Report: Selfless Traffic Routing

John Tian (johntian2014@gmail.com)
Clements High School, Sugar Land, Texas
University of Houston, Department of Computer Science, Houston, Texas

I. Introduction

Modern-day cities are built around cars. Due to immense populations depending on this method of travel, urban congestion has become a crucial issue. Simply finding the shortest route to a destination may no longer be the best choice for every driver on the road. In this report, I present an improved "selfish" routing algorithm that I experimented with before building up a "selfless" model, both of which have been evaluated on the SUMO Test Bed.

II. Sorting by Time

A. Explanation

My first approach encourages vehicles to choose the route that takes the least time. When a vehicle reach-es a decision point, we run Dijkstra's on its outgoing edges to find multiple paths (if possible) to the destination.

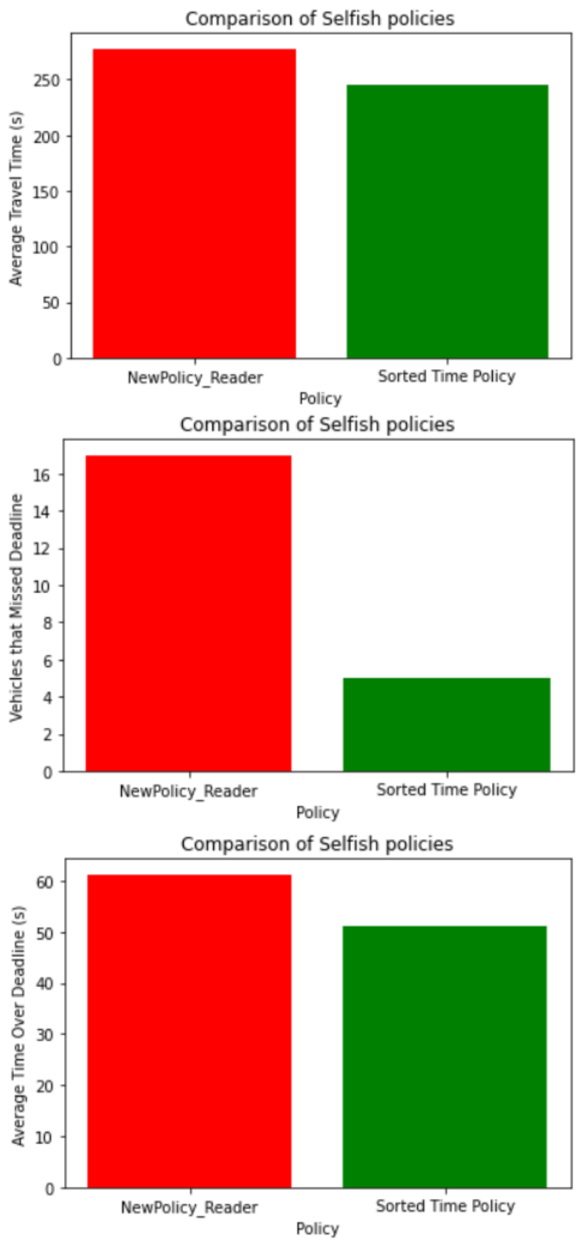
We can roughly estimate the time each route takes by considering each edge, e concerning its length, e_l, and its allowed speed, e_s. The estimated time that a vehicle can spend on an edge is thus e_l/e_s. Then we can sum up the times for each edge to get our estimation for the time it takes to follow each path.

After making these calculations, we can sort each list of edges and select the one that takes the least time as our route. In comparison, the base algorithm only has one route to choose from.

B. Results

This approach is tested in the base network with 354 vehicles against NewPolicy_Reader. Figure 1 reflects an approximate 11.7% decrease in average travel time as well as improvement in other metrics.

Figure 1



III. The Selfless Policy

A. Motivation

Returning to the problem in terms of distances, it is logical that we want to utilize the concept of *all-pairs shortest paths* – i.e., the ability to compute the shortest path between each pair of edges.

To do this, we can use Floyd-Warshall's algorithm to create an edge matrix on each decision step that stores the shortest distance between any two edges [1]. We can then trace the optimal path's steps for each vehicle [2].

Johnson's algorithm is another all-pairs shortest path technique, but I elected to use Floyd-Warshall due to its effectiveness with denser graphs.

B. Explanation

To make the algorithm selfless, I considered using feature engineering to generate different metrics that give us more detail on the traffic situation for the entire map [3]. Each of these metrics is recomputed on each decision step.

- Density: The number of vehicles on an edge divided by the length of the edge
- Flow: The density multiplied by the speed – i.e., the number of vehicles that exit the edge per second
- Time: The length of the edge divided by the mean speed of vehicles passing through it – i.e., how long it takes for a vehicle to travel through an edge
- Distance: The length of an edge
- Length Taken Up: The length of an edge that the vehicles occupy
- Halting: The percent of vehicles on an edge that halt (reduce speed to <0.1 m/s)
- Wait Time: The sum of each vehicle's waiting time on an edge
- On Edge: The percentage of all the vehicles that are on each edge.

After experimenting with these metrics to find an optimal way to weigh the Floyd-Warshall edge matrix, I came up with the following formula:

$$w_{ij} = max\big((1 - h_i) * dists_i + h_i * density_i, 0\big)$$

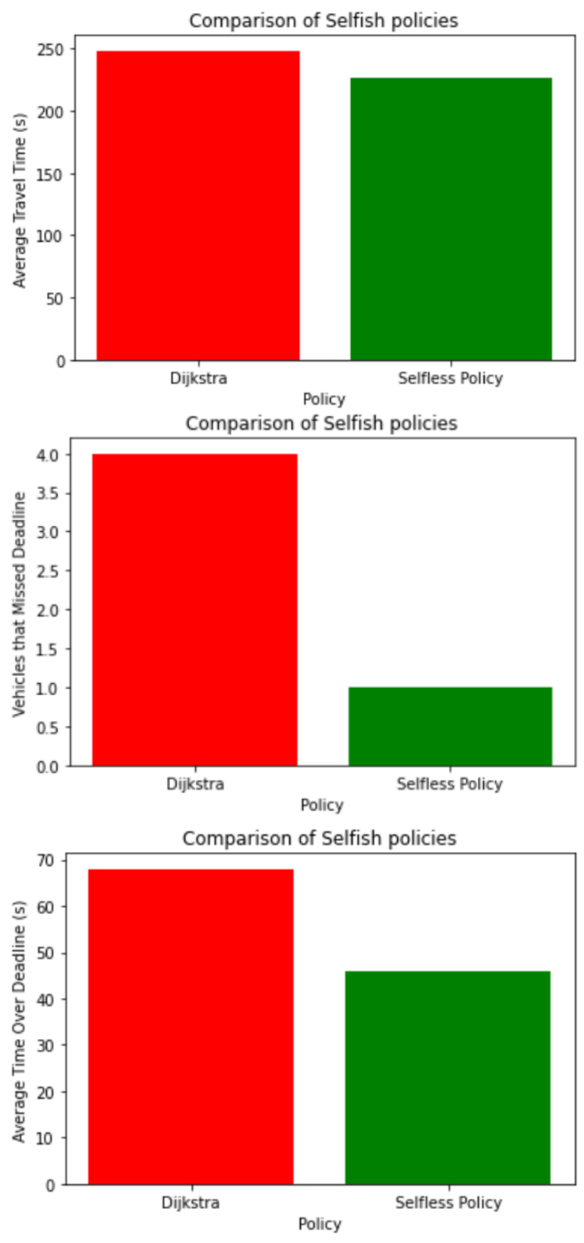
w_{ij} represents the weight from edge i to edge j and h_i represents the halting of edge i.

On each decision step, we compute the weights of the edge matrix using this formula and apply the same logic from the shortest paths Floyd-Warshall to reconstruct the optimal path. This time, however, the optimal path minimizes the weight rather than the shortest path between the vehicle's current edge and destination.

C. Results

This approach is tested in the base network with 354 vehicles against Dijkstra's Shortest Path Algorithm. The selfless policy outperforms the selfish policy as shown in Figure 2, with an approximate 8.8% reduction in travel time.

Figure 2



IV. Conclusion

By considering the global state of the map on each decision step by using feature-engineered weights, my selfless routing algorithm showed noticeable improvement over Dijkstra's on the test map.

There may be more optimal ways to define weights that further improve performance using the other metrics that I defined or ones that others can create in the future.

V. References

[1] "Floyd–Warshall Algorithm." *Wikipedia*, Wikimedia Foundation, 31 July 2022, https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm.

[2] "Floyd-Warshall Shortest Path Algorithm." *AlgoTree*, https://algotree.org/algorithms/floyd-warshall/.

[3] "Traffic Density Calculator." *CalCon Calculator*, 17 Feb. 2022, https://calconcalculator.com/everyday-life/traffic-density-calculator/.