International Journal of Control Theory and Applications

ISSN : 0974–5572

© International Science Press

Volume 9 • Number 42 • 2016

A Cloud-based Solution of Dynamic Traffic Routing Problem for Autonomous Robots

Rajesh Doriya[a] Avinash Kumar Singh[b] and Pavan Chakraborty[b]

[a]*Department of Information Technology, National Institute of Technology, Raipur India -492010*

E-mail: rajeshdoriya.it@nitrr.ac.in, Corresponding Author

[b] *Robotics and Artificial Intelligence Lab, Indian Institute of Information Technology, Allahabad India - 211012*

E-mail: avinashkumarsingh1986@gmail.com, pavan@iiita.ac.in

Abstract: Dynamic Vehicle Traffic Routing is one of the interesting and challenging research area due to its stochastic nature and practical usage in navigation system. This paper presents a solution to Dynamic Vehicle Traffic Routing problem in which a cloud based approach is utilized to compute the most suited route to reach the destination given that traffic is changing stochastically. Here, the efficient route is calculated on the basis of historical traffic data and ranking of roads. In our system, a cloud server observes the traffic patterns and send the updates after calculating most promising move to the vehicles as per their destination. We also tested the working of cloud-based dynamic traffic routing service for autonomous robots with various synthetic data set for 20 nodes by adapting different constraints.

Keywords: *Dynamic Vehicle Traffic, Robotic Services, Cloud Computing, Vehicle Routing Problem, Cloud Robotics.*

1. INTRODUCTION

Autonomous Guided Vehicle [1] is a type of mobile robot that can use vision or other sensors for navigation. In the arena of robotics, mobile robotics has gained much attention due to its economical impact. It has become the center of revolution and an active field of research as it reduces costs and increase the efficiency of the system by automation. One of the noticeable projects carried out by S. Thrun at Google is Google's Autonomous Car [2]. The routing of a car like a robot in an urban environment is one of the interesting and challenging research area due to its stochastic nature and practical usage in the navigation system.

In DVRP, all relevant information of the vehicle traffic is not known to the planner and the information changes after the initial route planned. The DVRP belongs to the category of NP-hard problems as the problem can be solved in reasonable time for realistic problem size.

In a survey carried out by Psarafits [3], only a few general results for a simple version of DVRP problem have been reported. Powell et. al [4] distinguish dynamism between a model and its application, according to him, a problem is dynamic if its parameters are a function of time and the model is solved repeatedly upon receiving new information. Several researchers presented many algorithms to solve classical shortest path problem. In a non-stochastic network, a linear or exponential utility function can be solved using an efficient Dijkstra's algorithm [5]. Hall [6] showed that it fails to address the problems of the non-stationary stochastic network. In the same class of problem, the AO* algorithm with heuristics such as air distance and expanding some nodes under breadth search can result in efficient results than the dynamic programming. A new algorithm for a least shortest path for the non-stationary stochastic network is also presented by Hooks [7].

Cloud-enabled robots [8-10] primarily expect two functionalities from any cloud service provider, i.e., computation offloading and collaboration and information sharing [11-13]. Both functionalities differ from each other by a far margin; the first cloud service emphasizes on performing computational expensive part of a robot at cloud infrastructure; and the second cloud service enables the robot to share their knowledge with others present in the cloud eco-system and it also promotes collaboration to perform a task.

In this paper, we have implemented a cloud-based solution of dynamic traffic routing problem for autonomous robots by synthetically generated traffic data.

2. DYNAMIC TRAFFIC ROUTING PROBLEM FOR ROBOTS WITH CLOUD

The study of Travelling Salesman Problem (TSP) [14] and Vehicle Routing Problem (VRP) [15] has carried a wide attention in the scientific community. TSP deals with the problem of visiting multiple cities (start and return to the same city) optimally in a single tour. Whereas, VRP depicts the problem of delivering logistics by different vehicles (may start with different depots) to different places optimally having that each vehicle has a limited carrying capacity. The problem of VRP can be classified into two categories: Static VRP and Dynamic VRP. Dynamic VRP is much more complex and challenging due to the following facts:

1. Time dimension is important,
2. Future information is unknown,
3. Neighboring events are very important,
4. Information needs to be updated at regular intervals,
5. Faster computation of information is required,
6. Presence of unavoidable side constraints (such as accidents, road blocks, etc.),
7. Different objective functions, etc.

Dynamic Routing in any kind of network has always been a challenging problem. In our system, we refer to Dynamic Vehicle Traffic Routing Problem (DVTRP), where a network of a city is considered as a problem space. However, if the problem is large then it is first divided into a smaller set of a problem then each small problem set is addressed individually and the result is combined.

A city network can be seen as a network or graph of several nodes (road junctions) joined by links (roads). The average velocity of the vehicles on each road is acquired through the deployment of fixed cameras with a small computational device. This computational device computes the average velocity of the vehicles from the images and transmits it to the OpenStack cloud [16]. The velocities are used to convert the distance graph of the cities into a time graph by dividing each branch distance by the average velocity of the car on that branch. Every car travelling in the city is connected to the cloud. The vehicle provides its destination and its present location through GPS to the cloud. The cloud maps these vehicles on the graph and optimizes the travelled time. The path is then conveyed to each vehicle accordingly by the cloud.

The problem of navigation for vehicular robots in a city like operating environment can be seen as the Dynamic Vehicle Traffic Routing Problem (DVTRP) where a vehicle is required to travel from source to destination point in shortest time considering the fact that traffic is changing stochastically. A solution to DVTRP in which a cloud-based approach is utilized to compute the most suited route to reach the destination given that traffic is changing stochastically. The efficient route is calculated on the basis of traffic history and ranking of roads. A cloud infrastructure observes the traffic patterns and sends the optimum route in terms of time to reach. In our system, we refer to DVTRP, where a network of a city is considered as a problem space. However, if the problem is large then it is first divided into a smaller set of a problem then each small problem set is addressed individually and the result is combined.

3. DVTRP PROBLEM FORMULATION

Let the network is $G \equiv (V, E)$ Where, V and $E \subseteq V \times V$ are finite sets representing the set of nodes and set of directed links. This model can be seen as the network of intersections (nodes) and roads (links). In this model, $(n, n') \in E$ iff there exists a road segment from n to n'. If $n_0 \in V$ is a start node and $g \in V$ is a goal node. The successor for each element can be denoted by SUCCESSOR (n) which is set of nodes having an incoming link emanating from n

$$\text{SUCESSOR}(n) \equiv [n' : (n, n') \in \text{E}] \tag{1}$$

A path $P = (n_0, n_1, \ldots, n_k)$ from $n_0 \in E$ forms a sequence of such that $n_k + 1 \in \text{SUCCESSOR}(n_k)$ where $k = 0,1,..., k - 1$. If T_G denotes the travel time between n_0 and g, then

$$[T_G]_k (t) = \Sigma T_{ij}(t) \tag{2}$$

Where T_{ij} denotes the path between two nodes at time t. Here, $i \neq j$ and k denotes the number of possible paths to reach goal node.

Since, the path travel time in traffic is not trivial therefore T_{ij} can be treated as a random variable. Let, μ and σ^2 are the mean and variance of the random variable. So, the problem of ending optimal path for vehicle in traffic can be denoted by the following optimization equations:

$$\min_{\text{Expected time}} [T_G(t)] \tag{3}$$

subject to, $$T(t) = \Sigma T_{ij}(t) Z_{ij} \tag{4}$$

where Z is the stochastic path incidence variable. Z_{ij} denotes a feasible path between any two nodes of the network. Since the path travel time in traffic is not trivial, therefore $T_{ij}(\cdot)$ can be denoted as a random variable. The algorithm takes the source and destination as input from the navigating vehicle. Then, starting with the source node n_0, it ends the best feasible successor SCS (n) using the current traffic data, and other heuristics. Deciding the next node n' is where a wise decision should be made. Node n' will be the successor, if the node is not forming a loop or it has not been visited from this node before.

4. IMPLEMENTATION OF DVTRP AS A SERVICE FOR ROBOTS AT CLOUD

Fig. 1 illustrates a graph of 20 nodes where starting node is $n_0 = 1$ and destination node is $g = 20$. Each link indicates a tuple of two values, the first value represents the average vehicle traffic on that link and the second value is the distance of that link. Some highlighted tuples show that vehicle traffic values have been changed from the last traffic map hence shows the dynamic nature of the graph. A path $P = 1 \rightarrow 4 \rightarrow 5 \rightarrow 10 \rightarrow 14 \rightarrow 15$ indicates a possible solution in the problem space, where source node is 1 and goal node is 15. In the graph, a loop consists of green colored links with dotted oval shows a possible deadlock situation $1 \rightarrow 6 \rightarrow 3 \rightarrow 1$. At its core, our algorithm searches the graph starting from source node, exploring adjacent nodes until the destination is found. While searching for a good path it will keep an account of the number of hops travelled, and hence reducing the congestion at the junctions which in turn helps increasing the overall network consistency. The heuristics will help in finding a robust path which can be judged from the previously available data. A feasible path (in terms of time taken) is computed by observing current traffic conditions from the network.

The algorithm takes the source and destination as input from the navigating vehicle. Then, starting with the source node n_0, it finds the best feasible successor SUCCESSOR (n) using the current traffic data, and other heuristics. In order to decide the next node (n'), a wise decision should be made. Although, node n' will be the successor, if the node is not forming a loop or it has not been visited from this node before. Now, if this node is not the destination node, then it takes the following into account:

1. The current traffic values on links (with help of average velocity and (n, n')).
2. The historical usage (traffic history), which gives the algorithm an intelligence to move towards goal node g and reducing the search space. It may also help in breaking ties between two roads with equal traffic values.
3. The probability of traffic change (vehicle traffic window, *i.e.*, traffic in different slots on a given day) at a particular time, which decreases the chance of selecting a road which is free at this moment but may get congested at some other time.
4. Constraints parameters which help in finding the best path by applying various constraints such as opting different hop count values in a vehicle traffic network.

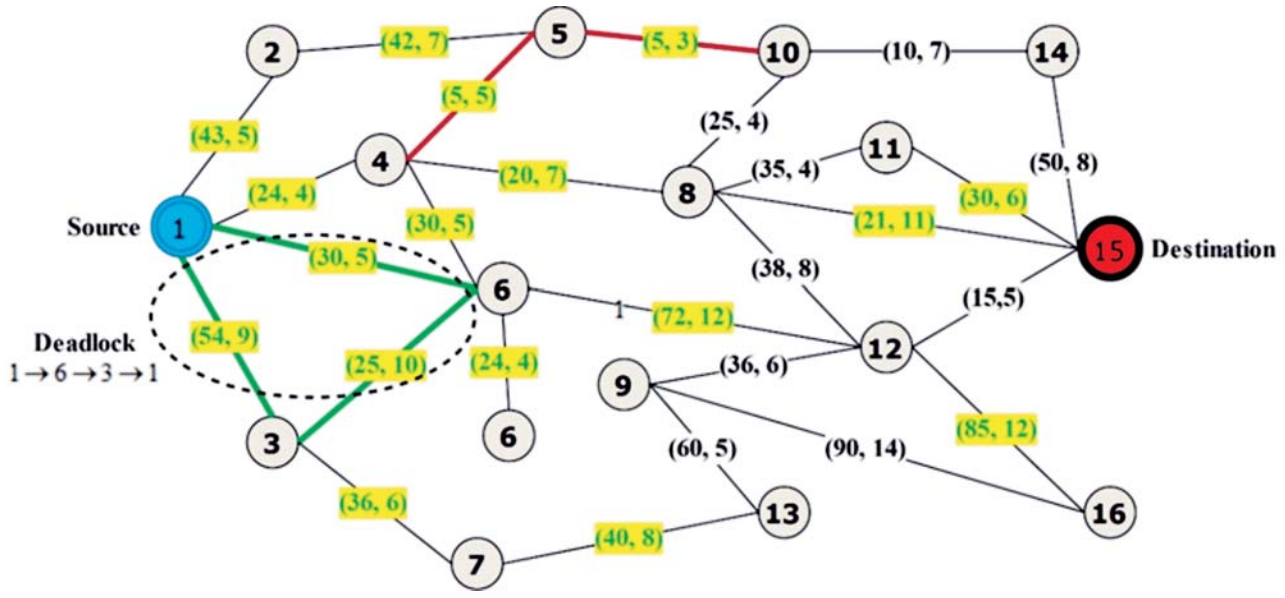

Figure 1: Problem space of 20 nodes for DVTRP

These parameters help in computing an expected value for every immediate neighbour using Dijkstra's algorithm [17] for finding a path with minimum weight forms all the nearest neighbours into consideration through which the better-seen option should be chosen. While traversing a new node, it keeps a variable hop count and marks itself as visited and adds the processing node to the path vector. Thereafter, it checks the average hop count variable, if the value of the hop count variable is greater than average hop count variable to reach the destination, then it discards that path for further exploration. If it exceeds, then the backtracking is initiated, otherwise, it recursively traverses to the adjacent nodes to find the destination. However, if the current node is the destination, then the search is terminated.

In our evaluation, CPU run time(s) is used as a principle metric to find minimum time to get the optimum route. In the quest of finding an optimum route, the T_{min}, T_{max} and T_{avg} indicate the minimum, maximum and average time obtained from 10 runs over a dynamically created network of 20nodes. Here, we have assumed that the starting node is $n_0 = 1$ and goal/destination node is $g = 20$. The σ and τ represent standard deviation and fault tolerance parameter respectively, which indicates an average number of faulty paths in a vehicle traffic network that can be caused from any reason of road block. Three cases for each network of 20 nodes with random traffic is taken into account. We represent simple Dijkastra's shortest route finding algorithm as Local Search (LS) parameter in this service.

In order to find the best optimum path, we have taken three constraints into consideration: Hop Count (HC), History Parameter (HP) and Fault Tolerance (FT). The HC parameter describes the number of links considered simultaneously to calculate the path to reach the destination. The HP parameter details the knowledge about the usage and trustworthiness of any link in the network. The FT parameter is used to create the random situations in the network. These random situations can be occurred due to road blockage, VIP movement, etc.

In Table 1, three sub-tables are shown, each entry shows the best-obtained path P for given constraints. In a sub-table, the best optimum path is evaluated after varying hop-count variable (where HC1 < HC2 < HC3) with the values 1, 2 & 3 respectively. From the running times in the given table, we see that T_{min} decreases when there is an increase in deviation of the average hop count for choosing a path. Here, T_{max} shows the worst case time where excessive backtracking happens. The second sub-table of each table shows the resulted paths when HP is applied in conjunction with HC. In the third sub-table, the HC, HP and FT parameters are applied simultaneously on the network. In each table, we have only shown the best path out of the 50 iterations.

Table 1
Results obtained after 50 iterations on 20 nodes with different constraints. Each path shows the best path generated from 50 iterations

(a) for constraint HC

S. No.	*Computed best path P from 50 iterations*	T_{min}	T_{max}	T_{avg}	σ	τ	*Constraints*
1.	1 → 3 → 2 → 9 → 13 → 17 →20	83	501	229.88	115.28	–	HC = 1
2.	1 → 2 → 3 → 4 → 5 → 10 → 12 → 18 → 20	125	412	306.70	47.22	–	HC = 2
3.	1 → 5 → 4 → 9 → 13 → 15 → 17 → 20	78	679	348.18	118.63	–	HC = 3

(b) for constraint HC and HP

S. No.	*Computed best path P from 50 iterations*	T_{min}	T_{max}	T_{avg}	σ	τ	*Constraints*
1.	1 → 7 → 8 → 9 → 13 → 17 → 20	94	415	203.94	68.91	–	HC = 1, HP
2.	1 → 7 → 8 → 9 → 13 → 17 → 20	92	491	246.74	86.97	–	HC = 2, HP
3.	1 → 2 → 9 → 8 → 15 → 17 → 20	83	548	286.20	146.62	–	HC = 3, HP

(c) for constraint HC, HP and FT

S. No.	*Computed best path P from 50 iterations*	T_{min}	T_{max}	T_{avg}	σ	τ	*Constraints*
1.	1 → 2 → 8 → 9 → 13 → 17 → 20	150	470	414.81	92.49	1.7	HC = 1, HP, FT
2.	1 → 7 → 8 → 9 → 13 → 18 → 20	309	759	416.48	105.46	1.32	HC = 2, HP, FT
3.	1 → 2 → 9 → 8 → 15 → 17 → 20	330	684	413.65	83.11	1.7	HC = 3, HP, FT

Our observation showed that the amount of backtracking has a considerable impact on the execution time of a program. With the increase in a number of nodes in the network, the number of intermediate nodes between the source and destination increase and the number of possible ways between them increases drastically which also increases the program execution time. As we can observe from Table 1 (*a*), when only LS and HC were applied, the output is almost arbitrary, *i.e*., any path satisfying the given constraints was the output which may or may not be an optimum one. In this case, the search is almost blind and so the amount of backtracking is also large most of the times, which can be seen by comparing the average times. However, when hop count is increased probability of getting optimum path is always increased.

In Table 1, when a sense of direction was given to the algorithm through the HP parameter the performance improved significantly. A number of times the search needed to backtrack reduced considerably and the output inclined towards the routes given in the table. The routes given in the table appeared repeatedly despite dynamic nature of the network. And by changing the TC parameter we see that when we can have alternate paths apart from those set of paths that are repeatedly chosen by the long-term history parameter and hence enhancing the quality of unbiasedness towards some good links which may be busy at specific times of the day.

Table 1 (*c*) shows the experiment with the generating faulty paths:

1. The readings do not include the worst case scenario of paths being blocked in a way that there is no path to the destination.
2. Minimum number of faulty paths generated is 1.
3. Also, the experiment does not consider a scenario when the parameters are badly tuned.
4. Average FT rate implies the average number of faults generated per iteration considering a set of 50.

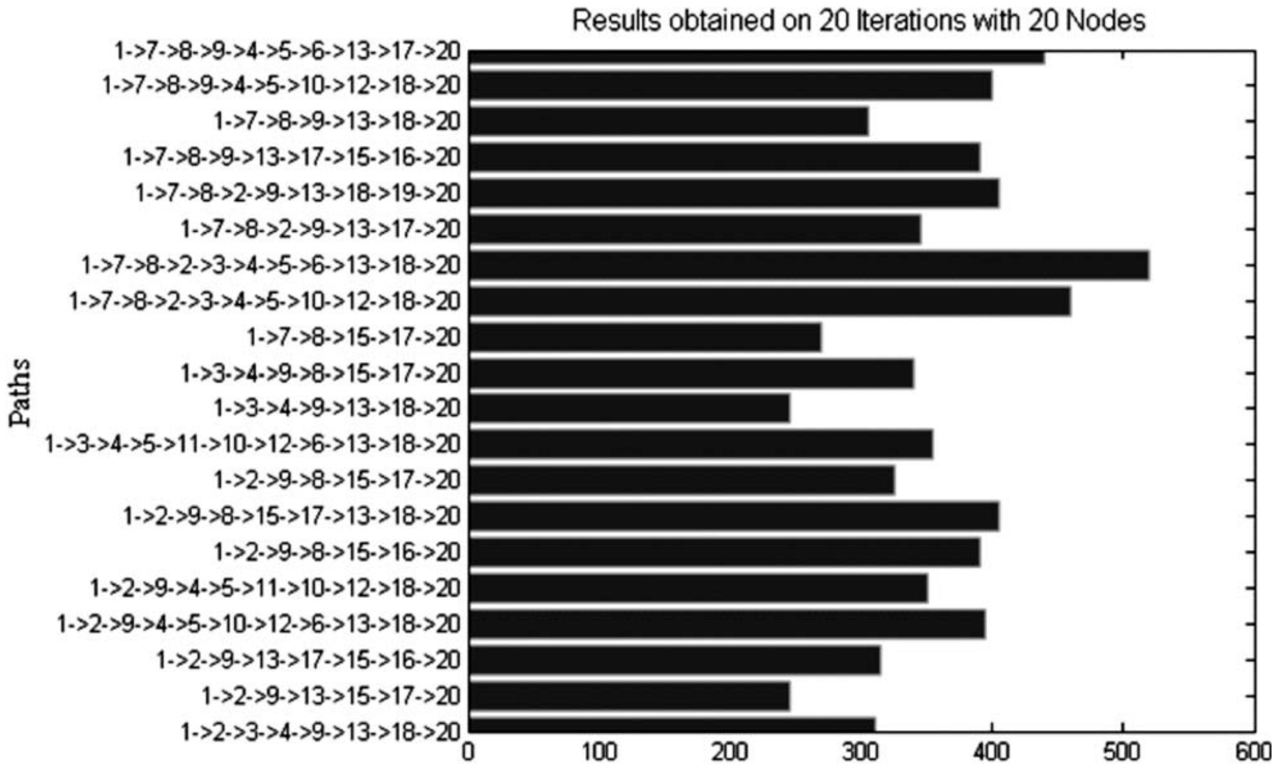

Figure 2: Results obtained on 20 iterations with 20 nodes. Y-axis and X-axis denote the paths generated and time took to calculate the path (in seconds)

Fig. 2 shows the effect of three constraint sets when experimented with 20 nodes having applied 20 iterations in each. Each constraint set has a different hop count value with different constraint parameters such as local search LS, history parameter HP, traffic change LS and fault tolerance parameter FT. In Fig. 2 horizontal bars for 20 nodes show that the optimum traffic evaluation varies with time due to the stochastic traffic conditions. It can be clearly observed that with the increasing in hop count, the probability of getting optimum path increases.

In the system, all the participating nodes share their information with the cloud at regular interval. Cloud infrastructure manages this traffic information with OpenStack LXCs. The cloud controller manages the overall traffic scenario of the network, where each node contributes to the system by sharing its local traffic information. The traffic pattern is observed and updated at a regular time interval in the cloud. In a cloud-based system, when an autonomous robotic car is placed in a vehicular traffic network for navigation between two points, it first observes the current traffic and sends a request, attaching its current location and destination to cloud infrastructure to provide the best approximate route to reach the destination at that particular time. The cloud controller returns back the optimum route to the robotic car for further navigation. Here, the robot's cooperation is undertaken by having different traffic data from the robots which are used to find the better path in the dynamic traffic.

In the experiments, three randomly vehicle traffic networks are generated with random time-varying probability. A directed network of a pre-specified number of nodes is created, where each node carries in or out degree in between one and four. The service is implemented in C and run on an instance with the configuration of Intel Core I7 2.2GHz with 4GB RAM under a Linux container of the cloud. The source and destination nodes are provided in the cloud at random to find an optimal route.

5. CONCLUSION

In this paper, we first discussed how an autonomous vehicle can be treated as a robot with an example of Google's self-driving car. Next, to exploit the collaboration and information sharing capabilities of the cloud infrastructure, DVTRP is formulated as a service for vehicular robots. Herein, the overall problem is formulated in the form of a dynamic graph network where traffic is changing stochastically. In the system, all the participating nodes share their information with the cloud at regular interval. Cloud infrastructure manages this traffic information with OpenStack LXCs. In order to investigate the working of the service, we have taken the synthetic dynamic traffic data with nodes. In the quest of an approximate optimum path, various parameters such as local search, hop count, history parameter and traffic change are introduced. Apart from this, several faulty paths are generated randomly to test the effectiveness and completeness of the proposed service. The network of nodes with 50 iterations each is analyzed to find the best approximate route to reach the destination.

REFERENCES

[1] I. F. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research,* vol. 170, pp. 677-709, 2006.

[2] J. Markoff, "Google Cars Drive Themselves, in Traffic," *New York Times,* vol. 9, 2010.

[3] H. N. Psaraftis, "Dynamic vehicle routing: Status and prospects," *annals of Operations Research,* vol. 61, pp. 143-164, 1995.

[4] W. B. Powell, P. Jaillet, and A. Odoni, "Stochastic and dynamic networks and routing," *Handbooks in operations research and management science,* vol. 8, pp. 141-295, 1995.

[5] S. Pemmaraju and S. Skiena, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®*: Cambridge university press, 2003.

[6] R. W. Hall, "The fastest path through a network with random time-dependent travel times," *Transportation science,* vol. 20, pp. 182-188, 1986.

[7] E. D. Miller-Hooks and H. S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks," *Transportation Science,* vol. 34, pp. 198-215, 2000.

[8] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5,* 2013.

[9] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering,* vol. 12, pp. 398-409, 2015.

[10] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: current status and open issues," *IEEE Access*, vol. 4, pp. 2797-2807, 2016.

[11] R. Doriya, N. Wadhwa, K. Suraj, P. Chakraborty, and G. Nandi, "Dynamic vehicle traffic routing problem: Study, implementation and analysis using ACO and GA," in *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on*, pp. 1164-1171, 2014.

[12] R. Doriya, P. Chakraborty, and G. Nandi, "'Robot-Cloud': A framework to assist heterogeneous low cost robots," in *Communication, Information & Computing Technology (ICCICT), 2012 International Conference on*, pp. 1-5, 2012.

[13] R. Doriya, P. Chakraborty, and G. Nandi, "Robotic Services in Cloud Computing Paradigm," in *Cloud and Services Computing (ISCOS), 2012 International Symposium on*, pp. 80-83, 2012.

[14] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems,* vol. 43, pp. 73-81, 1997.

[15] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research,* vol. 59, pp. 345-358, 1992.

[16] K. Jackson, *OpenStack cloud computing cookbook*: Packt Publishing Ltd, 2012.

[17] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley,* pp. 225-227, 1990.