Report: Optimizing Vehicular Routing using the Selfless Driving Model

Phuc Le
Cypress Woods High School, Cypress, TX 77429, USA
Department of Computer Science, University of Houston, Houston, TX 77004, USA
jackle61333@gmail.com

***Abstract*—With increasing traffic and vehicular autonomy, dynamic vehicular route guidance is crucial to minimizing congestion for networks and travel time for drivers. Traditional route guidance algorithms (RGA) direct vehicles along the least-cost path from origin to destination without regard for network conditions. In this report, we present various approaches to creating a dynamic RGA that aims to utilize network data to minimize the average travel time of all vehicles in the network, while also ensuring that vehicles' travel deadlines are met. The efficiency of the algorithms are then validated using a test bed based on Simulation of Urban MObility (SUMO), with the experimental results proving a clear performance improvement over traditional guidance algorithms.**

I. Introduction

Traditional RGAs guide vehicles by calculating the fastest and/or shortest path to travel from origin to destination with little regard for local network conditions. Because of this, we refer to it as a "selfish" policy. However, with increasing traffic and larger scaled networks, traditional RGAs may not be globally optimal if all vehicles are to be considered. Under the traditional RGA, multiple vehicles starting from similar origins with similar destinations will select similar routes, leading to significant congestion. This issue can be avoided with the use of a dynamic RGA, which aims to guide vehicles along globally optimal routes instead of individually optimal routes. Because this does not guarantee the most optimal path for each individual vehicle, we refer to it as a "selfless" policy.

In this report, we propose two algorithms that aim to emulate the selfless policy and minimize global average travel time. The first algorithm is a semi-selfish policy based on the traditional RGA. The second is a semi-selfless policy that aims to select globally optimal routes taking into account estimated network travel time data.

Both algorithms will then be tested on a SUMO-based test bed to validate their efficiency and compare them to the traditional RGA.

II. A Comparison of The Two Policies

As previously defined, a selfish policy is one where individual vehicles select routes that are optimal for itself, with little consideration for the conditions of the global network. A selfless policy is one where individual vehicles select globally optimal routes that meet its travel deadlines, even if it may be detrimental for itself, with the aim minimizing the global average travel time.

Under the selfish policy, each vehicle will choose the fastest route for itself, in the likely hopes that it will also satisfy its travel deadline. Conversely, under the selfish policy, each vehicle will choose its route to both ensure that its travel deadline is satisfied and to reduce the average travel time of all vehicles within the network.

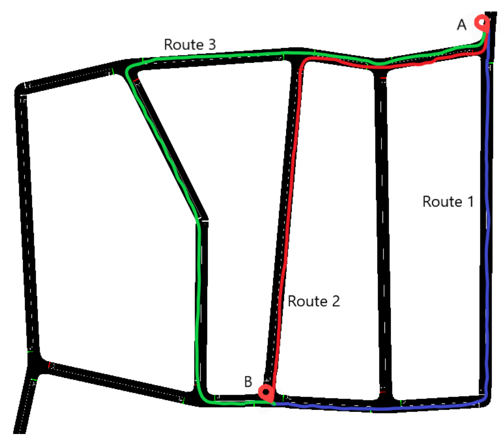
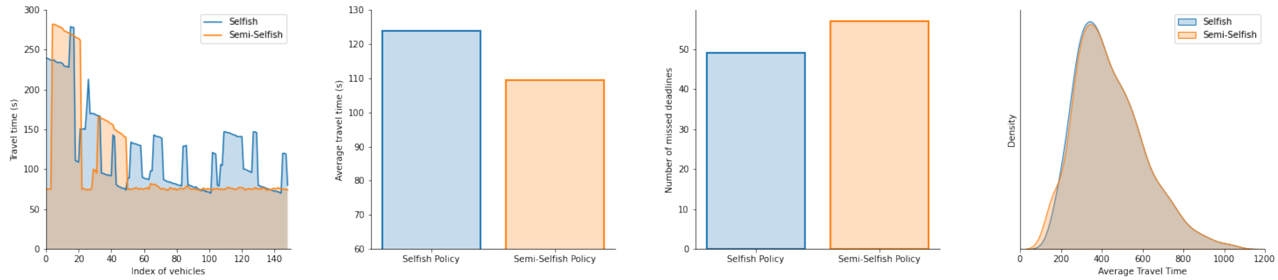


Fig. 1: Simulated road network generated from SUMO

For example, in Fig. 1, there are three possible routes from the origin point A to the destination point B. Route 2, indicated by the red line, is shortest in distance. It is followed closely by Route 1, indicated by the blue line. Finally, Route 3, indicated by the green line, is longest in distance.

Under the selfish policy, vehicles will tend to choose Route 2 because it is shortest in distance. This will inevitably cause congestion in that route, significantly increasing the travel time of subsequent vehicles. Under the selfless policy, vehicles will split themselves between the three routes before the congestion even occurs. Vehicles with urgent deadlines will be directed through Routes 1 and 2, the shortest paths. Vehicles with longer deadlines will be routed through Route 3, allowing for the other two routes to be open, thereby reducing congestion.

Although this means that some vehicles may spend more time traveling than they would under the selfish policy, the trade-off overall benefits the entire network by achieving a global optimum that minimizes congestion, thereby also minimizing the average travel time of the global network.



(a) The travel time of each vehicle in Γ

(b) Average travel time of all vehicles in Γ

(c) Total deadlines missed in Γ

(d) Distribution of average travel times over 1000 sets Γ

Fig. 2: Visual results of Semi-Selfish Policy evaluation

III. The Semi-Selfish Policy

A. Explanation of the Algorithm

The main components of the proposed semi-selfish policy are very similar to traditional selfish policies. For each vehicle in the network, an individually optimal route is generated using a traditional Dijkstra's algorithm, which finds the shortest, or in this case the least-cost, path from origin to destination. This route is then sent as instructions for the vehicle to execute.

However, unlike a fully selfish policy, which only takes the length of the road into consideration, we modify the algorithm to also take into account the density of a road. The density of a road is defined as the quotient of the number of vehicles currently on that road and the length of the road. It is calculated as:

$$\delta_i = \frac{n_i}{l_i} \tag{1}$$

where δ_i is the density of the ith edge, n_i is the number of cars on the ith edge, and l_i is the length of the ith edge.

This density is then weighted against the length of an edge to calculate the final weight of that edge. The final weight of an edge is defined as:

$$w_i = max(l_i, l_i * (\lambda\delta_i)) \tag{2}$$

where w_i is the weight of the ith edge and λ is the constant that defines the weight of the density in the overall weight of the edge. For testing, λ was set to 100. The purpose of $max()$ is so that edges still have a weight even if their density is zero.

We consider this policy to be a mixture of selfish and selfless, though more selfish than selfless because it will still choose congested routes if the length of the alternate open route outweighs the calculated density. Hence the reason why we call this policy a Semi-Selfish Policy. In addition, even though there are considerations over the conditions of the global network, vehicles are still choosing individually optimal routes instead of globally optimal routes.

B. Evaluation Procedure

To evaluate the efficiency of the semi-selfish policy, we will compare its results to the fully selfish policy using an identical set of vehicles. The algorithm used in the fully selfish policy is a basic Dijkstra's algorithm, using the travel distance as its weights. The algorithm used in the semi-selfish policy is the one detailed in the section before. Unless otherwise indicated, the road network used is the one in Fig. 1. The testing set is assumed to be a vehicle set Γ with 150 controlled vehicles and 50 random SUMO-generated vehicles. The plots will only show the data for the controlled vehicles.

C. Results

In Fig. 2(a), the index of a vehicle is based on its order of arrival to its destination. Vehicles with lower indices arrive earlier than those with higher indices.

Under the fully selfish policy, multiple spikes in travel time can be seen. This is most likely because of how the policy prioritizes the shortest paths. Every vehicle will always be routed towards the shortest path from their origin to their destination, leading to significant congestion along connecting routes. As more vehicles choose a path, the travel time of following vehicles increases. Once vehicles begin reaching their destinations and congestion clears, the cycle begins again with a new set of vehicles picking shortest paths.

In contrast, under the semi-selfish policy, there are only two large spikes followed by a long period of relatively average travel times. This is because, in the beginning when there are no vehicles on the network, the algorithm will direct vehicles along the shortest path, leading to significant congestion in those paths. However, this also means that following vehicles will directly avoid those congested paths and be routed towards alternate routes that reduce their overall travel time.

As shown in Fig. 2(b), the average travel time of all vehicles in Γ under the semi-selfish policy is approximately 13% lower that of the fully selfish policy. This is most likely because, while there is significantly less deviation in travel times for the semi-selfish policy, it has a larger initial congestion that nearly outweighs the efficiency improvement for the rest of the vehicles.

Fig. 2(c) reveals a small caveat with this semi-selfish policy. The semi-selfish policy had 57 deadlines missed while the fully selfish policy had 49 deadlines missed. The large majority of vehicles with missed deadlines for the semi-selfish policy are those who got stuck in the initial larger congestion. In contrast, the vehicles with deadlines missed in the fully selfish policy is distributed across the vehicle set, mostly centered around each spike.

Fig. 2(d) shows a distribution plot of results over 1000 trials, with 1000 different vehicle sets Γ tested against both policies. For a vast majority of cases, the semi-selfish policy performs very similarly to the fully selfish policy, differing for only 6.5% of vehicle sets. The semi-selfish policy performs better in cases where the average travel time is relatively low, at below 200 seconds. It performs marginally worst in cases where the average travel time is between 200 and 400 seconds. This is likely because of the previously mentioned larger initial congestion along routes. However, on average, in cases where the two policies differed, the semi-selfish policy performed 269% better than the fully selfish policy.

IV. The Semi-Selfless Policy

A. Travel-time-based weights

Similar to the semi-selfish policy, the proposed algorithm attempts to minimize average travel time and deadlines missed by modifying the weights of each edge of the network. However, instead of defining weights based on density, the weights are instead defined by the travel time across that edge.

The estimated travel time across an edge i is calculated using the number of cars on edge i, an estimation of the average velocity across edge i, and the length of edge i. There are two types of scenarios that may occur at any given edge:

Type 1. One or more vehicles are currently on edge i.

Type 2. No vehicles are currently on edge i.

For Type 1, it is possible to calculate the average velocity across i using the number of vehicles on the edge. As such, the average velocity across the ith edge for Type 1 is calculated as:

$$v^i_{avg} = \frac{1}{N_i} \sum_{k=1}^{N_i} v^i_k \tag{3}$$

where v^i_{avg} is the average speed across the ith edge, N_i is the number of vehicles on the ith edge, and v^i_k is the speed of the kth vehicle on the ith edge.

For Type 2, it is not possible to calculate the average velocity across i using the number of vehicles on the edge. As such, the average velocity across the ith edge for Type 1 is calculated as:

$$v^i_{avg} = V^i_{max} \tag{4}$$

where V^i_{max} is the maximum allowed speed of vehicles across the ith edge.

The travel time across the ith edge is then calculated as:

$$t_i = \frac{l_i}{v^i_{avg}} \tag{5}$$

where l_i denotes the length of the ith edge.

Finally, the weight of the ith edge is denoted as $w_i = t_i$, where w stores the weight of every single edge in the network.

B. Explanation of the Algorithm

The proposed semi-selfless policy is largely based off of a traditional route finding algorithm, the Floyd-Warshall algorithm. In this algorithm, given a directed graph G with n vertices, it computes the shortest path between each pair of vertices i and j. This shortest path is stored in a distance matrix δ, where δ_{ij} stores the length of the shortest path between vertices i and j. For our algorithm, a slight modification is made. Instead of i and j denoting vertices, they instead denote edges, where δ_{ij} stores the shortest path between *edges* i and j. The computation for this is done as follows:

- For every decision making phase k, where $k < n$, δ_{ij} stores the shortest length of edge pairs i and j at the kth phase.
- In the kth phase, the algorithm considers the alternate route, traversing from edge i to edge k then to edge j. The algorithm then compares this to the original route from edge i to edge j and updates δ_{ij} accordingly.

To make this algorithm usable for our scenario, we include a predecessor matrix ρ, where ρ_{ij} stores the edges of the shortest path from edge i to j.

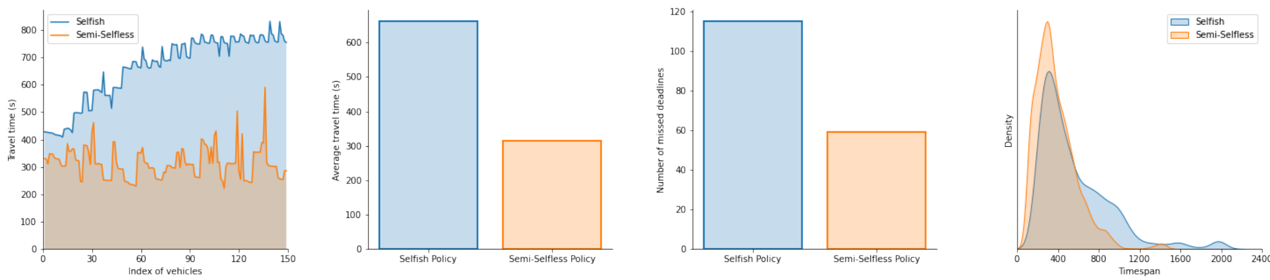
At every decision step, the weights are recalculated by the equations defined in the previous section. Then, using those weights as the costs of each edge, the modified Floyd-Warshall algorithm is used to calculate the best paths from every edge i to every edge j. Then, for all vehicles in the set, the optimal path is reconstructed using the predecessor matrix ρ, which is then returned as instructions for the vehicle to take.

The Floyd-Warshall algorithm was picked because of its computational efficiency and adaptability to new situations. As opposed to a traditional Dijkstra's algorithm where computations will be repeated as every vehicle must traverse the network by itself and construct the optimal path, Floyd-Warshall can precompute all the paths based on the travel times at that exact moment. In addition, if a vehicle were to decide to change destination, the path can easily be adjusted without recomputing the best possible path.

We also consider this policy to be a mixture between selfish and selfless. However, we believe it to be more selfless than the previously presented policy. This policy makes the network conditions central to its routing, instead of as a weighted factor like with the semi-selfish policy. Although it will be achieved indirectly, this policy will get much closer to the global optimum achieved by a fully selfless policy than the previously presented semi-selfish policy. Hence the reason why we consider this to be a semi-selfless policy.

C. Evaluation Procedure

To evaluate the efficiency of the semi-selfless policy, we will compare its results to the fully selfish policy using an identical set of vehicles. The algorithm used in the fully selfish policy is a basic Dijkstra's algorithm, using the travel distance as its weights. The algorithm used in the semi-selfless policy is the one detailed in the section before. Unless otherwise indicated, the road network used is the one in Fig. 1. For Fig. 3, the



(a) The travel time of each vehicle in Γ'

(b) Average travel time of all vehicles in Γ'

(c) Total deadlines missed in Γ'

(d) Distribution of average travel times over 1000 sets Γ'

Fig. 3: Visual results of Semi-Selfless Policy evaluation

testing set is assumed to be a vehicle set Γ', containing 150 controlled vehicles and 50 random SUMO-generated vehicles. To further test all indicative traffic scenarios, we will use testing set Π, which will have differing numbers of vehicles controlled by the policy and the number of vehicles randomly generated by SUMO, as well as the number of vehicles in general. All plots will only show the data for the controlled vehicles. For each ratio of vehicles in Π, we will test it against 100 distinct vehicle sets to eliminate possible sources of error. Due to computational resource and time constraints, we will not be able to test all scenarios in Π with a large vehicle set. These sets Π are defined as follows:

TABLE I: Testing sets Π that will be used

Total Vehicles	**Controlled**	**SUMO-generated**
80 vehicles	20 vehicles	60 vehicles
80 vehicles	40 vehicles	40 vehicles
80 vehicles	60 vehicles	20 vehicles
160 vehicles	80 vehicles	80 vehicles

D. Results for Γ'

In Fig. 3(a), the index of a vehicle is based on its order of arrival to its destination. Vehicles with lower indices arrive earlier than those with higher indices.

Under the fully selfish policy, as before multiple spikes in travel time can be seen. However, this time there is a clear general upwards trend as time goes on. This is most likely because, under the fully selfish policy, vehicles will continue to pick the shortest path without regard for network conditions. This means that even if a route is heavily congested, vehicles will continue to pick that path, leading to further congestion and slower travel times for following vehicles. It is interesting to note that, although this trend is expected and is seen in this simulation set, it was not observed in the previous evaluation with the semi-selfish policy.

In contrast, under the semi-selfless policy, there is a general horizontal trend, with each vehicle having near average travel times. This is most likely due to the architecture of the algorithm itself. When taking travel time into consideration, following vehicles will avoid congested paths that they know will take a long time to traverse across. This means that once a road has already been picked by a certain number of vehicles, following vehicles will always pick alternate, faster routes. This is clearly seen in the graph in the multiple spikes, indicating the vehicles stuck within the initial areas of congestion, and the rapid drop following the spikes, indicating the following vehicles that picked alternate routes.

As shown in Fig. 3(b), the average travel time of all vehicles in Γ' under the semi-selfless policy is approximately 112% lower that of the fully selfish policy. It is clear from this that by taking network information into account to avoid areas of congestion, the average travel time of all vehicles can be significantly reduced.

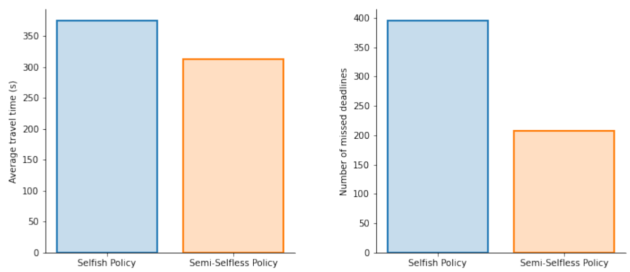
Efficiency improvements are further seen in Fig. 3(c), which compares the total deadlines missed between the two policies. The semi-selfless policy had 59 deadlines missed while the fully selfish policy had 115 deadlines missed.

Fig. 2(d) shows a distribution plot of results over 1000 trials, with 1000 different vehicle sets Γ' tested against both policies. The two policies differed in 84.7% of the trials. The semi-selfless policy performs generally better than the fully selfish policy in almost all scenarios. Across the 1000 trials, it has a much higher distribution in the lower timespans than the fully selfish policy.

E. Results for Π

Across all four ratios of controlled to uncontrolled vehicles, the semi-selfless policy performed better than the fully selfish policy.

In the first scenario, 100 trials were ran simulating a network with 20 controlled and 60 uncontrolled vehicles. This scenario is meant to simulate a situation where only a small number of vehicles in a network is autonomous and able to be controlled by a routing algorithm. The main goal is to test how well the policy holds in a scenario with a large number of random variables. The results show that the policy still holds in this scenario. As seen in Fig. 4(a), the semi-selfless policy performs approximately 19.5% better than the selfish policy across all trials. Fig. 4(b) shows that the semi-selfless policy had less deadlines missed compared to the selfish policy, at 208 against the selfish policy's 395 deadlines missed. It is important to note that this number is out of 2000 vehicles total across all trials. In addition, although not shown, the semi-selfless policy tended to have shorter travel times in the earlier indices of vehicles, but fails to beat the selfish policy in the later indices of vehicles.

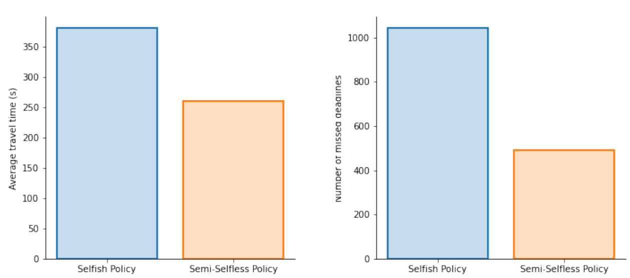


(a) Average travel time of all vehicles in Π

(b) Total deadlines missed in Π

Fig. 4: Results from 100 trials simulation a network with 20 controlled and 60 uncontrolled vehicles

In the second scenario, 100 trials were ran simulating a network with 40 controlled and 40 uncontrolled vehicles. This scenario is meant to simulate an equilibrium. The main goal is test how well the policy holds in a situation where there is a balance between controlled and random variables. The results show that the policy still holds in this scenario. The semi-selfless policy has significantly shorter travel times across the entire vehicle set. Shown in Fig. 5(a), the semi-selfless policy performs approximately 46.7% better than the selfish policy across all trials. Fig. 5(b) shows that the semi-selfless policy had significantly less deadline missed compared to the selfish policy, at 490 against the selfish policy's 1093 deadlines missed. It is important to note that this number is out of 4000 vehicles total across all trials.



(a) Average travel time of all vehicles in Π

(b) Total deadlines missed in Π

Fig. 5: Results from 100 trials simulation a network with 40 controlled and 40 uncontrolled vehicles

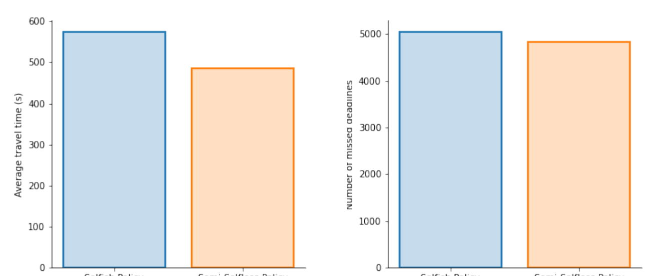
In the third scenario, 100 trials were ran simulating a network with 60 controlled and 20 uncontrolled vehicles. This scenario is meant to simulate a situation where the controller has near total control of the network, and is able to avoid the influence of random variables. The main goal of this is to test whether the policy still holds under a situation where there are few random variables to influence its decisions. The results show that the policy still holds. The semi-selfless policy again had significantly shorter travel times across the entire vehicle set. Shown in Fig. 6(a), the semi-selfless policy performs approximately 60.2% better than the selfish policy across all trials. Fig. 6(b) shows that the semi-selfless policy had significantly less deadlines missed compared to the selfish policy, at 544 against the selfish policy's 1202 deadlines missed. It is important to note that this number is out of 6000 vehicles total across all trials.

(a) Average travel time of all vehicles in Π

(b) Total deadlines missed in Π

Fig. 6: Results from 100 trials simulation a network with 60 controlled and 20 uncontrolled vehicles

In the fourth and final scenario, 100 trials were ran simulating a network with 80 controlled and 80 uncontrolled vehicles. This scenario is again meant to simulate an equilibrium, although one where the network is nearly at capacity and almost overcrowded. The main goal of this is to test if the policy still holds in a scenario where nearly all routes will have high travel times. Shown in Fig. 7(a), the semi-selfless policy performs approximately 18.1% better than the selfish policy across all trials. Fig. 7(b) shows that the semi-selfless policy only had slightly less deadlines missed compared to the selfish policy, at 4829 against the selfish policy's 5040 deadlines missed. It is important to note that this number is out of 8000 vehicles total across all trials.



(a) Average travel time of all vehicles in Π

(b) Total deadlines missed in Π

Fig. 7: Results from 100 trials simulation a network with 80 controlled and 80 uncontrolled vehicles

From these results, it can be clearly concluded that the semi-selfless policy will generally outperform a traditional selfish policy, achieving both goals of minimizing the total average travel time and the number of deadlines missed. However, there is a certain threshold where the semi-selfless policy performs best. This threshold is defined as a situation where a majority of vehicles in a network are controlled by the policy, as in the case with the 60-20 split between controlled and uncontrolled vehicles. In cases where a majority of vehicles are randomly generated, or where the network is overcrowded, the semi-selfless policy only performs slighlty better, with not enough difference to consider the improvement significant.

V. Conclusion and Future Works

In this paper, we proposed two different approaches towards creating a dynamic RGA and explained how the use of a dynamic RGA and the selfless policy that defines it can significantly reduce travel time and deadlines missed compared to the selfish policy. We explained the proposed the semi-selfish policy and evaluated it against the selfish policy using a SUMO-based test bed. To further improve efficiency and get closer towards a truly selfless policy, we then proposed a semi-selfless policy and again evaluated it using a SUMO-based test bed. In both cases, we validated that the policies showed some improvements over a traditional selfish policy. In addition, we found that under certain conditions, both policies showed significant and noticeable performance improvements.

In future works, further testing should be done on different networks of varying complexity. Each policy should be tested against different methods of origin and destination generation to ensure that it is applicable to all possible real-world scenarios. In addition, there are additional criteria that should be taken into account in the evaluation of the policies. First, the time complexity, or the execution time, of the algorithms should be quantified to properly compare computational performance improvements. Second, the policies should be tested with scenarios where there is not a set destination, such as when a vehicle must stop by certain positions within certain deadlines, and where a deadline has the possibility of changing during a vehicle's traveling period. In addition, neither policies proposed truly meet the definition of a selfless policy, where possible network conditions are considered prior to the decision making step, thereby reducing congestion before it can even start. There are numerous ways that we can create a selfless algorithm, and we plan to further explore this avenue to create something that can truly be considered selfless, while also ensuring that it is applicable in real world scenarios. Furthermore, while there is clear merit in the development of heuristic algorithms for the problem of selfless traffic routing, we also plan to deviate and explore the use of machine learning for this task. We would also like to further explore dynamic route finding via pre-computations or the use of pre-trained models, since some devices in the real world may not have enough computational power to run an entire heuristic route finding algorithm in real time.

References

[1] G. Dai, P. K. Paluri, T. Carmichael, A. M. K. Cheng and R. Miikkulainen, "Work-in-Progress: Leveraging the Selfless Driving Model to Reduce Vehicular Network Congestion", 2019 IEEE Real-Time Systems Symposium (RTSS), 2019, pp. 548-551, doi: 10.1109/RTSS46320.2019.00061.

[2] Daxin Tian, Yong Yuan, Honggang Qi, Yingrong Lu, Yunpeng Wang, Haiying Xia, Anping He, "A Dynamic Travel Time Estimation Model Based on Connected Vehicles", Mathematical Problems in Engineering, vol. 2015, Article ID 903962, 11 pages, 2015. https://doi.org/10.1155/2015/903962

[3] Jingwei Shen, Yifang Ban, "Route Choice of the Shortest Travel Time Based on Floating Car Data", Journal of Sensors, vol. 2016, Article ID 7041653, 11 pages, 2016. https://doi.org/10.1155/2016/7041653

[4] "Floyd Warshall Algorithm: DP-16", GeekForGeeks, 2021. Retrieved June 9, 2021, https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

[5] "Finding shortest path between any two nodes using Floyd Warshall Algorithm", GeekForGeeks, 2021. Retrieved June 11, 2021, https://www.geeksforgeeks.org/finding-shortest-path-between-any-two-nodes-using-floyd-warshall-algorithm/

[6] "Floyd-Warshall algorithm", E-Maxx Algorithms in English, 2021. Retrieved June 11, 2021, https://cp-algorithms.com/graph/all-pair-shortest-path-floyd-warshall.html

[7] "Floyd-Warshall Algorithm." George Mason Motion and Shape Computing Group, 2021. Retrieved June 13, 2021. http://masc.cs.gmu.edu/wiki/FloydWarshall