Dynamic Route Guidance in Vehicle Networks by Simulating Future Traffic Patterns

David Neiman, Zachary Rubinstein, Stephen Smith
The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{dneiman, zbr, ssmith}@andrew.cmu.edu

Abstract

Roadway congestion leads to wasted time and money and environmental damage. Since adding more roadway capacity is often not possible in urban environments, it is becoming more important to use existing road networks more efficiently. Toward this goal, recent research in real-time, schedule-driven intersection control has shown an ability to significantly reduce the delays incurred at signalized intersections. Such approaches operate by building a predictive model of when locally sensed approaching traffic is expected to arrive at a given intersection, and then using the model to generate a signal timing plan (a phase schedule) that minimizes the cumulative delay of this traffic as it moves through the intersection. In this paper, we consider whether further reduction in delay could be gained by applying this predictive model to dynamically reroute some portion of vehicles willing to share their destinations along less congested paths. We developed an algorithm that simulates the current traffic state forward at each vehicle decision point, based on knowledge of other vehicles' current routes and traffic signals' control algorithms, and evaluated it using the SUMO microscopic simulator on different road networks (one as a simple synthetic example and the other taken from the real world) using different traffic signal control algorithms (fixed-timing plans and schedule-driven intersection control). Experiments carried out on combinations of networks and traffic signal control algorithms show that our rerouting protocol reduces delay for both vehicles participating in route guidance (adopters) and those that do not (non-adopters) and that the reduction in delay generally increases as the proportion of adopters does.

Introduction

Roadway congestion leads to wasted time and money, environmental damage due to release of CO_2 and other greenhouse gases, and increased need for infrastructure repairs. One way to alleviate congestion is by adding new roadway capacity (for instance, adding more lanes to existing roads), but this is often infeasible in urban environments due to lack of space. In this paper, we focus on strategies for reducing travel times of vehicles sent through a given road network, by having vehicles use alternative routes based on predicting expected vehicle flow and traffic-signalization behavior along potential routes.

One way to reduce travel times without changing the road network is to improve the schedules of the traffic signals. A broad review of traffic signal strategies can be found in (Papageorgiou et al. 2003). In this paper, we either use fixed timing plans, or evaluate with a variant of the adaptive traffic signal scheduling algorithm Surtrac (Xie et al. 2012) (Xie, Smith, and Barlow 2012) (Smith et al. 2013). Adaptive traffic signal scheduling algorithms assume some form of detection, like cameras or induction loops, to detect traffic near a traffic signal, and then adjust the schedules based on the observed traffic. For example, Surtrac optimizes a short-term future schedule for each traffic signal to minimize the delay that incoming vehicles will experience, and can compute a new schedule roughly once a second on typical real-world road networks. Decentralized real-time intersection scheduling alone has been found to reduce delays compared to fixed timing plans, further improvement can be obtained by having intersections communicate predicted outflows to their downstream neighbors (Xie, Smith, and Barlow 2012). A later algorithm, called ERIS (Goldstein and Smith 2018), extends the Surtrac algorithm by keeping track of separate queues for each lane, rather than merging all lanes for a single light phase into a single queue. In this paper, we use this lane-based version of Surtrac to evaluate our algorithm's performance when combined with adaptive traffic light schedules.

A complementary approach to reducing congestion that we explore in this paper is to reroute vehicles onto more efficient routes. A general survey of routing algorithms can be found in (Bast et al. 2016). However, accounting for traffic can make vehicle routing a challenging problem since, especially in urban environments, many routes may have similar travel times in low-traffic situations, but as more traffic tries to use the network, congestion and traffic signal timings might make one route more favorable than another. Current systems like Google Maps predict the most efficient route using a combination of historical data and travel times of vehicles currently traversing particular routes. While useful, such data may not accurately reflect future travel times of vehicles due to secondary congestion formed by the rerouted vehicles (Cabannes et al. 2017).

Most routing algorithms, including ours, assume that individual drivers are trying to minimize their own travel times, rather than minimizing the average travel time

of everyone on the road. The end result is called User Equilibrium (UE), which Wardrop notes is in general different from the system-optimal solution (Wardrop 1952). One example of this is Braess's paradox, where a new road added to a network may increase overall travel times if travel times are dependent on the amount of traffic using each road. The UE solution involves drivers using the new road, and increasing demand on some previously-existing roads, which slows down all drivers using those previously-existing roads. The paradox is that ignoring the newly-added road would be closer to system-optimal, but because each driver tries to minimize their own travel time, the resulting selfish solution is worse than if the new road were not added. While some work has been done in trying to achieve system-optimal performance (see for example (Bagloee et al. 2017)), that work is usually in the context of self-driving vehicles where a human may not be choosing the route.

Other algorithms have been developed that use stochastic models based on Wardrop's User Equilibrium (UE) principle (Daganzo and Sheffi 1977), (Prashker and Bekhor 2004). Qian and Zhang (Qian and Zhang 2013) explore a combination of the Boston User Equilibrium (BUE) and Predictive User Equilibrium (PUE) models introduced in (Friesz et al. 1993). PUE assumes drivers pick their routes based on historical data (implemented as iteratively picking routes until equilibrium is reached) and do not change their routes, while BUE assumes drivers dynamically change their routes based on current traffic conditions. Qian and Zhang find that when a high proportion of drivers use the BUE model, such that they make real-time routing decisions based on the current traffic state, overall travel times increase. The main problems the BUE model causes in this example are that the current traffic state does not accurately represent the future traffic state when congestion is forming or dissipating, and that these route changes themselves affect the future traffic state in ways not accounted for by either the BUE or PUE models.

To avoid these problems, in this paper we propose a routing algorithm that explicitly simulates the current traffic state forward in time to more accurately predict travel time along different possible routes. This routing algorithm can be used on top of Surtrac or other traffic signal scheduling algorithms. We hypothesize that this algorithm can mitigate the previously-mentioned problems with dynamically changing vehicle routes, as well as taking advantage of information about traffic light schedules and other vehicles' intended routes to further reduce travel times.

Methods

In this paper, we explore routing vehicles by simulating the current network state forward to estimate traffic at future times. We assume some proportion of drivers will use our routing algorithm (we call these drivers adopters), while the rest will use fixed pre-computed routes (we call these drivers non-adopters).

We assume the routing algorithm has position data for vehicles currently in the road network (currently queried directly from simulation, but potentially gathered from cell phone location, connected vehicle technology, or existing traffic detectors), and predict where vehicles are going using either their intended routes (for adopters) or historical data about turning ratios (for non-adopters). By explicitly simulating forward in time, the algorithm more accurately predicts the fastest route for each vehicle to reduce travel times for the adopters. Our hypothesis is that this simulate-forward approach also reduces the overall congestion of the network, reducing travel times not only for the adopters but also the non-adopters. This procedure of simulating ahead can work with either fixed timing plans or adaptive traffic signal schedules like Surtrac, and we evaluate with both of these conditions.

It is worth noting that the routing algorithm and Surtrac have slightly different objectives. Surtrac is trying to reduce the total delay incurred by all vehicles waiting to get through each individual intersection, the routing algorithm is only trying to reach a UE solution. In other words, our algorithm helps self-interested drivers choose better routes for themselves, which we predict will lead to a better global solution. Due to these conflicting goals, it is unclear how these two systems will interact. We hypothesize that communication between dynamic vehicle routing and adaptive traffic signals can improve the effectiveness of both, since vehicle routing can get better estimates of travel times using information about traffic signal schedules, similar to how previous work has shown adaptive traffic signals can better predict future traffic and better optimize their timing plans using vehicles' intended routing information (Hawkes 2016). Our experiments show that as we increase the proportion of drivers using our routing algorithm (adoption probability), the delay (computed as total travel time minus what the travel time would be if travelling on the least-time route at the speed limit) almost always decreases, whether we use fixed timing plans or adaptive traffic signal schedules generated by Surtrac.

Our routing system consists of three parts:

- In the main simulation, vehicles follow their current routes through the network, and a routing simulation is invoked whenever a vehicle using our routing algorithm approaches an intersection.
- Routing simulations (see algorithm 1) get invoked to recompute an adopter's route whenever it approaches an intersection in the main simulation.
- When using Surtrac, the Surtrac algorithm gets invoked once per second in both the main and all routing simulations, to predict the traffic signal behavior in response to traffic.

Adopters of our routing algorithm trigger routing simulations whenever they pass detectors located 50m before each intersection (or at the start of the road segment if the segment is shorter than 50m), and change their routes in response. During these routing simulations, to simulate behavior of non-adopters, the algorithm samples a random route for each non-adopter, based on turning ratios at each intersection (where the turn ratios are calculated from the initial routes of all the vehicles), to reflect the assumption that, while we have access to general information about

traffic demand, individual non-adopters do not share their intended routes with the system.

The routing simulations keep track of the locations of clusters of vehicles, and predict their motion in future timesteps. These routing simulations act as predictive models for future system states, leading to more accurate route time estimates and better routes for adopters. In each routing simulation, as a cluster of vehicles reaches an intersection, the routing simulation sends each vehicle in that cluster through the intersection at the first possible time (while accounting for red lights and other vehicles that might cause delays). We assume the routing simulation knows the routes of all adopters, so when an adopter that we are not currently routing goes through an intersection, the simulation simply sends it to the next road on its route. Non-adopters are assumed to follow randomly sampled routes based on the turning ratios, so after going through an intersection they also get sent to the next road on their routes. However, whenever (a copy of) the vehicle being routed approaches an intersection, our algorithm creates new copies of that vehicle and sends them down all possible next lanes simultaneously in order to evaluate alternative routes (see algorithm 1, lines 10-31). The routing simulation ends when the first copy of the vehicle being routed reaches its destination, and the route it took is returned as the route that should be taken in the main simulation (algorithm 1, lines 26-28). Note that while some copies of the vehicle being routed might get stuck behind other copies of the vehicle being routed, the extra delay on the later copies does not matter since only the first copy to arrive at the destination determines the new route.

When running scenarios with traffic signals controlled by Surtrac, the Surtrac function gets invoked for every light in both the main and routing simulations, and loops over all traffic signals and computes planned schedules for them. We use a modified version of the original Surtrac algorithm, where we keep track of separate cluster lists for each lane of each incoming edge. To handle general traffic signal schedules where clusters may be able to proceed in multiple light phases, each time the algorithm schedules a cluster, it generates all partial schedules that result from sending that cluster in all phases in which that cluster is allowed to move through the intersection, and stores all of those resulting partial schedules.

As described in (Xie, Smith, and Barlow 2012), once an intersection computes its planned schedule, the clusters it scheduled could be transformed into predicted clusters for the downstream neighbors, allowing for longer-term scheduling (at the cost of solving a larger scheduling problem). However, in this paper, we disable that to make the routing algorithm run faster.

Algorithm 1 Routing simulation algorithm

```
1:  VOI = vehicle we are finding a route for
2:  VOIs = {VOI}
3:  Randomly sample routes for non-adopters, and read routes for adopters
4:  for lane ∈ all lanes on all edges do
5:      (Determine whether front vehicle has reached the next intersection yet)
6:      veh = front vehicle from front cluster in lane
7:      if veh.arrivaltime > current time then
8:          continue (skip this lane; veh has not arrived yet)
9:      end if
10:     (Determine veh's next lane(s))
11:     if veh ∈ VOIs and veh.nextlanes has not been computed yet then
12:         veh.nextlanes = {all lanes on all possible next edges}
13:     else
14:         veh.nextlanes = {all lanes on next edge in route that connect to next next edge in route (assuming it exists)}
15:     end if
16:     (Copy veh to its next lane(s), if possible)
17:     for nextlane ∈ veh.nextlanes do
18:         if there is not a red light between current lane and nextlane and nextlane has space then
19:             Append a copy of veh to the end of nextlane
20:             veh.arrivaltime = length of nextlane / speed limit on nextlane
21:             if veh ∉ VOIs then
22:                 veh has gone through intersection, so delete veh from lane and stop considering the rest of nextlanes
23:             else
24:                 Keep track of the route the new copy of VOI has taken so far
25:                 Only delete veh from lane if we have added it to each item in veh.nextlanes
26:                 if nextlane is on VOI's goal edge then
27:                     return the route that this copy of VOI took
28:                 end if
29:             end if
30:         end if
31:     end for
32: end for
```

Experiments

We evaluated our algorithm in the traffic simulator SUMO (Lopez et al. 2018), using the TraCI library to allow our Python code to communicate with SUMO simulations. SUMO can generate default fixed timing plans for traffic signals, so we copy the sequence of phases from that default traffic signal schedule. We use a minimum duration of 5

seconds for phases where some motion has a green light, and 7 seconds for phases where all motions have yellow or red lights (SUMO does not automatically schedule all-red phases, so we extend the yellow phase accordingly); all phases have a maximum duration of 120 seconds.

We evaluated our algorithm on a synthetic one-way network designed as a simple example for our algorithm, as well as on a more realistic network based on a section of downtown Pittsburgh. These networks are shown in figure 1.

The synthetic network was designed as a simple example to confirm that our algorithm exhibits desired behavior when used with SUMO's default fixed timing plans. In this network, vehicles come in from the left, right, and (primarily) bottom, and exit through the top. Vehicles coming in the bottom can either go left (very slightly shorter) or right, and where those routes meet the incoming traffic from the sides, there are traffic signals. We designed the fixed timing plans for these signals to be exactly out of sync with each other, so at any given time traffic from the bottom can proceed through either the left or right traffic signal, but not both. At those signals, traffic can choose to either go through the middle or around the outside. The middle is slightly shorter, but requires that the two incoming traffic flows merge into a single lane, which can get congested if all vehicles go through the middle. The middle and the two outside routes meet at the top, where there are three lanes so there is no further need to merge lanes as vehicles exit the network.

The two routing decisions here are whether the bottom vehicles go left or right, and whether vehicles at the traffic signals take the zipper merge or the outside route. With fixed timing plans, the expectation is that our routing will send the vehicles from the bottom toward whichever light will be green when they arrive, and that at the second decision point enough vehicles get sent through the outside routes that no significant congestion forms at the zipper merge. With adaptive traffic light schedules, it is unclear whether the bottom traffic will be guaranteed to hit a green light, but we still expect to avoid congestion at the zipper merge.

On this network, we sent 1000 vehicles in the bottom, and 300 in each side, evenly spaced over a simulated hour, and disregarded the vehicles sent in the first and last 10 minutes when computing delay statistics (the network starts and ends empty, so those vehicles do not give us a good idea of steady-state delay).

The downtown network consists of 11 intersections in downtown Pittsburgh, all with traffic signals. The demand was based on PM rush hour data from 2004, with 10% extra demand added, resulting in 5841 vehicles inserted over a simulated hour.[1] As on the simulated network, we disregarded the vehicles sent in the first and last 10 minutes when computing delay.

In the synthetic network, initial routes were obtained by setting origin-destination pairs and arrival times and running a simulation, letting SUMO's default routing strategy pick

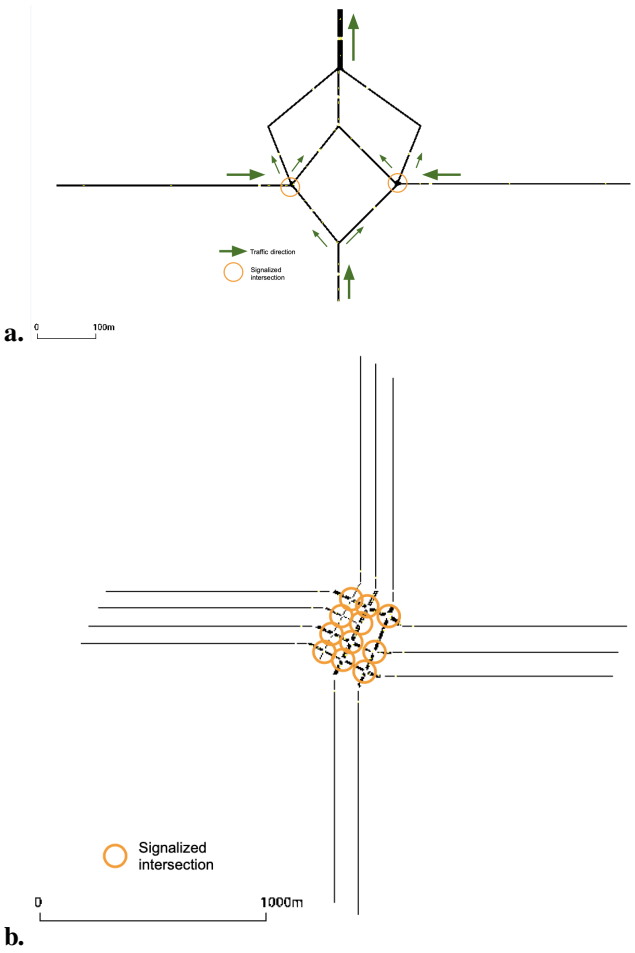


Figure 1: The networks we evaluated our routing algorithm on. **a:** A synthetic one-way network designed as a simple example for our routing algorithm. **b:** A more realistic network based on an area in downtown Pittsburgh.

routes for all vehicles; that default routing strategy does selfish routing using Djikstra's algorithm, where the edge weights are computed using the average speed of vehicles currently on each edge.[2] In the Pittsburgh network, we generated routes based on the demand data, by using the number of vehicles per hour on each road to generate turn ratios at each intersection and origins for all vehicles. We then sampled routes through the network for each vehicle, starting at its origin and using the turn ratios to pick subsequent roads, resulting in routes that approximately match the demand data. In both scenarios, those generated routes were saved to a file, and used as the initial routes during our routing experiments.

Also, for each combination of network and traffic signal scheduling algorithm, we evaluate various adoption probabilities. Each time a vehicle enters the network, we randomly decide whether it uses our routing algorithm based

[1] We added 10% because 1) Surtrac manages the actual PM rush hour volume so well there's no room for improvement and 2) the traffic has likely increased since 2004.

[2] According to SUMO routing documentation here: https://sumo.dlr.de/docs/Simulation/Routing.html

on the adoption probability; non-adopters continue with their original route, which is unknown both to the routing simulations and (if using Surtrac) the Surtrac calls, while adopters are allowed to change their route at points located 50m from intersections, and always share their planned routes with each other (for use in routing simulations) and the traffic signal control (for use in Surtrac calls if applicable).

Results

Synthetic Network

We first evaluated on our synthetic network using both fixed timing plans and Surtrac-generated adaptive timing plans; results can be seen in figure 2. For fixed timing plans, SUMO's default routing sends vehicles down the shortest path until it gets congested, then sends them down the alternative path until the shortest path clears up (or the alternative also gets congested). In contrast, our routing strategy sends the bottom vehicles in whatever direction makes them hit a green light (it can do this because it is aware of the traffic signal timings, and because the traffic signals are exactly out of phase with each other). At the second decision points, it does load-balancing such that no congestion builds up before the zipper merge. As a result, the adopters do better than the non-adopters, and we see about a 57% reduction in overall delay at 99% adoption compared to 1% adoption.

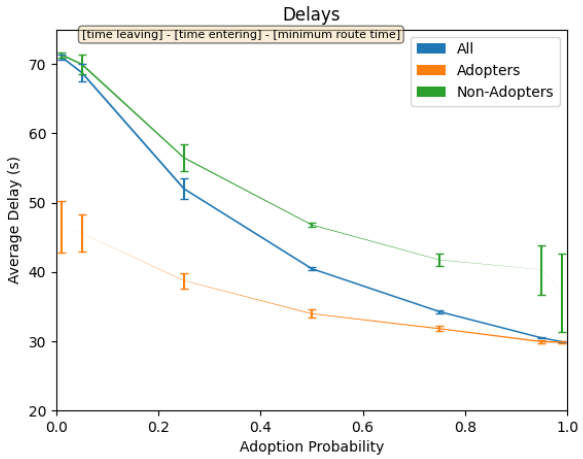
We see similar, though less dramatic, results when we use the Surtrac algorithm to generate timing plans for the traffic signals, as Surtrac on its own leads to significant improvement even without routing. However, we still see about an 25% improvement with 99% adoption and Surtrac compared to 1% adoption and Surtrac.

Pittsburgh Network

Having established that our routing algorithm works on a simple synthetic network, we moved on to evaluating on the more realistic Pittsburgh network (see figure 3). We could not find current or optimized timing plans for this network, so our fixed timing plans were again generated using SUMO's defaults. Unsurprisingly, Surtrac performs better than fixed timing plans on this network at all adoption probabilities (for example, about 7% less delay at 1% adoption). On this network, the combination of Surtrac and routing continues to do better than fixed timing plans and routing even as the adoption probability increases. We see about an 13% improvement with fixed timing plans and 99% adoption compared to fixed timing plans and 1% adoption. Similarly, we see about a 22% improvement in overall delay with Surtrac and 99% adoption compared to Surtrac and 1% adoption.

Especially at low adoption probabilities, the error bars are much larger compared to the synthetic network, which we believe is related to the network's ability to get deadlocked. (After five minutes of deadlock, SUMO starts teleporting vehicles to break the deadlock, but due to the significant delay caused to all deadlocked vehicles, the number and size of such deadlocks can greatly impact the average delay.)

a. Fixed timing plans (synthetic)



b. Surtrac (synthetic)

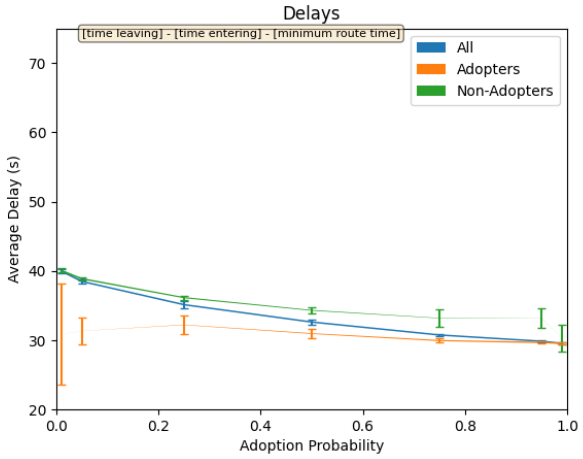


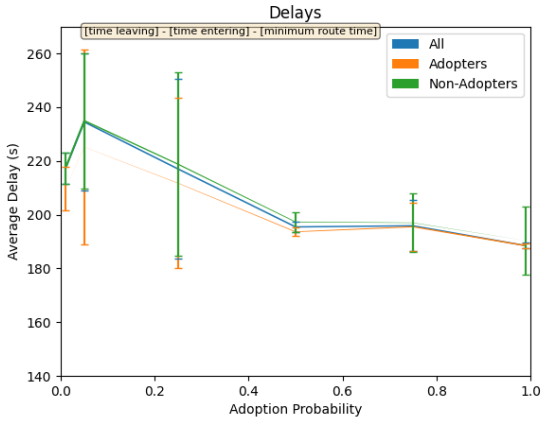
Figure 2: Average delays for all vehicles, adopters, and non-adopters in the synthetic network, for various adoption probabilities. Line thickness is proportional to the number of vehicles per run that contribute to that data point (for example, at 5% adoption, the "Adopters" line is 5% as thick as the "All" line). Average values were computed as the mean of the relevant delay values from 5 runs, and error bars show the standard deviation of the mean delay between those runs. **a:** Fixed timing plans using SUMO defaults. **b:** Adaptive timing plans generated from Surtrac.

Unsurprisingly, both with Surtrac and with fixed timing plans, increasing adoption probability decreases average delay. Adopters see further benefits compared to non-adopters, though even non-adopters experience reduced delays at higher adoption, likely due to the better load-balancing being done by the adopters.

Conclusions and Future Work

Our experiments show that our routing algorithm generally reduces delays for both adopters and non-adopters as

a. Fixed timing plans (Pittsburgh)



b. Surtrac (Pittsburgh)

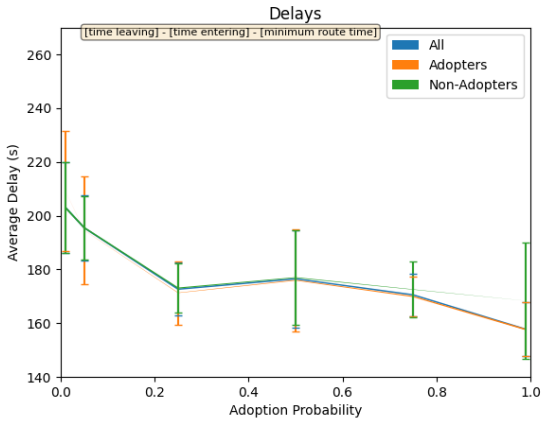


Figure 3: Average delays for all vehicles, adopters, and non-adopters in the Pittsburgh network, for various adoption probabilities. Again, line thickness is proportional to the number of vehicles per run that contribute to that data point. Average values were computed as the mean of 5 runs for fixed timing plans, and error bars show the standard deviation of the mean delays of those runs. **a:** Fixed timing plans using SUMO defaults. **b:** Adaptive timing plans generated from Surtrac.

adoption probability increases. This is in contrast to (Qian and Zhang 2013), which suggests that our approach of explicitly predicting future traffic states and sharing planned routes is effective at dealing with the uncertainty in demand caused by real-time route changes. In addition, our routing algorithm is compatible with adaptive traffic signal scheduling like Surtrac, and the combination of our routing algorithm and Surtrac performs as well as or better than either on its own.

Since adopters tend to get reduced delay compared to non-adopters, if our routing algorithm were implemented in the real world, this would give an incentive for drivers to start using our routing algorithm immediately, instead of needing a critical mass of adopters before any benefits are seen. Despite the fact that non-adopters will still follow their pre-determined routes, we find that they also see reduced delays as adoption probability increases, since adopters can switch away from routes that non-adopters might be over-utilizing.

Despite the benefits, there are some limitations to the current approach that prevent it from being directly implemented on a real road network. First, our current algorithm is slow to run. To increase speed, we timed out any routing calls that took more than a minute (updating the first part of those vehicles' routes if possible), turned off Surtrac's communication between adjacent intersections, and on the Pittsburgh network only ran Surtrac once every 5 simulated seconds; despite this, each of our 99% adoption simulations on the Pittsburgh network still take multiple days. The main contributor to the runtime is calling Surtrac potentially hundreds of times in the routing simulations. Some possible directions toward achieving real-time performance include replacing the Surtrac calls or routing simulations with faster approximations, or running some or all of the routing simulations in separate threads if more cores are available.

Another limitation of this work is that we assume we have perfect information about the number and positions of vehicles in the network at all times. In the real world we would have to extrapolate these from detector readings. Future work will explore adding detectors to the simulation and estimating vehicle positions using those (possibly with added noise) to see if our routing algorithm still performs well with this added uncertainty. Alternatively, with the development of connected vehicle technology, vehicles themselves might soon be able to report their own positions sufficiently accurately that vehicle detection would become less of an issue.

Similarly, the algorithm currently assumes that no new vehicles will enter the network in the middle of a routing simulation. To mitigate this, we included very long input roads in our test networks. However, a more realistic solution, which we intend to explore in the future, would be to introduce new vehicles into the routing simulation based on the rate that new vehicles have been entering the network on each of the input roads.

Finally, while our results look promising on these particular networks, we would like to explore our routing algorithm's behavior in a larger variety of networks and with more traffic signal scheduling algorithms to look for combinations that work particularly well. Such information could lead to new insights on how to further improve vehicle routing.

References

Bagloee, S. A.; Sarvi, M.; Patriksson, M.; and Rajabifard, A. 2017. A mixed user-equilibrium and system-optimal traffic flow for connected vehicles stated as a complementarity problem. *Computer-Aided Civil and Infrastructure Engineering* 32(7):562–580.

Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; and Werneck, R. F. 2016.

Route planning in transportation networks. In *Algorithm engineering*. Springer. 19–80.

Cabannes, T.; Vincentelli, M. A. S.; Sundt, A.; Signargout, H.; Porter, E.; Fighiera, V.; Ugirumurera, J.; and Bayen, A. M. 2017. The impact of gps-enabled shortest path routing on mobility: a game theoretic approach 2. *University of California, Berkeley* 29.

Daganzo, C. F., and Sheffi, Y. 1977. On stochastic models of traffic assignment. *Transportation science* 11(3):253–274.

Friesz, T. L.; Bernstein, D.; Smith, T. E.; Tobin, R. L.; and Wie, B.-W. 1993. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations research* 41(1):179–191.

Goldstein, R., and Smith, S. 2018. Expressive real-time intersection scheduling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Hawkes, A. 2016. Traffic control with connected vehicle routes in surtrac. *Unpublished Masters Thesis, Carnegie Mellon University robotics institute, pittsburgh, pA*.

Lopez, P. A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.-P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; and Wießner, E. 2018. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, 2575–2582. IEEE.

Papageorgiou, M.; Diakaki, C.; Dinopoulou, V.; Kotsialos, A.; and Wang, Y. 2003. Review of road traffic control strategies. *Proceedings of the IEEE* 91(12):2043–2067.

Prashker, J. N., and Bekhor, S. 2004. Route choice models used in the stochastic user equilibrium problem: a review. *Transport reviews* 24(4):437–463.

Qian, Z. S., and Zhang, H. M. 2013. A hybrid route choice model for dynamic traffic assignment. *Networks and Spatial Economics* 13(2):183–203.

Smith, S.; Barlow, G.; Xie, X.-F.; and Rubinstein, Z. 2013. Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 23, 434–442.

Wardrop, J. G. 1952. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers* 1(3):325–362.

Xie, X.-F.; Smith, S. F.; Lu, L.; and Barlow, G. J. 2012. Schedule-driven intersection control. *Transportation Research Part C: Emerging Technologies* 24:168–189.

Xie, X.-F.; Smith, S. F.; and Barlow, G. J. 2012. Schedule-driven coordination for real-time traffic network control. In *Twenty-Second International Conference on Automated Planning and Scheduling*.