Vehicle Route Planning using Dynamically Weighted Dijkstra's Algorithm with Traffic Prediction

Piyush Udhan[1,^,*], Akhilesh Ganeshkar[1,^], Poobigan Murugesan[1,^], Abhishek Raj Permani[1,^],
Sameep Sanjeeva[1], Parth Deshpande[1,a]

Abstract— **Traditional vehicle routing algorithms do not consider the changing nature of traffic. While implementations of Dijkstra's algorithm with varying weights exist, the weights are often changed after the outcome of algorithm is executed, which may not always result in the optimal route being chosen. Hence, this paper proposes a novel vehicle routing algorithm that improves upon Dijkstra's algorithm using a traffic prediction model based on the traffic flow in a road network. Here, Dijkstra's algorithm is adapted to be dynamic and time dependent using traffic flow theory principles during the planning stage itself. The model provides predicted traffic parameters and travel time across each edge of the road network at every time instant, leading to better routing results. The dynamic algorithm proposed here predicts changes in traffic conditions at each time step of planning to give the optimal forward-looking path. The proposed algorithm is verified by comparing it with conventional Dijkstra's algorithm on a graph with randomly simulated traffic, and is shown to predict the optimal route better with continuously changing traffic.**

Index Terms—**Dijkstra's algorithm, vehicle routing, traffic flow theory, traffic prediction model**

I. INTRODUCTION

Traffic congestion is a significant challenge in modern society, characterised by slower speeds, longer trip times, and increased vehicle queueing [1]. In a recent report by TomTom, a leading Global Positioning System (GPS) company, the issue of traffic congestion and the resultant loss of time, energy and money of the commuters in major metro cities around the globe is highlighted and the time lost is estimated to be greater than three-fourths of the actual travel time [2]. Traffic congestion costed a total of $160 billion in the United States of America (USA) due to 6.9 billion extra hours travelled and 3.1 billion additional gallons of fuel purchased in 2014 [3]. As per the INRIX Roadway Analytics in 2017, the most congested 25 cities of the U.S.A. are estimated to cost the drivers $480 billion, over the next 10 years, owing to lost time, wasted fuel, and carbon emitted during congestion. In 2018, the total cost of lost productivity due to congestion was found to be $87 billion in the U.S.A., highlighting the massive effect of traffic congestion on the global economy.

Municipalities, the world over, keep making heavy investments in building new roads, expanding the existing ones and repairing the damaged parts, although controversies on whether broader and a greater number of roads alleviate congestion persist [4]. Traffic congestion also causes increased vehicular emissions of harmful greenhouse gases such as CO_2, CO, SO_2, NO_2 which results in a significant rise of air pollutant exposures of commuters and urban populations due to the increased time spent in traffic [5]. Continued exposure to such harmful emissions can lead to severe respiratory problems and health risks [6][7].

One way to reduce traffic congestion is to enable traffic flow optimisation. This would provide insights into more efficient routing in conjunction to smart roads networks. Better routing would eventually lead to better utilisation of all the roads in the network, supporting optimal traffic flow and enabling a reduction in emissions.

Bellman-Ford [8], A* [9], Floyd Warshall [10] and Dijkstra's [11] are some of the common vehicle routing algorithms. One of the main advantages of Dijkstra's algorithm is its considerably low complexity, especially when dealing with a sparse road network. There are quite a few efforts studying the improvement of Dijkstra algorithm and its practical applications. One such effort attempted to solve the Optimal Route Planning in a Parking Lot based on the Dijkstra algorithm [12]. The authors combined an impedance function model with the traditional Dijkstra algorithm to obtain the dynamic time of route. A balance function between distance and time was adopted as the weight matrix. In another paper, modelling of the optimal route-finding problem was carried out using a combination graph with three cost components namely travel distance, toll costs and road surface conditions [13]. An optimal route was then calculated based on a combination of these weights fed to the Dijkstra in the form of an adjacency matrix. In a related study, the authors optimized Dijkstra's Algorithm from a control network scale [14]. Nordbeck proposed a shortest path algorithm based on the ellipse, which limits the search node collection in a certain area and greatly narrows the search scale.

Although the above studies tried to improve the efficiency of the Dijkstra algorithm, none of them actually solved the vehicle routing problem dynamically. The traffic density,

---

average speed on a patch of road, travel time and traffic flow keep changing dynamically, i.e., at nearly every instant. There is need for a forward-looking traffic model that can predict the various traffic parameters in a road network at future time-steps and feed those in the form of adjacency matrices to the routing algorithm. The routing algorithm itself needs to be modified beyond the conventional Dijkstra's algorithm, so that it can work with dynamic inputs and provide an optimal route calculated by taking into consideration the future traffic conditions in the road network.

In this paper, traffic flow theory is used for developing a new routing algorithm based on the traffic model of the road network. Consequently, Dijkstra's algorithm is used here over the A* algorithm, as it traverses through all the nodes of the network. Fundamental diagrams from traffic flow theory provide a good relation between traffic flow rate, traffic density and average speed, that helps model traffic well. This paper also uses the traffic flow conservation law to understand traffic flow at junctions and a Gaussian probabilistic modelling for external traffic inflows into the road network. Here, travel times are chosen as the weights in the adjacency matrix over which optimisation is carried out, instead of a common metric like distance. This is because travel time is an all-encompassing factor, calculation of which takes into consideration all the traffic parameters in the road journey.

The rest of the paper is structured as follows. Section II follows with the methodology used to modify Dijkstra's algorithm using traffic prediction. Section III discusses the implementation with the steps involved in generating and analysing the traffic. Section IV compares the proposed algorithm with Dijkstra's algorithm for various routes obtained from real datasets, with concluding remarks in Section V.

II. METHODOLOGY

A. Dijkstra's Algorithm

Dijkstra's algorithm is an algorithm used for finding the path with the minimum sum of weights between two nodes in a graph. A road network can be represented as a graph. Here, a 'node' represents a junction which connects two or more paths, and an 'edge' represents a road connecting the 'nodes' [11]. Figures 1 and 2 below depict how a graph containing nodes and edges is generated from an underlying road network [15].

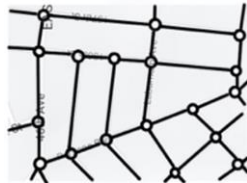<div style="display:flex">
Fig. 1. Actual Road Network   Fig. 2. Derived Graph
</div>

1) Modified Dynamic Weight Dijkstra's Algorithm: Conventional Dijkstra's algorithm helps find the shortest path between a source node and a destination node in a static, weighted graph. The algorithm entails visiting each neighbouring node and updating its value, which is a measure of the total weight of the shortest path to it from source, if a smaller distance is found in the iterative process. In this manner, the shortest path from the source node to every other node in a static graph can be found. Considering the graph shown in the Figure 3 as an example, where the source and destination nodes are enumerated as 0 and 9 respectively. The highlighted path, which has been arrived at by visiting neighbouring nodes and iteratively updating their values, is the least weighted route between the two nodes.
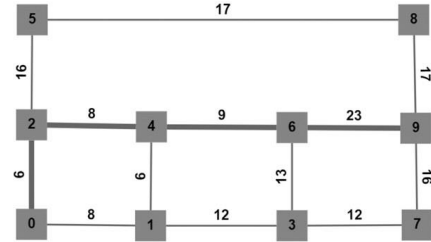

Fig. 3. Shortest Path on Weighted Graph using Dijkstra's Algorithm

For the problem statement this paper attempts to solve, the weights of the edges are dynamic and can change with time, in which is representative of a real traffic network. Here, the weight of an edge signifies travel time conforming to traffic flow theory. The inputs to the proposed algorithm are a graph and the source node number. The graph is in the form of a dictionary with the timestamps as keys and an adjacency matrix as the value at each key. The matrix is of Nth order, where N is the total number of nodes in the graph, and the value of adj[i,j] gives the travel time between nodes i and j at the time instant t as given by the key. This matrix is obtained as an output from the prediction model, which is outlined further in the paper.

The algorithm is an iterative procedure that repeatedly attempts to improve an initial approximation of the values of time[v]. The initial approximation is simply time[0] = 0 (as the source node is indexed as 0) and time[v] = ∞ for v = 1, 2, …, (n-1). In each iteration, a new node is processed, and its time[v] value is used to update the time[v] value of its immediate successors if the time to reach them is lower through the parent node v. Compared to standard Dijkstra's algorithm, the visited array is removed because this is a dynamic weights problem. So, there might arise a condition where at a later time instant, the weight of an edge falls, requiring the weights of all the connected nodes to be updated, even if they were previously visited. To keep a track of the path taken to reach a particular node minimally, there is a parent array, which is updated with the new parent node

if the path to reach the successor is lower through the new node. The details of this are shown in Algorithm 1.

Algorithm 1 Pseudocode for Proposed Routing Algorithm

Input: Graph(dict), source node

  Initialisation:

1: **for** each node *v* **do**
2:     time[*v*] ← ∞
3:     parent[*v*] ← NULL
4: **end for**
5: time[source] ← 0
6: push(0, source) → Q
    LOOP Process
7: **while** Q not empty **do**
8:     *u* ← node in Q with min time[*u*]
9:     remove *u* from Q
10:     **for** each neighbour *v* of *u* **do**
11:       alt ← time[*u*] + TRAVELTIME*[u,v]* at time*[u]*
12:       **if** alt < time[*v*] **then**
13:         time[*v*] ← alt
14:         parent[*v*] ← *u*
15:         push(alt, *v*) → Q
16:       **end if**
17:     **end for**
18: **end while**
19: **return** time[ ],parent[ ]

In this case, Dijkstra's algorithm would use the travel time[*u, v*] at time t = 0 which would give an initial path as well as a travel time. However, the traffic parameters would have changed by the time a node *u* is reached. Hence, the travel time at that time instant, which is the output of the prediction model considering updated traffic parameters, is considered in this implementation.

B. Prediction model

1) Traffic Flow Theory: Traffic flow theory is used to understand various parameters that help define the flow of traffic. The change in traffic parameters of a particular edge over time is attributed to the flow of traffic in and out of the corresponding lane. Each lane is represented as an edge and its weight is defined as the time taken to traverse the lane, which is directly dependent on the traffic in that lane and a few other factors as explained below.

The travel time of the vehicle through a particular lane depends on the length of the lane and the average speed, which in turn depends on the traffic in that lane. Traffic flow theory gives a relationship between the traffic flow rate *q*, density *k* and speed *u* given by

$$q(i,j) = k(i,j) \cdot u(i,j), \qquad (1)$$

where, $q(i,j)$ is the traffic flow rate between nodes *i* and *j*, defined as the average number of vehicles passing a specific point per unit time. Similarly, $k(i,j)$ is the traffic density, defined as the average number of vehicles occupying a section of the road per unit distance, and $u(i,j)$ is the space mean speed defined as the spatial average speed of all vehicles.

The conservation law states that the measurable parameters of an isolated physical system remain constant. In this system, the conserved parameter is the number of vehicles on a given stretch of road. Since the number of vehicles on a given stretch of road between [*x, x'*] is constant, for there to be a change in the traffic density(ρ), there must be an inflow or an outflow, such that the traffic flow is conserved, as illustrated in Figure 4.
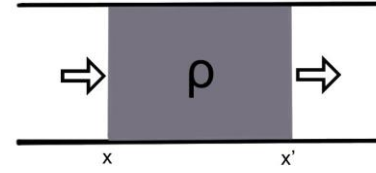

Fig. 4. Conservation Law Illustration

In 1955, Lighthill, Whitman and Richards proposed a macroscopic traffic flow model known as the LWR model [16][17]. According to this model, if a vehicle can be assumed to be a molecule, then the traffic in the lane can be presented as an incompressible fluid which cannot be compressed beyond a certain density. Based on the conservation laws and the relation between the traffic variables, traffic flow is represented by the model as follows

$$\frac{\partial k}{\partial t} + Q'(k) \cdot \frac{\partial k}{\partial x} = f(), \qquad (2)$$

where, *k* is the density, *Q* is the flux or the flow rate, and *f*() represents the source term which represents the inflow and outflow.

This methodology involves the representation of the system in a state space form. The associated model then uses the conservation law and dynamic speed formula with a linear traffic stream model to study the traffic flow. The density and aggregate space mean speed are considered to be the state variables of interest in this model. The relative difference in the entry and exit flows is taken as the input variable and the space mean speed is the output variable. The classified volume per minute and spot speeds are extracted from the data collected, and the traffic variables, flow and space mean speed, are calculated. The actual densities are calculated using an input-output analysis.

Hence, the traffic flow at the next timestep is determined by considering the present inflows and outflows.

III. IMPLEMENTATION

A. Prediction Pipeline

In this section, the architecture of the proposed prediction model is briefly introduced. The prediction model consists of two parts, data simulation and prediction, as shown in Figure 5. The road network is forwarded to the data simulation component in a suitable format, which is then used to generate data at time t = 0 and pass it to the prediction component for predicting traffic flow at future time instances. The components are designed to be independent of each other.
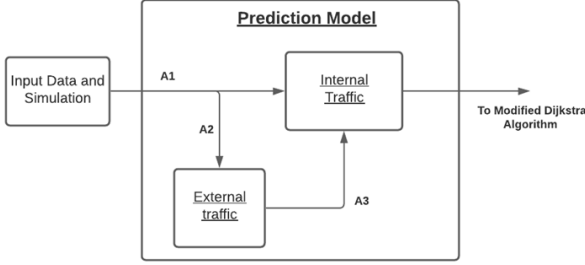

Fig. 5. Prediction Model Pipeline

B. Input Data and Simulation

A road network represented in the form of a graph is fed as an input to the model. All the information about the graph is organised into two files: one providing information about the nodes and the other about the edges and edge weights. Table # provides details about various classes used in the model and their significance in the model. The classes described as basic requirements are required for expressing the road network in the form of a graph. The table also shows the parameters that are used as outputs.

Fig. 6. Class Information

At time t = 0, the traffic data simulated consists of vehicle positions that are generated randomly over the graph by specifying the number of vehicles as an argument. Vehicles that enter the network from outside it are characterized as external traffic, and the vehicle data generated at every node has an open edge for external traffic inflow. Thus, at time t = 0, the graph and traffic data are provided to the prediction component.

C. Prediction Component

The task of the prediction component is broken down as predicting the future states of external and internal traffic.

1) External traffic: A Gaussian probability distribution function (GPDF) is used here to simulate external traffic. GPDFs are used to represent the probability distribution function of a normally distributed random variable, with a mean μ and variance σ^2.

The Gaussian equation is given by

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \forall x \in X \quad (3)$$

Where, μ is the mean, σ^2 is the variance, and X is the domain of the random variable.

High-order Gaussian process dynamical models for traffic flow prediction [18] and a probabilistic model fusing multisource data on road traffic [19] are two models which use Gaussian models extensively for both prediction and data aspects of their model. These have provided valuable insights and superior results regarding the use of Gaussian models in the vehicle routing problem. Hence, this study uses the Gaussian model.

It is assumed that external traffic enters the road network from a node at time t = 0. The value corresponding to the number of vehicles about to enter the graph from a particular node is used as the mean value for the GPDF, and the variance is initialised in the range of 0 to 3. This is done for every node at which external traffic can enter the road network. After every time instance, this step is repeated, and the number of vehicles generated in the last iteration is assigned as the new mean for the GPDF for that node. Consequently, external traffic is generated at nodes in the graph and passed on to the internal traffic component.

TABLE I
PREDICTION MODEL

Step	Input	Principles	Output
1	Vehicle positions Edge Data	Flow conservation	Traffic density
2	Traffic Density Edge Data	Fundamental Diagrams	Average Speed Traffic flow
3	Average Speed	Distance-velocity	Travel time

	Edge Distance/weight	-time equation	

2) Internal Traffic: This section covers how internal traffic is predicted using the data provided at time $t = 0$. The graph (road network) is received via data buses A1 and A3 from the Data and External traffic components, respectively, as shown in Figure 5.

The steps for predicting the internal traffic are as follows.

Step 1: Taking vehicle positions and edge data as arguments, the model applies traffic flow conservation. Using the vehicle positions, it calculates the number of vehicles present along every edge. Furthermore, using vehicle characteristics (like thickness) and edge characteristics, the traffic density is calculated at every edge. At this stage, the value assigned to the edge, i.e., the traffic along that edge, is the ratio of the number of vehicles to the edge thickness.

Algorithm 2 Determine Speed of Vehicles

Input: Traffic density

    Initialisation:

1: $v_{ret} \leftarrow$ free-flow speed of the edge, where the vehicle is present

2: curr_tr_dens $\leftarrow$ traffic density × vehicle thickness

3: curr_tr_dens $\leftarrow \dfrac{\text{curr\_tr\_dens}}{\text{edge\_length}} \times$ edge_thickness

4: **if** curr_tr_dens $< \beta$ & curr_tr_dens $\le \alpha$ **then**

5:      $v_{ret} \leftarrow$ (edge.free_flow_speed - edge.jam_speed) × $(1 -$ curr_tr_dens$) +$ edge.jam_speed

6: **else if** curr_tr_dens $> \beta$ **then**

7:      $v_{ret} \leftarrow$ jam speed of edge is assigned

8: **end if**

9: **return** v_{ret} {speed value for vehicle returned}

10: **return** time[ ], parent[ ]

Step 2: Using the traffic densities, the speed of every vehicle is calculated using the function shown in Algorithm 2. The average speed is then calculated by taking the mean of the speed of all vehicles along that particular edge.

Step 3: Using the average speed of an edge and the edge length, the travel time for a particular edge is calculated and assigned as the edge-weight for that particular edge.

3) Other background processes: When a vehicle reaches the end of an edge, i.e., it reaches a node X, it needs to decide which edge it will travel along next. Hence, the model then analyses the traffic in all the edges directly connected to node X, by calling the function described in Algorithm 4. In doing so, a probability distribution is generated that is used to select the

next edge to travel to. A vehicle can also exit the network when it reaches an open node and chooses an edge with (id - 1).

Output: For every time instant, the predicted travel time for every edge is assigned as the weight of that edge, and the transformed graph is then passed on to the routing algorithm for route planning.

Algorithm 3 Transformation Function Logic

Input: Edge traffic

    {two thresholds predefined, α and β where $\beta > \alpha$, α and β are the thresholds for edge states

    - traffic less than $\alpha \Rightarrow$ edge is in free flow state

    - traffic greater than α and less than $\beta \Rightarrow$ edge is in normal flow state

    - traffic greater than $\beta \Rightarrow$ edge is in jam state

    based on state, a quadratic functional diagram transformation is applied and the value is stored in *result*}

1: **return** *result*

Algorithm 4 Algorithm for Selecting Next Edge

Input: node X

1: $K \leftarrow$ dictionary with edge id as key and traffic situation as value

2: sum $\leftarrow 0$

3: **for** every edge E directly connected to node X **do**

4:      tr $\leftarrow$ get value of traffic in that edge

5:      **if** node $X \equiv$ start node of edge E **then**

6:          tr $\leftarrow$ transformation_function(E.forward traffic)

7:      **else**

8:          tr $\leftarrow$ transformation_ function(E.backward traffic)

9:      **end if**

10:      $K[E.\text{id}] =$ tr

11:      sum $+= K[E.\text{id}]$

12: **end for** {Normalising values in K}

13: **for** every edge E.id in K **do**

14:      $K[E.\text{id}] / =$ sum

15: **end for**{Using these values assigned for every directly connected edge in dictionary K as probabilities}

16: choice = random.choices(node connections, weights=k) {this returns chosen edge id}

17: **return** choice

IV. EXPERIMENTATION, RESULTS AND OBSERVATIONS

In the dataset [20] used for testing, the road networks of five major cities of the globe are expressed in the form of a graph. The junctions are exhibited as nodes, and the roads connecting them as edges. The rationale behind using this dataset for road networks is to calibrate and evaluate the model on data which mimics real life road networks and traffic conditions as much as possible. As this dataset is designed to replicate actual road networks, it serves as an ideal data feed for the model.
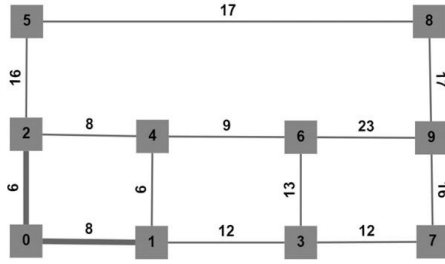

Fig. 7. Road Network at t = 0

As covered in the input data and simulation component of the prediction pipeline in Section #, random traffic data is generated over the road network. The model is tested with different
⟨source_node, destination_node⟩ pairs to get the total travel time as an output for both, the model proposed in this paper and Dijkstra's algorithm.

In the case of conventional Dijkstra's algorithm, the shortest path is computed by applying the unmodified routing algorithm on the graph shown in Figure 7 at t = 0, without considering the dynamic nature of traffic and resulting dynamic edge weights.

Figures 8 and 9 depict the implementation of the proposed routing algorithm. This algorithm considers the dynamic nature of traffic over the road network in subsequent timestamps. It computes the total time required for every route to reach destination node by considering the traffic at future time steps from the prediction model. All the possible edge combinations at different time stamps are computed to obtain the shortest route.

As seen in the graph, there are two possible paths from the source node 0, E_{0-1} and E_{0-2}. The proposed algorithm evaluates each of these two edges at t = 0, and then further evaluates the following edges at corresponding future time stamps based on the dynamic traffic and edge weight inputs from prediction model. The detailed computation data is presented in Table #.
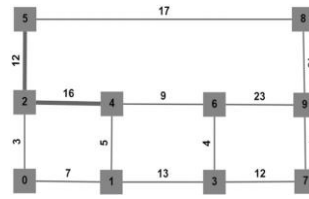

Fig. 8. Road Network at t = 6   Fig. 9. Road Network at t = 8
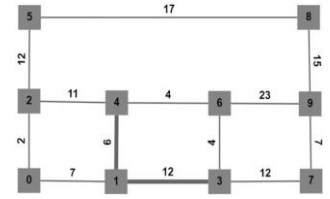
The graph changes at every time instant, as the weights of the edges of the graph change with time. The proposed routing algorithm evaluates all the dynamic weights and time stamp permutations to determine the shortest path from source to destination, as shown in Figures 10 and 11.
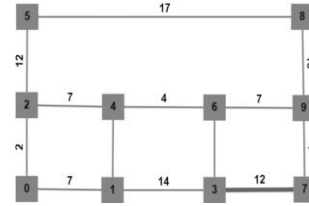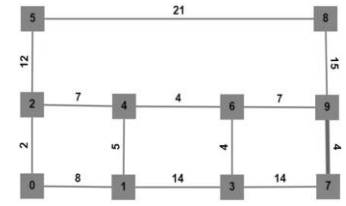

Fig. 10. Road Network at t = 20   Fig. 11. Road Network at t = 32

Table # shows the weights of the edges as rows and time stamps as columns. The value in each cell denotes the weight of that edge at a particular time instant. The change from one time stamp to another can be considered as a transition. The eights of those edges that are permissible and are under evaluation at that time stamp are highlighted in bold and underlined. The weights of those edges that constitute the optimal path selected by the proposed algorithm are circled in red.

Edges / Time Stamp	t = 0	t = 6	t = 8	t = 14	t = 18	t = 18	t = 20	t = 32
E_{0-1}	**8**	1	1	8	8	8	1	8
E_{0-2}	**6**	3	2	3	2	2	2	2
E_{1-3}	12	13	**12**	16	15	15	14	14
E_{1-4}	6	5	**6**	5	5	5	5	5
E_{2-4}	8	**16**	11	11	1	1	7	7
E_{2-5}	16	**12**	12	12	12	12	12	12
E_{3-6}	13	4	4	4	4	4	4	4
E_{3-7}	12	12	12	14	16	16	**12**	14
E_{4-6}	6	6	4	**4**	4	4	4	4
E_{5-8}	17	17	17	17	**17**	17	17	21
E_{6-9}	23	23	23	23	23	**23**	7	7
E_{7-9}	16	7	7	11	7	7	7	**4**
E_{8-9}	17	15	15	15	15	15	15	15

Fig. 12. Dynamic Weights of Edges on Transitions

A. Comparison

Comparing results from Dijkstra's algorithm and the routing algorithm proposed in this paper, the final paths selected by both can be seen in Figures 13 and 14, respectively.
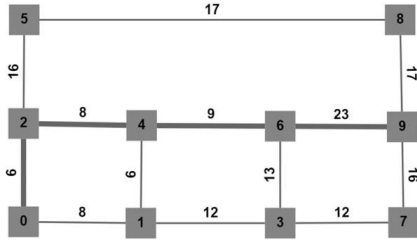
Fig. 13. Shortest path using conventional Dijkstra's algorithm

A comparative analysis of the above results is carried out next. When using Dijkstra's algorithm, the
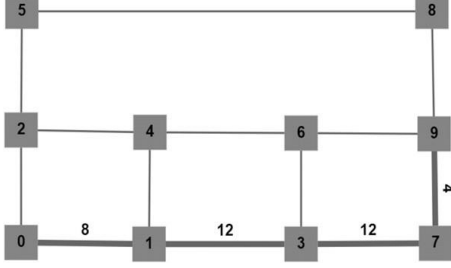

Fig. 14. Shortest path using the proposed routing algorithm

shortest path has a total travel time of 46 units. Whereas when using the proposed dynamic routing algorithm with from the traffic prediction model, the shortest path is [0 → 1 → 3 → 7 → 9], with a total travel time of 36 units. This comparison shows that the final path determined by the proposed algorithm is shorter in terms of travel time as compared to the one obtained without considering predicted traffic flows. In this case, the new model leads to a 21.7% decrease in travel time by simply looking forward in time for traffic movement.

B. Further Experiments and Results

As mentioned earlier, the road network for the city of California is used in the form of a graph as an input to the model. From the total of 21,047 nodes and 21,692 edges present in the graph a subset of 80 nodes and 150 edges are chosen as our experimentation set.

In the input data and simulation component of the prediction pipeline, random traffic data is generated over the road network. The model is tested with the specified graph and around 10,000 different ⟨source_node, destination_node⟩ pairs to get the average difference between the least travel time returned by the proposed routing algorithm and conventional Dijkstra's algorithm. The average difference parameter is given by

$$\lambda = \sum_{i=1}^{N} \frac{T_i - \tau_i}{N},$$

(4)

where,

λ = Average difference

τ_i = Output travel time by proposed algorithm for ith test case

T_i = Output travel time by Dijkstra's algorithm for ith test case

N = Total number of test cases

The average travel time difference is recorded to be 3.87971 units (1 unit = 1000 s). Model performance was also analysed for different values of α and β, and the results achieved are shown in Figure 15. This depicts the significant deduction in travel time achieved when using the proposed routing algorithm with traffic prediction as compared to conventional Dijkstra's algorithm.

Note: In α-β experimentation, the arbitrariness in results can be attributed to a difference in choice of next edge for the vehicle between simulations for different α-β values.
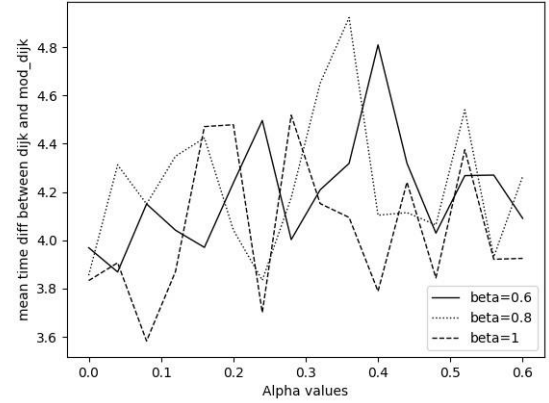

Fig. 15. α-β Experimentation

V.   CONCLUSION

This paper extended Dijkstra's algorithm to solve the dynamic vehicle routing problem by incorporating traffic movement prediction in the planning stage. It was shown that the weights derived from the traffic flow data dynamically affect the calculations in Dijkstra's algorithm based on time. The simulation results showed that compared to standard Dijkstra's algorithm that only considers static traffic data, the proposed routing algorithm with traffic prediction gives optimal routes with a significant reduction in travel time with flowing traffic. This algorithm can be modified in the future for several applications and specializations of the vehicle routing problem by adding constraints pertaining to specific use cases.

REFERENCES

[1] Caves, R.W. (2004). Encyclopedia of the city. Routeledge. P. 141

[2] C¸olak, S., Lima, A. & Gonzalez,´ M. Understanding congested travel in urban areas. Nat Commun 7, 10793 (2016).
https://doi.org/10.1038/ncomms10793

[3] Afrin, T.; Yodo, N. A Survey of Road Traffic Congestion Measures towards a Sustainable and Resilient Transportation System. Sustainability 2020, 12, 4660. https://doi.org/10.3390/su12114660

[4] Braess, D., Nagurney, A. & Wakolbinger, T. On a paradox of traffic planning. Transport. Sci. 39, 446–450 (2005).

[5] Han Xue, Shan Jiang, Bin Liang, "A Study on the Model of Traffic Flow and Vehicle Exhaust Emission", Mathematical Problems in Engineering, vol. 2013, Article ID 736285, 6 pages, 2013.
https://doi.org/10.1155/2013/736285

[6] Kai Zhang, Stuart Batterman, Air pollution and health risks due to vehicle traffic, Science of The Total Environment, Volumes 450–451, 2013, Pages 307-316, ISSN 0048-9697, https://doi.org/10.1016/j.scitotenv.2013.01.074.

[7] Zuurbier, M., Hoek, G., Oldenwening, M., Meliefste, K., van den Hazel, P., & Brunekreef, B. (2011). Respiratory Effects of Commuters' Exposure to Air Pollution in Traffic. Epidemiology, 22(2), 219–227. http://www.jstor.org/stable/23047537

[8] R.E. Bellman, "On a routing problem",Quart. Appl. Math. 16 (1958) 87–90.

[9] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968, doi: 10.1109/TSSC.1968.300136.

[10] Robert W. Floyd. 1962. Algorithm 97: Shortest path. Commun. ACM 5, 6 (June 1962), 345. DOI:https://doi.org/10.1145/367766.368168

[11] E.W. Dijkstra, "A note on two problems in connection with graphs",Numer. Math. 1 (1959) 269–271.

[12] Yujin, Ge Xiaoxue (2017). Optimal Route Planning of Parking Lot based on Dijkstra Algorithm. International Conference on Robots and Intelligent System (ICRIS). doi: 10.1109/ICRIS.2017.62

[13] S. Kirono, M.I. Arifianto, R.E. Putra, A. Musoleh, Riani Setiadi (2018). Graph-based modeling and dijkstra algorithm for searching vehicle routes on highways. International Journal of Mechanical Engineering and Technology 9(8), 2018, pp. 1273-1280.

[14] Haitao Wei, Shusheng Zhang and Xiaohui He (2021). Shortest Path Algorithm in Dynamic Restricted Area Based on Unidirectional Road Network Model. Sensors 2021, 21, 203. https://doi.org/10.3390/s21010203 [15] Google Maps.

[16] Lighthill, M.J. and Whitham, G.B. (1955). On Kinematic Waves. II. A Theory of Traffic Flow on Long Crowded Roads. Proceedings of the Royal Society of London. Series A, 229, 317-345.
http://dx.doi.org/10.1098/rspa.1955.0089

[17] Ali, A. and Andallah, L. (2016) Inflow Outflow Effect and Shock Wave Analysis in a Traffic Flow Simulation. American Journal of Computational Mathematics, 6, 55-65. doi: 10.4236/ajcm.2016.62007

[18] Jing Zhao, Shiliang Sun (2016). "High-Order Gaussian Process Dynamical Models for Traffic Flow Prediction," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, pp. 2014-2019, July 2016, doi: 10.1109/TITS.2016.2515105.

[19] L. Lin, J. Li, F. Chen, J. Ye and J. Huai, "Road Traffic Speed Prediction: A Probabilistic Model Fusing Multi-Source Data," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 7, pp. 1310-1323, 1 July 2018, doi: 10.1109/TKDE.2017.2718525.

[20] Dataset: https://www.cs.utah.edu/ lifeifei/SpatialDataset.htm