and Surekha Mariam Varghese, "Green route: an ecofriendly route suggestion and description based on congestion and air quality," in *International Journal of Innovative Technology and Exploring Engineering*, 2019.
3. C. J. Eyckelhof, and M. Snoek, "Ant systems for a dynamic TSP: Ants caught in a traffic jam," in *Proc. ANTS 2002*, 2002, pp. 88-89.
4. Yahui Liu, Buyang Co, and Hehua Li, "Improving ant colony optimization algorithm with epsilon greedy and Levy flight," in *Complex and Intelligent Systems*, 2020.
5. F. G. Guimaraes, F. Campelo, H. Igarashi, D. A. Lowther, and J. A. Ramirez, "Optimization of cost functions using evolutionary algorithms with local learning and local search," in *IEEE Trans Magn*, 2007, pp. 1641-1644.
6. C. Qian, Y. Yu, and Z. H. Zhou, "Pareto ensemble pruning," in *Twenty-ninth AAAI Conference on Artificial Intelligence*, 2015.
7. C. Qian, J. C. Shi, K. Tang, and Z. H. Zhou, "Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee," in *IEEE Trans Evol Comput*, 2017, pp. 595-608.
8. T. Weise, Y. Wu, R. Chiong, K. Tang, and J. Lassig, "Global versus local search: The impact of population sizes on evolutionary algorithm performance," in *J Glob Optim*, 2016, pp. 511-534.
9. Andrei Lissovoi, "Analysis of ant colony optimization for dynamic shortest path problems," Master's Thesis, Department of Informatics and Mathematical Modeling, Technical University of Denmark, 2012.

AUTHORS PROFILE

Reshma M earned a Bachelor of Technology in Computer Science and Engineering from Jawaharlal College of Engineering and Technology, Lakkidi in 2018 and a Master of Technology in Computer Science and Engineering from Mar Athanasius College of Engineering, Kothamangalam, which is affiliated to Abdul Kalam Technical University. Her research interests cover Big Data Analytics, Database Management, and Data Mining.

Neena Thomas received Bachelor of Technology in Computer Science and Engineering from Cochin University of Science and Technology in 2017 and Master of Technology in Computer Science and Engineering from Mar Athanasius College of Engineering Kothamangalam affiliated to APJ Abdul Kalam Technological University. Her research interest is in Big data processing, Database Management and Data Mining.

Dr. Surekha Mariam Varghese currently works as Professor in Computer Science and Engineering department, Mar Athanasius College of Engineering, Kothamangalam, Kerala. In 1990, she received her Bachelor of Technology in Computer Science and Engineering from the College of Engineering, Trivandrum affiliated with Kerala University, and a Master in Computer and Information Science Technology from CUSAT, Kochi, in 1996. In 2009, she received a Ph.D. in Computer Security from CUSAT, Kochi. She has accumulated 27 years of teaching and research experience at different Indian institutions. Her research interests cover Machine Learning, Network Security, Database Management, Data Structures and Algorithms, Operating Systems and Distributed Computing.