Traffic Routing in the Context of a Centralized Driverless System

Brian Wong and Kevin Mu

***Abstract*—Vehicle routing is a very practical problem with many real-world applications. Currently, vehicles traveling to a destination generally route themselves using a GPS service like Google Maps. To avoid the jarring user experience of changing the route midway through the journey, these systems are limited by the constraint that they must decide the entire route at departure time. Furthermore, each vehicle possesses no knowledge about the other cars' future plans. These routing constraints can lead to inefficiencies in the distribution of traffic and thus cause longer travel times. In this paper, we explore whether algorithms designed for a driverless and centralized context, where these constraints do not apply, can reduce the average travel time of vehicles in a graph.**

***Keywords*—vehicle routing, network, graph, algorithm.**

I. Introduction

Imagine a busy city where many drivers want to travel from their homes to some set of destinations. If the number of vehicles in transit is high, traffic congestion can occur. What is the best way to route cars in order to minimize this congestion and lower average travel times? Today, most drivers of vehicles have access to GPS that computes travel routes, either via a dedicated device or an app such as Google Maps. At departure time, a user enters their destination, and the GPS computes an approximate shortest route using distances from its internal map, potentially also taking into account current traffic conditions [5]. There are two potential sources of inefficiency with this approach:

Fixed Route Constraint: The route is predetermined at departure time, and cannot change in response to new traffic conditions. After receiving an initial route, many people first read the step-by-step directions before embarking on the journey in order to get a general idea of the route. As a result, changing the route mid-way would introduce user experience issues. In order to avoid these potentially jarring alterations in the route, most (if not all) GPS providers follow the standard of fixing the route at departure time.

Independence Constraint: Because each car's route is decided independently, one user's GPS cannot factor in the future plans of other cars currently on the road. Without taking into account other cars' future plans, it has been shown that the Nash equilibrium that results is not guaranteed to be Pareto efficient [2, 8, 9]. This means that there is some allocation for which everyone can be better off, but that this allocation cannot be reached in a competitive society. This inefficiency is not only theoretical. Chen et al. claim a perfectly efficient routing system can increase highway efficiency by 20-50% [7].

II. Motivation

In the future, we expect this routing paradigm to change radically with the introduction of smart cars. In particular, the Google driverless car is projected to enter consumer production as early as 2017-2020 [11]. If the economics of driverless cars prove as favorable as anticipated, it is feasible that the Google cars could eventually comprise a majority of cars on the road. This situation would open the door for centralized routing that optimizes for the efficiency of the entire system.

This new class of routing algorithms can take advantage of the fact that routes can be dynamically changed in the middle of travel without incurring any negative effect on the driver, and that routing can be controlled centrally in order to improve the efficiency of the traffic outcome. In other words, the fixed route constraint and the independence constraint associated with current routing technology will be eliminated. The question we address in this paper is whether the removal of these constraints allows for more efficient routing algorithms.

III. Related Work

The search for vehicle routing techniques constitutes an important research problem. Routing algorithms have been and continue to be studied

and improved. In his seminal paper, Dijkstra first presented an efficient algorithm for single-source shortest path routing in 1959 [3]. Modern vehicle routing is obviously much more complex and now takes into account not only physical distances between locations but also traffic conditions, which can have a large impact on travel time [1, 5, 6].

Several authors have explored different routing methods with the aim of improving traffic congestion. Proposed methods involve more accurate traffic-prediction models using Markov chains and other probabilistic tools [1], as well as routing that combines historical traffic data with real-time information to determine ideal routes [10]. However, detailed traffic history is not always available, and it is often desirable to create algorithms free from the assumption that future traffic will resemble past traffic. This is especially important when the city graph has recently undergone changes, perhaps because of the addition of a new highway or the temporary closing of a road for maintenance. Furthermore, with the widespread adoption of driverless cars potentially on the horizon, it is important to explore routing algorithms that relax the two constraints explained in the introduction. Our aim in this paper is to develop and evaluate assumption-free routing algorithms that take advantage of the relaxed constraints inherent in centralized driverless routing.

IV. Simulation

The simulation of our routing algorithms hinges on a model of traffic flow implemented in Python. We describe this model here.

A. Traffic Congestion

In order to model a small city, we use a directed graph. Denote by $R_{a,b}$ the directed road from a to b if such a road exists. Since our graphs are not necessarily complete, there will not be a road $R_{a,b}$ for all pairs of nodes (a, b). If $R_{a,b}$ exists, the number $L_{a,b}$ corresponds to the time it takes to traverse $R_{a,b}$ when there is no traffic. Each road is mapped to a cost function of the form

$$f_{a,b}(n) = L_{a,b} + Tn,$$

such that $f_{a,b}(n)$ is the amount of time required for a car to traverse $R_{a,b}$ when there are n cars ahead of it on the road. T is a global parameter which we call the traffic multiplier. We chose this linear model of traffic congestion because it is simple and transparent, while also conforming to our intuition about traffic. However, this model could of course be substituted with a more true-to-life function based on research with traffic data.

B. Graph Generation

In our graph environment, a link from node a to node b is unidirectional, but b can also be linked to a to simulate a bidirectional, two-way street. A graph with n nodes is generated by first choosing random integer coordinates on an $p \times q$ grid, where p and q are height and width parameters specified by the user. We then use an algorithm to connect each node with its C nearest neighbors, where C, the connectivity constant, is also specified by the user.

We decided to keep the graph small due to computational limitations. Even with the current scope, each simulation takes about ten minutes. To accurately model even a small city the real-world, we would likely need to construct a graph with thousands of nodes and links. It was beyond the scope of this project to model a graph with a very large number of nodes, but there currently exist many potential ways of scaling routing methods via hierarchical routing [1].

As a further note, our algorithms would run more quickly and be able to handle larger graphs in the real world, as the computation could be parallelized and distributed to the onboard computers of the driverless cars.

C. Cars

Each car object is instantiated with a source node and a destination node. It holds data for the last node it departed from and the node it is traveling toward, as well as its progress along that road (measured in number of time steps). Each car also keeps track of the cost of the road it is on (also measured in time steps), which is fixed when the car enters the road. This road cost is determined by the number of cars in front of it currently on that road. Lastly, each car remembers how long it has been traveling; this data will be used to evaluate the routing algorithms once all cars have arrived at their destinations.

D. Traffic Flow

Our simulation works by discretizing time and performing one iteration at each time step. Each iteration is composed of a four-step process.

1) Step 1: First, we increment each car's progress on the road it is on. If this increases the progress beyond the cost of the road, we update the car's current node to reflect its arrival at a new node (the other endpoint of the current road). This new node is added to a set that holds all of the junctions containing cars; these are precisely the nodes for which routing information will be required.

2) Step 2: In step 2, we reassess traffic information. Denote by $Q_{a,b}$ the number of cars currently on the road $R_{a,b}$. The calculation of Q involves a straightforward loop through the cars, incrementing the corresponding $Q_{a,b}$ for the car's current node and next node.

3) Step 3: Once the Q values are determined, they can be fed into the cost functions for the graph to determine the current cost $S_{a,b}$ of each path:

$$S_{a,b} = f_{a,b}(Q_{a,b}).$$

4) Step 4: Step 4 is the routing step and its implementation is dependent on which routing algorithm is used. Using the cost graph calculated in Step 3, each algorithm must produce the next hop for all cars currently waiting at junctions.

V. Algorithms

A. Naive Baseline

This is a naive routing algorithm that does not take into account changing traffic conditions. The real-world analog would be a situation where everyone uses a GPS device that only routes based on road distances and not current traffic. We implemented and evaluated this naive baseline in order to verify our initial assumption that taking into account live traffic conditions provides significant efficiency gains.

The implementation of this algorithm is straightforward; at the start of the simulation, we compute a cost graph based on zero traffic conditions. Applying Dijkstra's algorithm to this zero traffic cost graph gives us a naive routing table that we hold constant throughout the simulation and use to route all cars.

B. Fixed-Route Baseline

This algorithm is based on the standard GPS model; at departure time, current traffic conditions are evaluated and used to decide upon a route that stays fixed during the car's travels.

The implementation of the Fixed-Route Baseline involves the addition of a route attribute to the cars. At each time step, this algorithm evaluates current traffic conditions and then creates a routing table using Dijkstra's algorithm. Each car instantiated at this time step is seeded with a route based on this routing table. The car then follows this fixed path from source to destination without adapting to changing traffic conditions. In other words, Step 4 (detailed above) does not use the results of Step 3 to route cars, instead using the fixed route specific to each car. We implemented this baseline for two reasons:

1. Comparing this algorithm with the Naive Baseline offers concrete proof that taking into account traffic conditions is essential to efficient routing algorithms.

2. This baseline is modeled after the real-world functioning of current GPS-based car routing. Since these systems are restricted by the lack of centralization as well as the fixed-route constraint (which derives from the fact that human drivers would be averse to last-minute route changes), comparing this baseline with the following two algorithms demonstrates the efficiency gains that are possible with a centralized driverless car system.

C. Dynamic Route

This algorithm explores the benefits of relaxing the fixed-route constraint, which is the first benefit of a driverless car system. Each car decides where its next hop will be only when it needs to–once it arrives at the next junction. For example, if the car C is currently traveling on the road $R_{a,b}$, it will not decide which road to take after reaching b until it reaches b. Obviously this is not feasible for a human-driven car because the driver would not want to be in a constant state of wondering about its route, but this kind of dynamic routing can be exploited by a driverless car.

Note that in practice, the calculation of the next hop would likely take place a few time steps before actually arriving at a junction (since the driverless

car would still need to know which turn to take slightly in advance in order to position itself in the correct lane). In our simulation, we do the calculation upon arrival at the junction for simplicity; this will not significantly overestimate efficiency unless there are wild second-to-second fluctuations in traffic (which is almost never the case).

At an implementation level, this algorithm uses the cost graph constructed in Step 3 of the simulation to create a routing table by running Dijkstra's from each node where cars are currently stationed. This routing table is then used to route cars in Step 4.

D. Centralized Dynamic Route (CentDyn)

In the Dynamic Route algorithm described above, there is just one routing table generated at each time step. However, many cars are routed at each time step. Arbitrarily order these cars from 1 to k. When the nth car is routed, the routing table is outdated because it does not take into account the newly decided paths of the first $n-1$ cars routed at this time step. In a non-centralized setting, it is impossible to take the other cars' routes into account (because route information is not shared). However, if a majority of cars on the road are Google cars, route sharing enables more efficient planning.

In order to simulate this route-sharing, we allow the nth car to route itself based on the route decisions of cars $1, 2, \ldots n-1$. For the Google driverless car system, this could be enabled if each car broadcasted its next hop to all the other cars when it calculates what the hop should be. As noted in **C**, the cars will decide their next hops slightly in advance of arriving at their next junction; this information about their future plans will aid in the routing of other cars.

At the implementation level, this algorithm updates the cost graph constructed in Step 3 k times at each iteration, where k is the number of cars that are being routed. Therefore the nth car can take advantage of the route data from $n-1$ other cars.

VI. Performance

To compare the performance of our algorithms, we perform 400 simulations on a graph with 12 nodes; each simulation runs until 30 cars have arrived at their destinations. Each simulation gives us an average total journey time for the 30 cars. We then average these numbers across the 400 simulations and use the new average journey time as our metric.

Because of computational limits, we could not have too many cars traveling at one time. However, modeling substantial congestion is an important aspect of the performance comparison. In order to simulate congestion without using a computationally intractable number of cars, we decided to do our performance evaluation with a single source and a single destination. At each time step 4 cars are spawned.

A. Four Algorithms

First we compare the performance of the four algorithms against a standard model where $C = 6$ and $T = 8$. (From III, recall that C is the connectivity constant and T is the traffic multiplier.) Average journey times for the algorithms are depicted below.

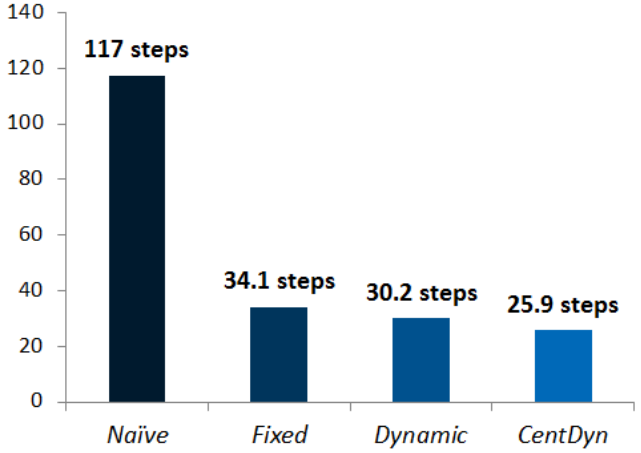

Figure 1. *Average number of timesteps per car for complete journey when* $C = 6, T = 8$.

The large delta between the performance of the Naive Baseline and that of the Fixed-Route Baseline immediately verifies that adapting to changing traffic conditions is essential for efficient routing. Furthermore, we observed an 11.4% decrease in average travel time when moving from the Fixed-Route Baseline to the Dynamic algorithm, and a further 14.2% decrease when CentDyn was used.

B. Varying the Connectivity Constant

We also sought to investigate how the parameters of our model, the connectivity constant and the traffic multiplier, affected the relative performance of our algorithms. In the following diagram we fix

$T = 8$ and compare $C = 6$ with $C = 8$. We do not depict the performance of the Naive Baseline (which was above 100 for all simulations) in order to highlight the more nuanced differences between the other three algorithms.

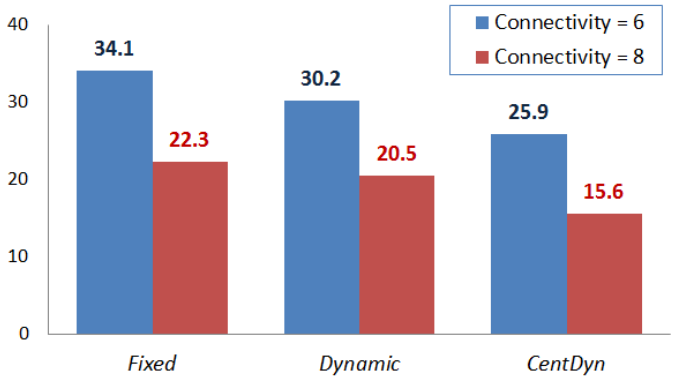

Figure 2. *Graph showing the effect of the connectivity constant on performance. Average number of timesteps per car for complete journey when $T = 8$ is fixed.*

When connectivity was increased, we found that CentDyn outperformed Dynamic by a wider margin: 23.9% instead of 14.2%. We believe that this is because the extra edges allowed for more implicit coordination among cars traveling in the same locality.

C. Varying the Traffic Multiplier

In this last comparison, we fix $C = 6$ and compare $T = 8$ with $T = 10$.

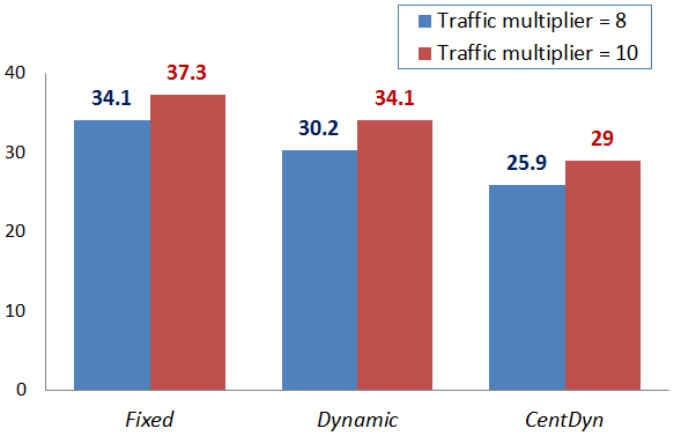

Figure 3. *Graph showing the effect of the traffic multiplier on performance. Average number of timesteps per car for complete journey when $C = 6$ is fixed.*

First note that the travel times measured when $T = 10$ were higher than their counterparts when $T = 8$. This makes sense since increasing T amplifies the slowing effect of traffic on road speed. The performance of CentDyn relative to Dynamic was about the same in both simulations (14.9% vs. 14.2%), suggesting that the efficiency gains of CentDyn are stable in the face of varying traffic sensitivy.

VII. Conclusion

In this project, we found that by relaxing the fixed-route constraint and the independence constraint inherent in current GPS systems, we can achieve significant decreases in average travel time. Our best algorithm, the Central Dynamic Route protocol described above, addresses both of these potential inefficiencies by exploiting dynamic, often last-minute routing, as well as route-sharing between cars in the same locality. We found that this algorithm leads to a 24% average travel time decrease compared to the current GPS model (see Figure 1) in our simulation environment.

In the future, we would like to continue our investigation by expanding the simulation to more accurately model the real world and its traffic conditions. For instance, we would like to implement hierarchical routing to accommodate larger graph sizes. Furthermore, a more accurate cost function could be constructed based on past congestion data and used to make our simulation more accurate. There are already several existing methods of traffic prediction, so the next step would be to combine our algorithms to see if further improvement can be obtained.

With a Google car revolution potentially just a few years away, more research on this important practical problem has the potential to significantly reduce traffic and travel times in congested cities.

VIII. References

[1] Yuan, J., Zheng, Y., Xie, X., & Sun, G. (2011, August). Driving with knowledge from the physical world. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 316-324). ACM.

[2] Vetta, A. (2002). Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In Foundations

of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on (pp. 416-425). IEEE.

[3] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische mathematik, 1(1), 269-271.

[4] Tam, Donna. (2012). Google's Sergey Brin: You'll Ride in Robot Cars within 5 Years. Cnet. N.p., 25 Sept. Web.

[5] Chen, C., Jia, Z., & Varaiya, P. (2001). Causes and cures of highway congestion. IEEE Control Systems Magazine, 21(6), 26-32.

[6] Kaufman, D. E., Smith, R. L., & Wunderlich, K. E. (1991). An iterative routing/assignment method for anticipatory real-time route guidance (No. 912815). SAE Technical Paper.

[7] Chen, C., Jia, Z., & Varaiya, P. (2001). Causes and cures of highway congestion. IEEE Control Systems Magazine, 21(6), 26-32.

[8] Arnott, R., De Palma, A., & Lindsey, R. (1991). Does providing information to drivers reduce traffic congestion? Transportation Research Part A: General, 25(5), 309-318.

[9] Arnott, R., & Small, K. (1994). The economics of traffic congestion. American Scientist, 446-455.

[10] Kim, S., Lewis, M. E., & White III, C. C. (2005). Optimal vehicle routing with real-time traffic information. Intelligent Transportation Systems, IEEE Transactions on, 6(2), 178-188.