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 three 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 individually optimal routes taking into account estimated network travel time data. The third is a heuristic-based semi-selfless policy that selects globally optimal routes by taking into account both the estimated network conditions as well as the travel deadlines of each individual vehicle.

All 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 selfless 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.

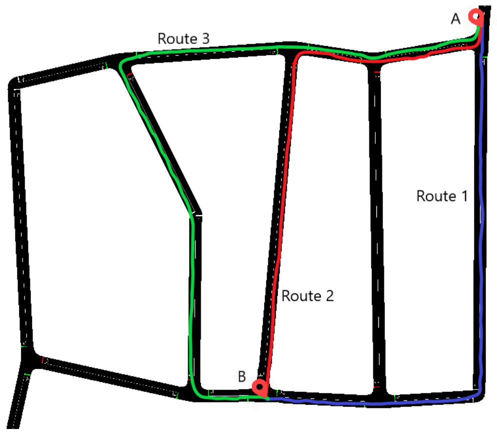
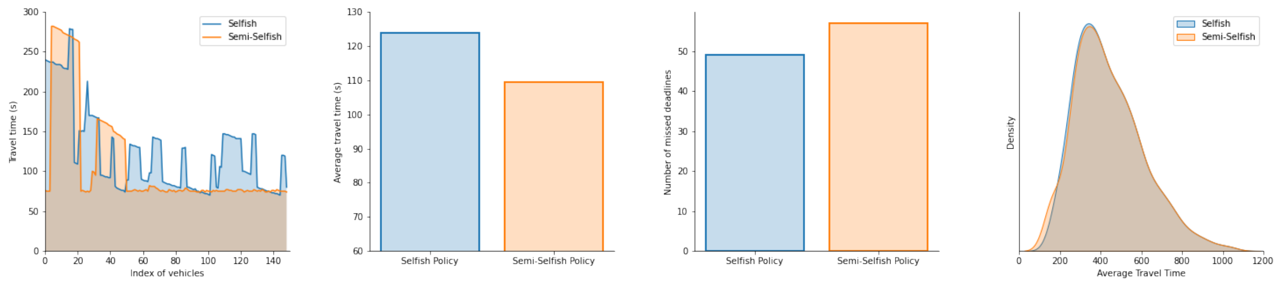


Fig. 1: Simulated road network generated from SUMO

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.

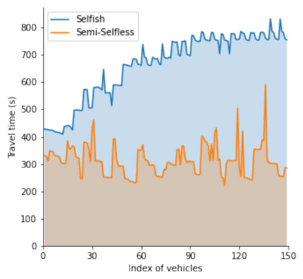
At every decision step, the weights are recalculated by the equations defined in the previous section. Then, using those weights as the costs of each edge, the modified Floyd-Warshall algorithm is used to calculate the best paths from every edge i to every edge j. Then, for all vehicles in the set, the 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.

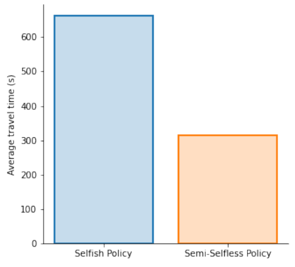
We also consider this policy to be a mixture between selfish and selfless. However, we believe it to be more selfless than the previously presented policy. This policy makes the network conditions central to its routing, instead of as a weighted factor like with the semi-selfish policy. Although it will 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

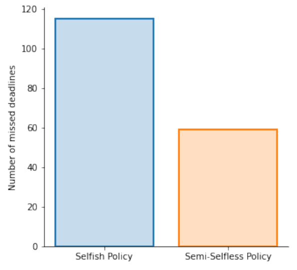
To evaluate the efficiency of the semi-selfless policy, we will compare its results to the fully selfish policy using an identical set of vehicles. The algorithm used in the fully selfish policy is a basic Dijkstra's algorithm, using the travel distance as its weights. The algorithm used in the semi-selfless 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



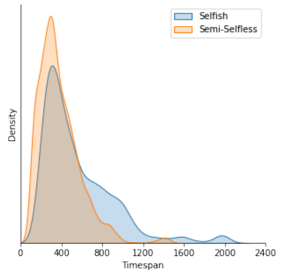
(a) The travel time of each vehicle in Γ'



(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.

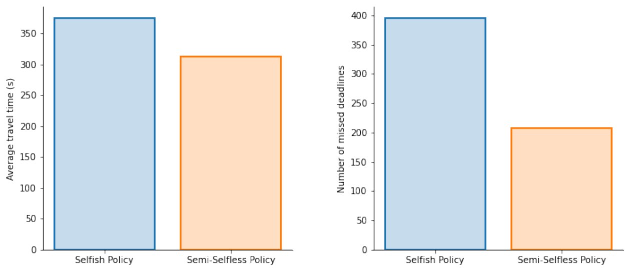
Efficiency improvements are further seen in Fig. 3(c), which compares the total deadlines missed between the two policies. The semi-selfless policy had 59 deadlines missed while the fully selfish policy had 115 deadlines missed.

Fig. 3(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.

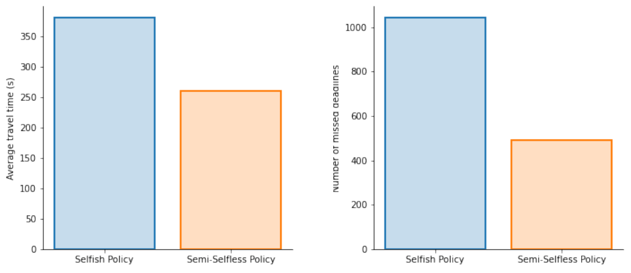


(a) Average travel time of all vehicles in Π

(b) Total deadlines missed in Π

Fig. 4: Results from 100 trials simulating 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.

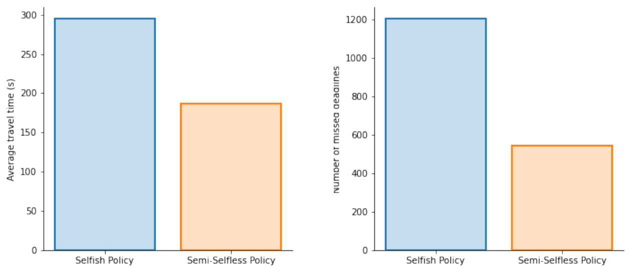


(a) Average travel time of all vehicles in Π

(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.

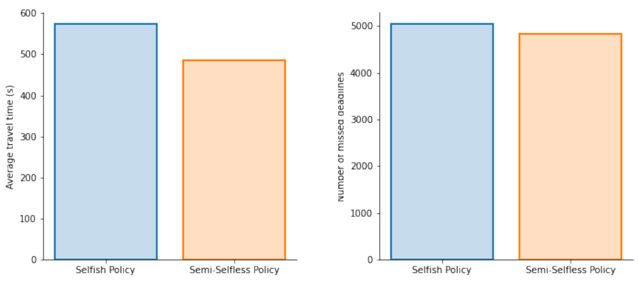


(a) Average travel time of all vehicles in Π

(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 slightly better, with not enough difference to consider the improvement significant.

V. The Heuristic Policy

A. A modification on the travel-time-based weights

Similar to both of the previous policies, the proposed algorithm modifies the weights of each edge of the network in order to minimize the average travel time and the number of deadlines missed. It does this by setting the weight of each edge to the estimated travel time across that edge. However, we no longer use the mean speed of vehicles across the edge for our estimation, as we did in the previous policy, due to several reasons. First, on edges where there is a large congestion, where speeds are less than 0.1 m/s, the assigned weight across that edge can get extremely large (in testing, an average edge had a weight of around 50, whereas edges with standstill traffic had weights of upwards of 10,000). While this may sound good, it would encourage the algorithm to take absurdly long paths, which would usually take more time than if it were to just take the congested path and wait through it. Second, when using the mean speed of vehicles across the edge, two similarly sized edges, one with a low amount of congested vehicles (eg. 3 vehicles) and one with a large amount of congested vehicles (eg. 20 vehicles), would be weighted equally. This is obviously counterintuitive, since the congested path with the low amount of vehicles would take much less time than the one with the large amount of vehicles.

To remedy both of these problems, we have modified our travel time estimation to use the estimated occupancy of the edge. The estimated travel time across an edge i is calculated using the number of cars n^i_{curr} currently on edge i, the estimated maximum number of cars n^i_{max} on edge i, the maximum allowed speed v^i_{max} across edge i, and the length l^i of edge i.

We first find the estimated velocity across edge i

$$v^i_{est} = v^i_{max} * log(\frac{n^i_{max}}{n^i_{curr}}) \tag{6}$$

where v^i_{est} is the estimated speed across the ith edge.

However, in the case that there are no vehicles currently on the ith edge, the previous equation will not work. To fix this, we set n^i_{curr} to an arbitrary low value (in our case, we use 0.01). Based on the equation above, setting n^i_{curr} to a low value would increase the estimated velocity across edge i, thereby encouraging vehicles to take that empty edge.

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

$$t_i = \frac{l^i}{v^i_{est}} \tag{7}$$

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 heuristic policy combines the routing of the previously used Floyd-Warshall algorithm with a slightly modified A* algorithm. The A* algorithm is similar to Dijkstra's, except that it also takes into account a heuristic. In the case of a maze, this heuristic would usually be the Eulerian distance between the current position and the goal position. Through this heuristic, the A* policy will be more inclined to take paths that minimize its heuristic. In the case of mazes, instead of propagating outwards in all directions, it will prioritize propagating towards the destination.

The steps of our algorithm are as follows. At each decision-making step, we first compute the weights of each edge using the criteria explained in the previous subsection. We then use an identical Floyd-Warshall algorithm to the previous policy to calculate the shortest travel time between every edge on the network. Then, for each vehicle in the set, we use the A* algorithm to calculate the best path between the vehicle's current edge and its destination. This algorithm then returns the path the vehicle should take, which we then use to send instructions to each vehicle.

For our situation, we have modified the heuristic used in the A* algorithm to better improve the routing performance. To calculate this heuristic, we used the remaining estimated travel time t^v_{rem} between a vehicle's current position and its destination, the current time of the simulation t_{curr}, and the deadline of the vehicle t^v_{dl}. The difference between this remaining travel time and the deadline is then calculated as

$$t^v_{est} = (t^v_{rem} + t_{curr}) - t^v_{dl} \tag{8}$$

Since this number can get very large, which would overpower the existing weight on the edge, we use the z-score with a $\mu = 0$ and $\sigma = 50$. This z-score is calculated as

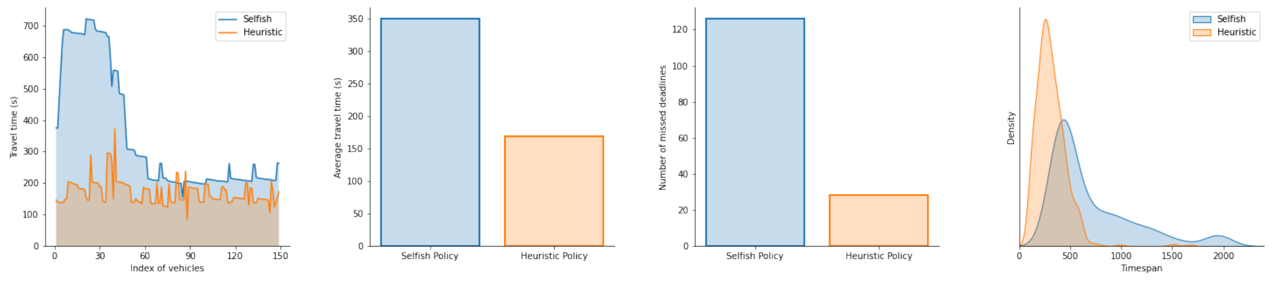
$$z = \frac{t^v_{est} - \mu}{\sigma} \tag{9}$$

We then return this z-score as the heuristic for the A* routing algorithm.

By calculating it in this way, we allow for negative heuristics (ie. if the estimated travel time of this path passes the allowed deadline, the heuristic is positive and the cost for the edge is increased. Else, the heuristic is negative and the cost decreases, thereby encouraging the algorithm to pick this edge).

C. Evaluation Procedure

To evaluate the efficiency of the heuristic 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 heuristic policy is the one detailed in the section before. Unless otherwise indicated, the road network used is the one in Fig. 1. For Fig. 8, the 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



(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. 8: Visual results of Heuristic Policy evaluation

controlled by the policy. The number of vehicles controlled in the policy will range from 10 vehicles to 150 vehicles, in increments of 10. Each set will have a constant 50 random SUMO-generated vehicles. Testing set Π will also test three different methods, hereon referred to as patterns, of starting and destination generation. Pattern 1 means that all vehicles have the same starting and destination point. Pattern 2 means that all vehicles will have the same destination point, but will have different starting points. Pattern 3 means that all vehicles will have different starting and destination points. All graphs in Fig. 8 are conducted under Pattern 1. For each situation in Π, we will test it against 10 distinct vehicle sets to eliminate possible sources of error. In addition, testing set Π will contain results comparing the performances of the fully selfish policy, the previously described semi-selfless policy, and the heuristic policy.

D. *Results for Γ'*

In Fig. 8(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, a very large travel time can be seen in earlier vehicles. This is most likely because, with similar starting and destination points, the fully-selfish policy will always route vehicles across the exact same paths. This means that because it does not take into account network conditions, the selfish policy will continue to send vehicles along congested paths, leading to further congestion for all vehicles on that path until a certain point. This large initial spike is also likely due to the presence of uncontrolled vehicles on paths favored by the selfish policy.

In contrast, the heuristic policy makes sure to route vehicles across multiple different paths to reduce congestion. While some spikes in travel time can still be seen, the heuristic policy is mostly consistent and flat, indicating that most vehicles are reaching their destinations in average times. When taking both travel time and deadline into consideration, vehicles will avoid congested paths that they know will take a long time to traverse across, and will pick paths that can get them to their destination before their deadline. This means that once a road has already been picked by a certain number of vehicles, following vehicles will always pick alternate, faster routes.

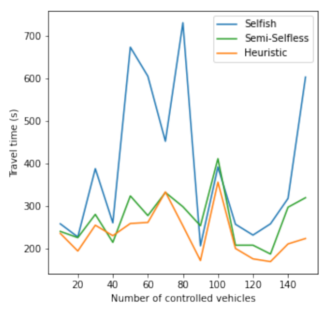
As shown in Fig. 8(b), the average travel time of all vehicles in Γ' under the heuristic policy is nearly half that of the fully selfish policy. It is clear from this that by taking network information and individual vehicle deadlines into account to avoid areas of congestion, the average travel time of all vehicles can be significantly reduced.

Efficiency improvements are further seen in Fig. 8(c), which compares the total deadlines missed between the two policies. Since a central part of the routing procedure for the heuristic policy is the deadline, it is expected that the number of deadlines missed would be significantly reduced. The heuristic policy attempts to ensure that vehicles will reach their destination before their deadline expires, evident in the fact that the number of deadlines missed in the heuristic policy is nearly 5 times lower than that of the selfish policy.

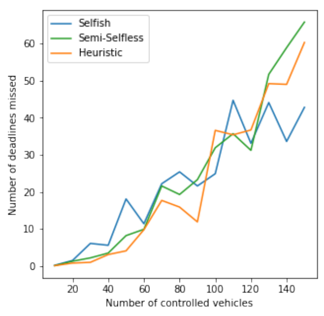
Fig. 8(d) shows a distribution plot of average travel times over 1000 trials, with 1000 different vehicle sets Γ' tested against both policies. Compared to the previously mentioned semi-selfish and semi-selfless policies, the distribution of the heuristic policy shows clear and distinct differences to the selfish policy. First, the mean of the distribution plots is much lower in the heuristic policy than the selfish policy, at around 250 seconds compared to 500 seconds. This indicates that the heuristic policy is getting vehicles to their destination positions in a much faster time, which is consistent with our other observations. More importantly, the variance of the distribution plot in the heuristic policy is much less than the selfish policy. This shows that vehicles are generally getting to their destinations with similar travel times, indicating that the heuristic policy does not have the same problem as the selfish policy where there is large congestion and some vehicles reach their destinations much later than others. We believe this large difference in distribution to be one of the most significant factors attesting to the heuristic policy's improvement over the selfish policy, since it shows that the heuristic policy is able to get very close to the selfless global optimum, where it mostly avoids congestion and minimizes the average travel times of all vehicles instead of a select few.

E. *Results for Π*

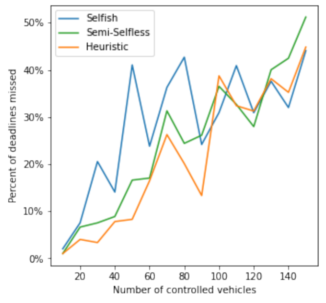
Across most scenarios in Π, the heuristic policy performed better than both the semi-selfless and the selfish policies.



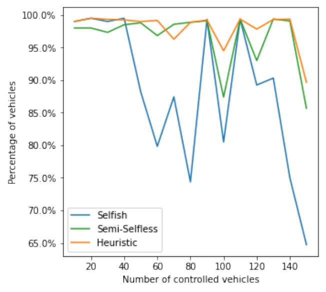
(a) The average travel time of vehicles in Π



(b) The average number of deadlines missed in Π



(c) The percentage of deadlines in Π that were missed

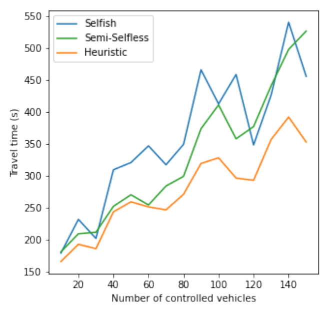


(d) The percentage of vehicles that reached their destinations in Π

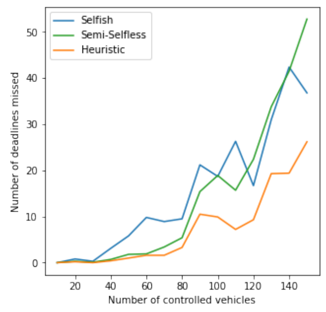
Fig. 9: Policy comparisons on vehicles sets in Π under Pattern 1

Under Pattern 1, where the starting and destination points of all vehicles in the set are identical, the heuristic policy showed better average travel times across all scenarios compared to the selfish and semi-selfless policies, as shown in Fig. 9(a). This trend also continues onto the average number of deadlines missed. Although from Fig. 9(b), it may seem that the heuristic policy performs worse than the selfish policy as the number of controlled vehicles increases, this is not actually the case. Due to the nature of the simulation, vehicles that do not reach their destination are not recorded. By looking at Fig. 9(d), we see that as the number of vehicles increases, the selfish policy sees a substantial drop in the percent of vehicles that reach their destinations. This indicates that although the average number of deadlines missed is lower for the selfish policy, there are also much less cars that do reach their destination. This issue is rectified through Fig. 9(c), which shows the percentage of deadlines missed compared to the total number of cars that reached their destination. Through this, we see that in cases where the network is overloaded (ie. there are too many cars on the network), the heuristic policy performs nearly equal to the selfish policy. In other cases, it performs substantially better.

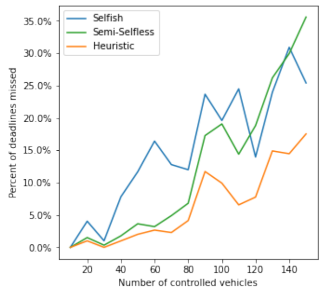
Under Pattern 2, where the destinations are the same but the starting positions are different for all vehicles, we see a more consistent trend. Seen in Fig. 10 (a), as the number of controlled vehicles increases, the heuristic policy continues to performs significantly better than the selfish and semi-selfless policy. Its improvements over the other two policies becomes more pronounced the more controlled vehicles there are, indicating that when vehicles are beginning from different positions, it is able to more efficiently route them along different paths to avoid congestion. Shown in Fig. 10(b), Fig. 10(c) and Fig. 10(d), the heuristic policy also has significantly less deadlines missed, and consistently ensures that vehicles reach their destination. This is in contrast to the selfish policy, where there again is a substantial drop in the percentage of vehicles that reach their destinations as the number of controlled vehicles increases. In addition, the heuristic policy consistently has a higher percentage of vehicles that reach their destination than both of the other policies, while also having fewer average deadlines missed.



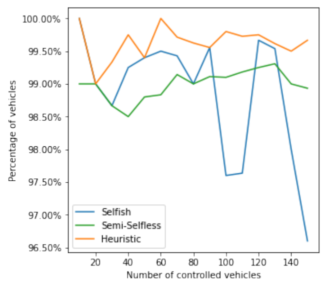
(a) The average travel time of vehicles in Π



(b) The average number of deadlines missed in Π



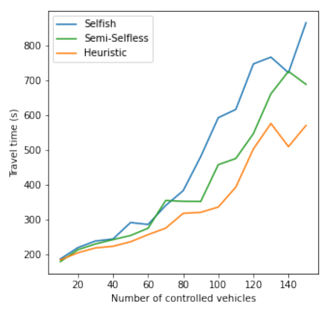
(c) The percentage of deadlines in Π that were missed



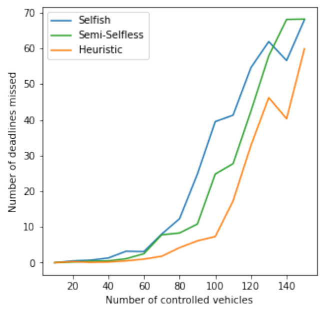
(d) The percentage of vehicles that reached their destinations in Π

Fig. 10: Policy comparisons on vehicles sets in Π under Pattern 2

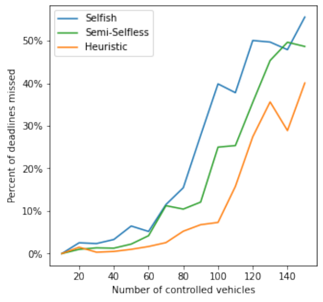
Finally, under Pattern 3, where the starting and destination positions of all vehicles are different, we see a similar but more pronounced trend to what we saw in Pattern 2. As seen in Fig. 11(a), there is a large difference in the average travel time, with the heuristic policy consistently performing at least two times better than the selfish policy. As the number of controlled vehicles increases, this difference becomes more pronounced. Interestingly though, the difference stays somewhat consistent between the heuristic and the semi-selfless policy. Looking at Fig. 11(b) and Fig. 11(c), the heuristic policy has substantially less total deadlines missed compared to both of the other policies. This difference is even more pronounced when we look at the percentage of deadlines missed, indicating that, as before, as there are more vehicles on the network and the network gets overloaded, the selfish and semi-selfless policy struggles to ensure that all vehicles reach their destination. This is seen especially in Fig. 11(d), where there are large drops shown in the semi-selfless and the selfish policy, but none are seen in the heuristic policy.



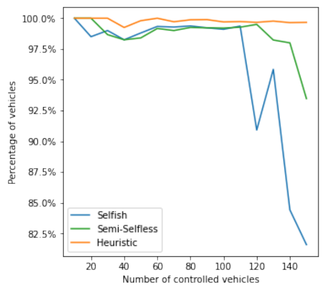
(a) The average travel time of vehicles in Π



(b) The average number of deadlines missed in Π



(c) The percentage of deadlines in Π that were missed



(d) The percentage of vehicles that reached their destinations in Π

Fig. 11: Policy comparisons on vehicles sets in Π under Pattern 3

From these results, it can be clearly concluded that the heuristic policy will generally outperform both the traditional selfish policy as well as the semi-selfless policy introduced in this paper. It achieves both goals of minimizing the total average travel time and the number of deadlines missed. Unlike with the semi-selfless policy, there is not a threshold where the heuristic policy will begin to perform worse. Seen in the results presented, regardless of the number of controlled vehicles on the network or the way in which the starting and destination points are generated, the heuristic policy performs better than the selfish policy. The only situation where it may perform worse is when the starting and destination points of vehicles are identical, and there is a large number of controlled vehicles. In this case, the heuristic policy performs marginally worse. However, we believe this very small difference to be inconsequential, and may be due to random error.

VI. Conclusion and Future Works

In this paper, we proposed three 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 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. Finally, we combined the semi-selfless policy with a new heuristic to create the heuristic policy. In all cases, we validated that the policies showed some improvements over a traditional selfish policy. In most scenarios, the semi-selfish policy performed similarly, if not marginally better, than the selfish policy; the semi-selfless policy performed generally better in some scenarios, but performed equally if not worse in others; finally, the heuristic policy performed significantly better in almost all scenarios.

In future works, it may be helpful to improve the interpretability of the algorithms. Though there are many ways this can be done, perhaps the most intuitive way is through a graphical user interface (GUI) that can display which paths a certain vehicle will choose and why it will choose that path instead of other competing paths. Further testing should be also done on different networks of varying complexity. In addition, the time complexity, or the execution time, of the algorithms should be quantified to properly compare computational performance improvements. In addition, although the heuristic policy is close, none of the 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, such as through the use of artificial intelligence via reinforcement learning or through some other form of heuristic algorithm. We also believe further exploration should be done on 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

[8] J. Zhang, S. Pourazarm, C. G. Cassandras and I. C. Paschalidis, ”The Price of Anarchy in Transportation Networks: Data-Driven Evaluation and Reduction Strategies,” in Proceedings of the IEEE, vol. 106, no. 4, pp. 538-553, April 2018, doi: 10.1109/JPROC.2018.2790405.

[9] Anton Agafonov, Vladislav Myasnikov, ”Efficiency comparison of the routing algorithms used in centralized traffic management systems”, Procedia Engineering, 2017, pp. 265-270, doi: 10.1016/j.proeng.2017.09.617.