DUE-STR: A Heuristic Extension Of The Selfless Traffic Routing Model Utilizing Dynamic User Equilibrium

---

A Senior Honors Thesis Presented to the

Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment of the Requirements for

the Degree of Bachelor of Science

By

Thomas Carroll

May 2022

DUE-STR: A Heuristic Extension Of The Selfless Traffic Routing Model Utilizing Dynamic User Equilibrium

_____

Thomas Carroll

APPROVED:

_____

Albert M. K. Cheng, Thesis Director
Computer Science

_____

Carlos Rincon,
Computer Science

_____

David P. Shattuck, Honors reader
College of Engineering

_____

Dan E. Wells, Dean
College of Natural Sciences and Mathematics

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Professor Albert M. K. Cheng for his amazing support and guidance this last year. I would not have been able to make it this far without having encountered Professor Cheng during his operating systems course. For this opportunity and the myriad of other opportunities given to me by Professor Cheng, I am immensely grateful. I hope that I can repay Professor Cheng for these wonderful opportunities through hard work and dedication to research and teaching under his tutelage. I hope to be able to continue to work with Professor Cheng and the real-time systems lab as a graduate student in pursuit of a Ph.D.

I would like to also thank Guangli Dai of the real-time systems lab for his incredible help organizing and writing the material used in this thesis. I learned an incredible amount working under Guangli, with his incredible problem-solving skills and competence in the field of computer science being a goal I strive for every day. It truly was a wonderful opportunity to be able to work with such a talented and intelligent person.

I would lastly like to thank my parents for their incredible support throughout my college experience. I would not be in the incredible position I am today without their support, love, guidance, and persistent encouragement to get the most out of higher education.

DUE-STR: A Heuristic Extension Of The Selfless Traffic Routing Model Utilizing Dynamic User Equilibrium

---

An Abstract of a Senior Honors Thesis

Presented to the Faculty of the

Department of Computer Science

University of Houston

In Partial Fulfillment of the Requirements for

the Degree of Bachelor of Science

By

Thomas Carroll

May 2022

ABSTRACT

Routing vehicles through a traffic network such as a modern-day city has been a much-studied topic, with routing algorithms such as Dynamic User Equilibrium (DUE) having been well documented. The focus of many such works has been on the optimization of average travel time through traffic networks aiming for the more efficient routing of vehicles. In this thesis, we outline our plans for routing to satisfy arrival deadlines, where vehicles are routed with the primary objective of getting somewhere on time. We consider vehicle routing through a smaller section of a city, known as a traffic sub-network, using a centralized scheme as a guiding traffic assignment agent. We introduce our preliminary implementation of a routing algorithm built on the Selfless Traffic Routing (STR) model and Dynamic User Equilibrium (DUE) to show the viability of such a scheme on a traffic network. We present our experimental results from running this scheme on a real-world traffic network. We consider a pre-vehicle movement rerouting scheme capable of being competitive against more informative real-time models. We evaluate DUE-STR and these models using the number of arrival-deadline misses and the average travel time performance metrics for vehicles. We find mixed results between DUE-STR and other models, with our DUE-STR model mostly having better results when considering deadline misses and mostly having worse results when considering average vehicle travel time. We explore reasons why the results may not be quite as good as well as potential solutions to solve these issues.

Table of Contents

List of Figures

Introduction[1]

Getting to work on time in a modern city is slowly becoming more and more difficult as the average travel time of a one-way commute hit a new high in the U.S. of 27.5 minutes in 2019 [1]. Among several strategies proposed to reduce congestion [2], routing traffic more efficiently is the vital option that we focus on. Intuitively, studying methods for better vehicle routing typically focuses on reducing average travel time for all vehicles throughout the network, by controlling congestion [3]. However, exclusively focusing on travel time can lead to drivers arriving late to a destination as solely focusing on travel time does not consider individual driver deadlines. Since most people travel with a known arrival deadline to their respective destinations, deadline-prioritized routing algorithms can reduce congestion while minimizing the impact on people's daily plans compared to those that do not consider travel deadlines. To satisfy the travel deadline of each vehicle, this thesis proposes an efficient approach while also minimizing road congestion, i.e., the average travel time of all vehicles.

Our solution is based on a few assumptions, most namely that all the vehicles in the network are autonomous self-driving fully controlled vehicles, routed by a centralized traffic routing agent, with full a view of the traffic network. As will be elaborated upon, each routing scheme must be able to answer two cardinal questions fundamental to vehicle routes: “how to route vehicles”, and “how to predict vehicle travel time on a certain route”. We cover how other methods

and our centralized routing controller are able to answer both of these questions.

To route vehicles based on their deadlines while minimizing the traffic congestion in a network, a centralized routing system is needed. However, as [4], [5] point out, such a large-scale centralized routing system is intractable with real-time requirements on a huge amount of data transmission and computation. Therefore, our scheme divides a large network into multiple sub-networks, where each sub-network is small in scale so that a centralized routing system is viable inside the sub-network [6]. An example of a large network is shown in Figure 1, with the boundaries of the many subnetworks outlined in different colors. In this thesis, we focus on developing an efficient centralized deadline-prioritized routing algorithm within each sub-network.

Figure 1: An Example of a Network and the Divided Sub-Networks

The Selfless Traffic Routing (STR) model [7] has been proposed to solve the routing problem inside a sub-network. Here, ``selfless" means some vehicles may be routed to a route with longer travel time to reduce road congestion while their deadlines can still be satisfied. The STR

model is still in its infancy [7]; as such our contributions to expanding it include:

1) using an approximated Dynamic User Equilibrium (DUE) for alternative path generation to create a novel vehicle routing algorithm geared towards reducing deadline misses and

2) showing our algorithm's efficacy through experiments using Simulation of Urban MObility (SUMO) [8].

The remainder of the thesis is thus organized as follows:

- In Chapter 1 we briefly review the STR model and DUE solution concept as well as cover relevant topics making up the core theory behind our model.
- In Chapter 2 we introduce the proposed novel approach DUE-STR.
- Finally in Chapter 3 our experimental evaluations of DUE-STR are presented and discussed.

Chapter 1 Related Work[2]

Examples of currently implemented, publicly available tools used to help drivers reduce their travel time include Waze [9], Google-maps [10], and Houston TranStar [11]. With support from these tools, as well as future self-driving technologies such as those from Tesla [12] and Waymo [13], the potential for a fully autonomous grid of self-driving cars becomes a real possibility in the future. To this end, routing algorithms concerning autonomous vehicles have been studied by various other researchers [4], [3], [14], in order to produce better routing policies for the autonomous vehicles of the present and future. However, most of these methods have traditionally focused on reducing travel time or "cost" for each individual vehicle. To build up to DUE-STR we first discuss earlier algorithms including Dijkstra's algorithm, the traffic assignment problem (TAP), Dynamic User Equilibrium (DUE), one-shot assignment, and finally the selfless traffic routing (STR) model. Reviewing these models as well as how they are related to the two cardinal questions, will give us a basis for how routing has been calculated in previous methods and how our method builds upon them. As mentioned before the two cardinal questions of vehicle routing include how to route vehicles and how to predict the travel time of a vehicle using said route. We begin creating an understanding of these two questions by considering a basic solution to answer the question of "how to route vehicles": Dijkstra's Algorithm.

[2] Content in this chapter has been taken from the previously published work "Work In Progress: A Solution Based on Dynamic User Equilibrium Toward the Selfless Traffic Routing Model" By Thomas Carroll, Albert M. K. Cheng and Guangli Dai [30].

1.1. Previous Models and Work

1.1.1. Dijkstra's Algorithm

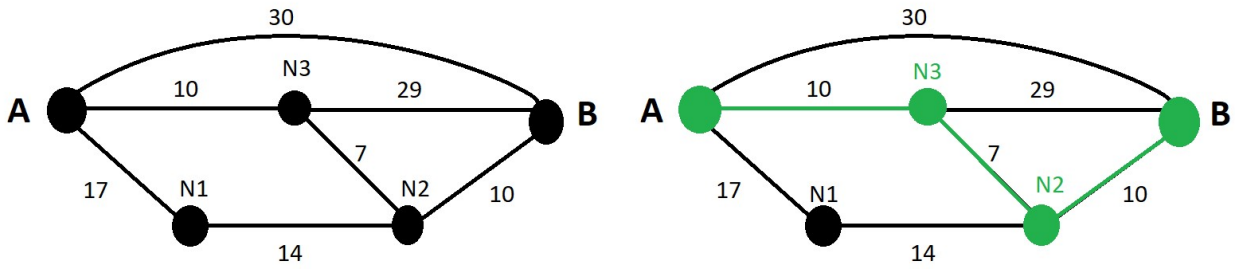

Figure 2: A Shortest Path Solution.

The most primitive version of cost optimization in a route-based problem can be found in the shortest path algorithm, Dijkstra's Algorithm [15]. Dijkstra's algorithm considers a basic connected graph consisting of dots and lines between them (Figure 2), referred to hereafter as nodes and edges respectively, where each edge has an associated "cost". The problem Dijkstra's algorithm is trying to solve is finding the path with the lowest associated cost between two nodes. In the case of Figure 2, we try to find the path with the least cost between the origin node A and destination node B. The set of origins and destinations, in terms of vehicular routing, is formally known as an Origin-Destination (O-D) matrix, which contains the origins and destinations of all vehicles in a system. Dijkstra's algorithm solves the problem by starting at the origin node and exploring the nodes immediately connected to the origin node. Dijkstra's algorithm then moves to nodes one by one documenting their connections and associated cost to one another, resulting in the shortest path highlighted in green in the 2nd image in Figure 2 (A->N3->N2->B).

We can relate this specific example to vehicle routing by considering each node as an intersection and each edge a street between two intersections with an associated predicted travel time, we will refer to this method as the shortest path method further ahead.

With a basic solution for the question of "how to route vehicles", we would now like to look at how we might apply a solution to this question on a more theoretical level while considering real-time traffic conditions.

1.1.2. The Traffic Assignment Problem

While finding the shortest path using Dijkstra's algorithm may be optimal in a static system with no change in the cost of the edges, routing multiple vehicles may change the cost by creating congestion on certain edges. For example, if we routed 100 vehicles down the shortest path in Figure 2, we may end up with a congested path, meaning that a longer path may end up with a better travel time. This type of problem is formally known as the Traffic Assignment Problem (TAP) [16], with the overall goal being to minimize travel time for all vehicles in the network by assigning different amounts of vehicles to various routes. To answer the first cardinal question "how to route vehicles" for the TAP, we consider a solution by John Glen Wardrop, who presents two principles of Wardrop equilibrium that can solve the TAP [17]. The Wardrop equilibrium considers all vehicles in the network as independent agents aiming to maximize their overall gain in a game-theoretic way, which, in the case of a TAP, is minimizing individual travel time. For such a scenario, Wardrop states that a type of equilibrium will form in the traffic network when given a set of routes from node A to B where the travel time of any used routes are all the same and no greater than the travel time of an unused route [17]. This is known as user equilibrium, the

first of the two equilibriums, as no single agent can make a choice that is any better or worse than their current choice.

The second of the two equilibriums is known as system optimal equilibrium; here instead of individual vehicles behaving selfishly, vehicles work together to reduce the average travel time of all vehicles in the network. This version of equilibrium is the version that fully autonomous networks of self-driving vehicles bring into the realm of possibility. While we do not focus on modifying or aiming for system optimal equilibrium, it is worth mentioning since the selfless traffic routing model applies the idea of vehicles working together in pursuit of an objective [7], which will be elaborated on more in Section 1.2.

To give an example of user equilibrium in practice we can consider the diagram in Figure 3 [18]. We consider a diagram of two nodes, an origin A and destination B (a single entry in an O-D matrix), with two possible routes and an equation denoting the time it would take to travel down a route as a function of the number of vehicles currently assigned to the route per hour (x). Here the y-intercept for these functions represents the default amount of travel time a vehicle would take to travel down the route based on general parameters like speed limit and distance. This linear function is what answers the cardinal question: "how to predict vehicle travel time on a certain route" for the TAP. We consider the number of vehicles to be in the unit of thousand/vehicles-per-hour and the time to be in minutes. For this example, we consider 15,000 vehicles per hour (15 thousand/vehicles-per-hour), and to achieve a state of user equilibrium in this diagram we want to assign vehicles to each value of x such that $t1 = t2$.

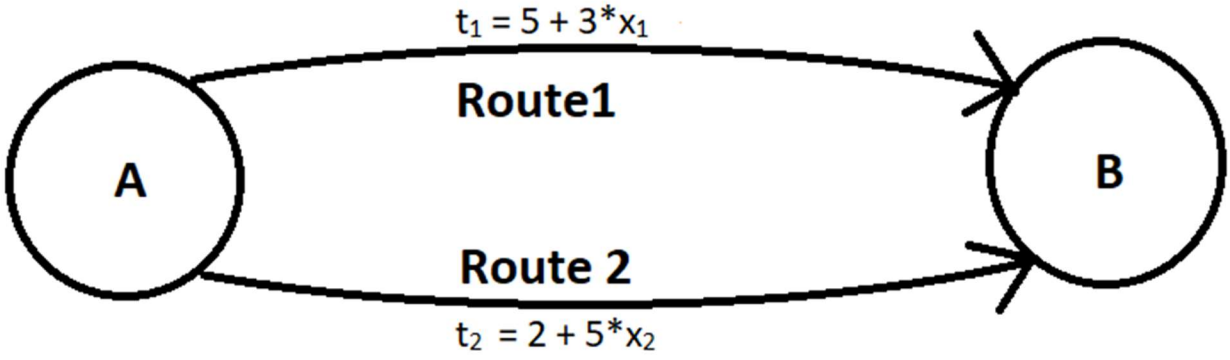

Figure 3: A Traffic Assignment Problem

We calculate the solution to Figure 3:

Given:

$$x1 + x2 = 15, t1 = t2, t1 = 5 + 3 * x1, and\ t2 = 2 + 5 * x2$$

We can solve for $x1$ and $x2$:

$$5 + 3 * x1 = 2 + 5 * x2$$

-> $$x2 = \frac{3 + 3 * x1}{5}$$

-> $$x1 + (3 + 3 * x1)/5 = 15$$

-> $$x1 = 9, x2 = 6$$

$$t1 = 32 \text{ and } t2 = 32$$

We can thus assign 9,000 vehicles to route 1 and 6,000 vehicles to route 2 ($x1 = 9, x2 = 6$) to get an even travel time down both routes of 32 minutes. Thus, from the perspective of a driver attempting to move from A to B both routes are equally as good, resulting in a fully realized user equilibrium for this system. This TAP example outlines how a route controller might route vehicles to create user equilibrium in this example network, however actually implementing a similar scheme in a real network such as the one in Figure 1 is significantly more complicated.

1.1.3. Dynamic User Equilibrium

A big assumption is made when calculating a TAP, specifically the use of a single function that can perfectly calculate travel time based on the number of vehicles assigned to a route. Finding such a function in a simulation like SUMO, much less in real life, is virtually impossible to create on the same level of accuracy as a TAP. Factors such as vehicles that do not match the speed limit, the timing of traffic lights, different vehicle accelerations, different vehicle braking speeds, etc. on top of the complexity of the network, are to blame for this. As a result, the ideal equations of a TAP from Figure 3 are mostly theoretical. So, if these equations are theoretical how can we attempt to approximate them in a real-world example and answer the question "how to predict vehicle travel time on a certain route"? A simple solution is to use historical trips of vehicles that have previously traveled the network and record the conditions of the network at the time of travel.

Using historical predictions has been used in the past to predict travel time [19], [20] [21] to predict vehicle travel time, however these techniques produce approximations of travel times and not the idealized functions of Figure 3. As mentioned before creating a perfect travel time prediction function is practically impossible; this will result in some error in the number that we use to represent travel time. This error will thus affect the user equilibrium in Figure 3, where ideally we would want to have the x values as $x1 = 9, x2 = 6$, thus we may have to settle for an approximation of $x1 = 7.8, x2 = 7.2$ due to the approximation of travel time. This reveals that to achieve user equilibrium in practice solving two approximation problems is required, in other words, the two cardinal questions cannot be made optimal in a reasonable time and must thus be approximated. This means approximating the travel time of a vehicle from A to B and then

creating an approximation of how many vehicles to assign to each route to equalize travel time to user equilibrium like Figure 3, all while considering the changing state of the traffic network. We would like to look at the method that approximates both problems, by considering the user equilibrium approximation method, Dynamic User Equilibrium (DUE), a type of Dynamic Traffic Assignment [22].

With continuously changing traffic, we need DUE, which aims to create user equilibrium by assigning routes based on changing real-time conditions. A key challenge here is that DUE requires the travel time of vehicles to create the user equilibrium, while the travel time of vehicles depends on current traffic conditions, which in turn depends on vehicle route assignment handled by DUE. Due to the inter-dependency between the travel time and the vehicle route assignment determined by DUE, an iterative process must be executed aiming to approximate a DUE. This is implemented in Simulation of Urban Mobility (SUMO) [8] in the form of duaiterate [23], meaning that to find a good approximation of user equilibrium duaiterate must execute all vehicle trips recursively. Duaiterate makes changes to the routes of vehicles between iterations by considering the travel time of each vehicle during the previous iteration. Thus, duaiterate is able to use previous travel times to solve the problem of approximating travel time, and the changes in routes to approximate the user equilibrium in Figure 3 on a network-wide scale. This comes at the tradeoff of the resulting routes only being approximations of ideal user equilibrium routes and the large computation time needed to run this operation. For reference, running duaiterate on SUMO with 100 iterations can take upwards of two minutes, with various levels of improvement and sometimes regression in average travel time against the control shortest path assignment.

A heuristic of DUE known as one-shot assignment [22] is also provided in SUMO in the

form of one-shot.py [24]. A One-shot assignment attempts to do what DUE does during execution, with the tradeoff of not explicitly aiming to create a perfect user equilibrium. This is done by periodically updating the travel time of various edges in the network and assigning vehicles to routes based on the travel time of the routes at the time a vehicle requests to be routed (when they enter the network). Computationally, one-shot.py is able to generate routes this way in about 2 seconds, which is roughly the time a single iteration of duaiterate takes to complete. Performance-wise, the resulting routes are also rather mixed in their performance, typically doing slightly worse or slightly better than the control shortest path in terms of average travel time. The results can be seen in Chapter 3, Figure 6, where we evaluate the performance of DUE-STR.

It should be noted that the computation times of duaiterate and one-shot.py were derived from a computer with a 4 core i7-4790k at 4 GHz with 16 Gb of RAM, with Python using roughly 30% of CPU time.

1.2. The Selfless Traffic Routing Model

The STR model first proposes the idea of “selfless" in traffic routing. The goal of the STR model is to guarantee the travel deadline of each vehicle while minimizing the average travel time of all vehicles. More specifically, this means a vehicle with a closer deadline (V_1) will be routed down paths with shorter expected travel times and a vehicle with a farther deadline (V_2) will be routed down paths with longer expected travel times [7]. This creates a non-optimal route for V_2 whereby this vehicle is routed “selflessly”, allowing V_1 to have a greater chance of meeting its deadline at the cost of a longer trip for V_2. In essence, the STR model provides us with a

methodology for solving the cardinal question of "how to route vehicles". Together with the STR model, a testbed, based on Simulation of Urban MObility (SUMO) [8] is built to evaluate the performance of different routing policies [7]. Our experimental evaluations in Chapter 3 are developed based on this testbed.

In other previous works that study the traffic routing problem [3], [16], [25], reducing average travel time was the main goal, while the deadline of each vehicle is ignored in these models. While routing vehicles with the goal of minimizing average travel time is a viable way to increase the efficiency of road systems, not considering the travel deadlines of individual drivers can have disastrous consequences. Considering the example in Figure 3, we aimed to equalize travel time throughout the network, but by doing so have created a "selfish" system where individual drivers are looking for the route with the lowest travel time at their own time of travel. This can cause problems for vehicles with travel deadlines, as their choice of routes will both be of the same travel time. If no route provides a travel time quick enough to allow deadlines to be met, then many people may be late to arrive to their destinations. This is an issue that we consider and are able to get around through the use of a tool in SUMO in our implementation of DUE-STR, with further details provided in Chapter 2.

1.2.1. Division of Networks into Sub-Networks

Minimizing deadline misses in a real-time network has been explored before [4], [5], however, these implementations have focused on decentralized strategies. A decentralized strategy considers local information and general global information such as route distance [5], which means the routing algorithm cannot have real-time routing information, i.e., the actions of other vehicles

in the sub-network. While a centralized strategy can help utilize the network more efficiently, it requires a much larger amount of data transmission and computation. Thus, as discussed in the introduction, we divide large-scale networks into smaller sub-networks [6] and develop centralized deadline-prioritized routing algorithms in each sub-network.

In our current implementation, we do not consider the optimal size and area a subnetwork should cover, but rather estimate what a sub-network could potentially be. While this scheme may be fine for local streets in a non-freeway setting, considering small subnetworks for small stretches of freeway especially with no entrance or exits present seems like an obvious waste. We think that if ideal splits between subnetworks are to be considered, there must be a split between types of subnetworks based on the type of road considered.

Chapter 2 DUE-STR[3]

Our model is built on SUMO, taking full advantage of its demand modeler tool DUAROUTER [8] and DUE approximator tool duaiterate [23]. We aim to combine the STR model with DUE to route vehicles in such a way to avoid deadline misses, answering the cardinal question of "how to route vehicles". Our proposed approach, referred to as **DUE-STR,** intends to swap the routes between vehicles so that vehicles with closer deadlines are given routes with lower travel time in a single pre-run step. Where a run is when the vehicles traverse the network along their routes along with all relevant pre-movement information for vehicles including start time, origin, destination, etc. Thus, since DUE-STR only considers this pre-movement information, DUE-STR modifies the routes of vehicles prior to any vehicle movement with no real-time updates from DUE-STR. We use a K-Nearest Neighbors (KNN) regression model to predict the travel time for each route [26], which answers the cardinal question "how to predict vehicle travel time on a certain route". Specifically, DUE-STR aims to use the approximated DUE created by duaiterate, so route swaps between vehicles are done such that no new routes are created, but rather the route a swapped vehicle travels is different from its original.

We mentioned earlier in Section 1.2 that user equilibrium has little to no room to consider rerouting vehicles due to the equilibrium of travel time created throughout the routes, giving us no ability to reroute vehicles with deadlines that cannot be met by the current travel time provided by

[3] Content in this chapter has been taken from the previously published work "Work In Progress: A Solution Based on Dynamic User Equilibrium Toward the Selfless Traffic Routing Model" By Thomas Carroll, Albert M. K. Cheng and Guangli Dai [30].

the available routes. We get around this issue by considering that duaiterate can, on average, only manage to create an approximation of user equilibrium. This means that rather than routing vehicles in a perfect split like in Figure 3 (9000 route 1, 6000 route 2) an imperfect approximation is made such as (7900 route 1, 7100 route 2). This creates an imbalance of travel times between the two routes, meaning that we are in fact not working with a perfect user equilibrium. This actually works in our favor since we can analyze the results of this imperfect user equilibrium by looking at the estimated travel times of each route. We can then swap vehicle routes based on the estimated travel time of each route created by duaiterate, prioritizing vehicles with low deadlines, while also still having a decent DUE approximated set of routes aimed at improving overall travel time over the shortest path method.

In our current setup of DUE-STR, vehicle trips are generated before any vehicle movement, as mentioned before, the problem with this is how to solve the problem of getting good predictions of vehicle travel time before execution. We begin by first collecting data on trips through the sub-network using a more primitive version of DUE-STR. This primitive version of DUE-STR generates reroutes similar to our current implementation of DUE-STR but with much less efficacy. We use these primitively generated reroutes to generate historical route data, recording the start time, distance, number of edges, number of potentially encountered vehicles, a simple estimated travel time (sum-of-distance/speed-limit for all edges in a route), and finally actual travel time. With this data, we can ensure that acceptable travel time prediction happens before a run. For our purposes in this thesis, a KNN regression agent, trained on our collected data for the red sub-network in Figure 1 is used to predict travel time. To get a prediction, we input the same data that we collected, namely the start time, route length, number of edges, number of potentially

encountered vehicles, and estimated travel time and we receive a predicted travel time for that route. The prediction agent has a mean absolute error (MAE) of ≈74.9 seconds and a mean absolute percentage error (MAPE) of ≈30.26%. MAE is a measure of average absolute error between the actual and predicted values. MAPE is a measure of the percentage of the average difference between the predicted and true values.

Mean absolute error is calculated:

$$\boldsymbol{MAE} = \left(\frac{1}{n}\right) * \sum_{i=1}^{n} |yi - xi|$$

Mean Absolute percentage error is calculated:

$$\boldsymbol{MAPE} = \left(\frac{1}{n}\right) * \sum_{i=1}^{n} \left|\frac{yi - xi}{yi}\right|$$

Where n is the number of entries, y is the true value, and x is the value predicted by the model.

This allows us to put the numbers 74.9 seconds and 30.26% into perspective meaning our KNN regressor agent has an average absolute error of 74.9 seconds and 30.26% average difference between the true and predicted values. Since the travel times are in seconds and we are dealing with trips in the hundreds of seconds, the accuracy of the KNN travel time prediction agent is acceptable for our experiments.

The goal of our experimental setup is to showcase our routing scheme by simulating an area of high traffic, with vehicles seeking to simply pass through the sub-network by a given time, represented by a local-sub-network deadline. We consider the sub-networks in Figure 1, specifically the one outlined in red with green dots labeled A-F indicating origin-destination points. The focus here is to consider the main roads, thus dots N1 and N2 are excluded from our selection

for being connected to neighborhoods, as they are less likely to be used by a vehicle simply looking to pass through the sub-network. Following these rules, trips start at one of these points and end at another, generating an O-D matrix based on these criteria.

With an O-D matrix and network as input, we can generate the shortest path base routes by using DUAROUTER. DUAROUTER uses Dijkstra's algorithm to output a base routes file explicitly defining the route of each vehicle. This base route file is taken alongside the network file as input for duaiterate. Duaiterate uses the generated shortest path from the DUAROUTER base trips to reroute vehicles throughout the network. A reroute occurs by changing a vehicle's decision at an intersection thus changing a vehicle's route. Duaiterate outputs a set of DUE-Approximated ν (nu) routes that include several reroutes of vehicles away from the shortest path. We run duaiterate iteratively as mentioned in Chapter 1, setting the number of iterations to 100, typically giving us improved performance. The resulting output is a set of new (ν) DUE approximated trips, with various vehicle reroutes throughout the sub-network.

With the DUAROUTER base routes and duaiterate ν routes in hand, we have our input for DUE-STR. We begin by taking all vehicles and sorting them in increasing order of deadlines into a list V. We iterate through V, considering vehicles with the lowest deadlines first. We consider each vehicle one at a time, denoted by Vi. Vehicles ahead of Vi in the list are considered next, denoted by Vj where $j > i \, \forall \, Vj \in V$. Next, it is ensured that Vi and Vj have the same destination, otherwise, the current Vj is skipped. If these conditions are cleared, we move to the core of the algorithm, where multiple conditions are checked to fully execute a swap.

2.1. Algorithm 1: DUE-STR

Input: Shortest Path Base Routes from DuaRouter, DUE-Approximated ν Routes from Duaiterate

Output: Deadline-Prioritized DUE-Approximated Final Routes

1: Compare Base and ν routes to identify and list in E, all shared e edges.

2: Sort all Vehicles into a list V in increasing order of deadlines.

3: **for** $i = 1 \rightarrow |V|$ and $j = i + 1 \rightarrow |V|$ **do**

4: **if** Vi and Vj do not have the same destination **then**

5: continue

6: **end if**

7: **if** ∃ edge $e \in E$ shared by Vi and Vj and edges after e are completely different, except for the destination, in the ν route & the expected travel time for routes after e of Vi is than that of Vj t**hen**

8: swap the edges after e of Vi and Vj.

9: **end if**

10: **end for**

For two vehicles to be viable candidates for swapping, we refer to line 7 in Algorithm 1. There is a total of three conditions for a swap to occur that all revolve around the existence of a shared edge e between Vi and Vj. A shared edge is an edge that is present in the original shortest path routes of two or more vehicles that share the same destination. The shared edge is then an edge where one or more of these vehicles reroute away from the shortest route to an alternative

route as generated by duaiterate. We thus begin by identifying all shared edges e using the base and ν routes marking them all in a list E. This streamlines the process of searching for shared edges during swapping later on in the algorithm.

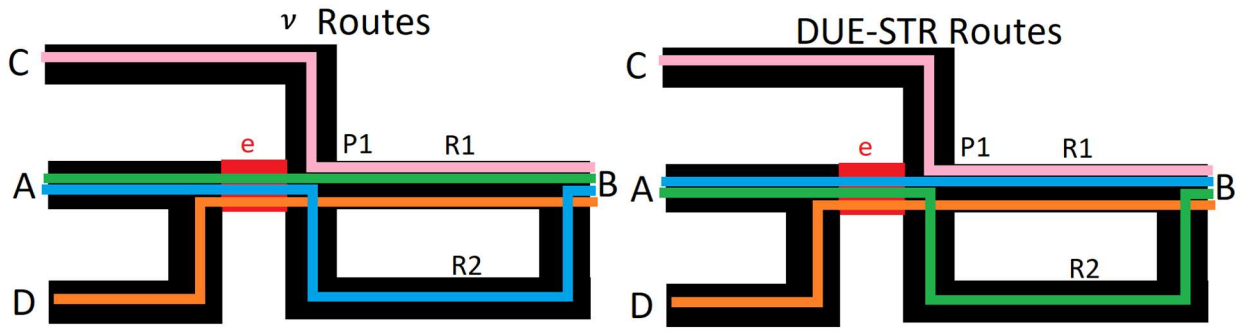

Figure 4: An Example of a Reroute on the Green Vehicle.

We consider the sample case shown in Figure 4. In this example, there are 4 vehicles, each following a route represented by a different color. A, C, and D are various start locations with two sub-routes of interest, $R1$ and $R2$, to the shared destination B. A shared edge e, marked in red, is a small part of the road right before the intersection $P1$. In the base routes in Algorithm 1, all vehicles are routed along the shortest path, in this case $R1$ to B. Figure 4 shows the ν routes, where the vehicle following the blue line is rerouted from $R1$ to $R2$ by duaiterate. Next, to accomplish the edge swap described in Algorithm 1, DUE-STR will enumerate edges in the green route where the green route is treated as the first encountered Vi, which is to say it has the earliest deadline of all the other vehicles in our example. The vehicle in the blue route will be found as a Vj since it shares the same edge $e \in E$ and the same destination with Vi. If the KNN time prediction agent indicates that the expected travel time of Vj is less than Vi, we mark a potential swap on this edge e.

After enumerating all feasible Vj's, like the one in the blue route, we choose the best Vj with the least expected travel time and swap the edges after e. Suppose the vehicle in the blue route is the chosen Vj, the green and blue routes after e will be swapped, as shown in the DUE-STR Routes graph in Figure 4.

For instances where multiple shared edges are present along the route of a vehicle, we repeat the same as above for each shared edge and return the best Vj from each of the relevant shared edge. We are left with a list of multiple Vjs with uniquely associated shared edges e to consider. We again choose the best Vj from this list and swap the Vi with the relevant e.

Note that though the pink route also includes intersection $P1$, it is not considered due to not sharing an edge e; this is to prevent the pink vehicle from making a U-turn. It is also possible for more than one e to exist, in which case the same process as above occurs, except that the best Vj from multiple shared edges are considered and subsequently chosen.

Chapter 3 Experiments and Results[4]

Our experimental setup consists of different "runs", where a "run" contains the information about the vehicles that will be moving through the red sub-network in Figure 1. This information includes between 300 and 500 randomly generated fully controlled vehicles with deadlines in the range of 500 to 1000 seconds, a randomly generated O-D matrix detailing the origin and destination of each vehicle. Our aim in these experiments is to simulate a sub-network with heavy traffic to show the viability of DUE-STR in a high traffic environment where a few choices in routing can have a dramatic impact on the performance of the network. The subnetwork in Figure 1 is generated via a SUMO tool [8] that extracts real-world map data from Open Street Map [27] and returns a fully functional SUMO network based on the selected map region. In this case, we select a map region close to the University of Houston where the traffic going in and out of the area can be expected to be relatively heavy. We test our algorithm with 10 randomly generated sets of runs and compare the performance of four generated route files. These route files are generated by DUAROUTER using the Dijkstra's algorithm as an experimental control, a 100-iteration execution of duaiterate, an execution of one-shot.py using an update time of 10, and finally our DUE-STR using the files duaiterate and DUAROUTER created as input. Each routing file is tested in a separate execution, simulating the network as all vehicles enter and travel to their destinations, where vehicles have the same set of O-D pairs and deadlines, allowing for easy

[4] Content in this chapter has been taken from the previously published work "Work In Progress: A Solution Based on Dynamic User Equilibrium Toward the Selfless Traffic Routing Model" By Thomas Carroll, Albert M. K. Cheng and Guangli Dai [30].

evaluation of multiple algorithms. The deadlines missed and average travel time for the base DUAROUTER generated routes, the ν routes from duaiterate, the DUE-STR final routes, and the one-shot generated trips are shown in Figure 5 and Figure 6, respectively. The graphs' x-axis shows the ID of an individual run followed by a dash and a number representing the number of vehicles generated in that particular run. The y-axis represents the number of deadline misses in the case of Figure 5 and the average travel time in Figure 6.

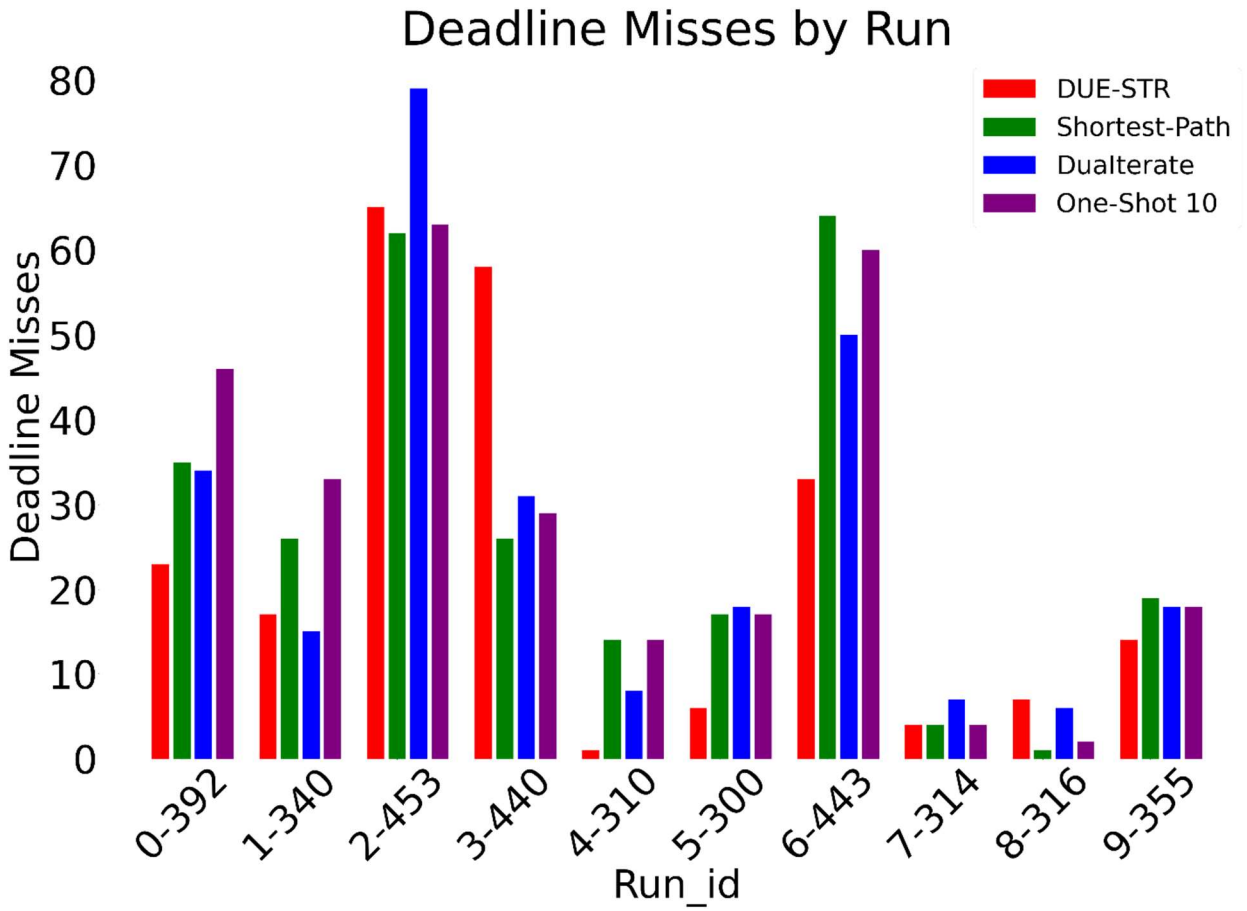

Figure 5: Performance of Different Traffic Routing Algorithms

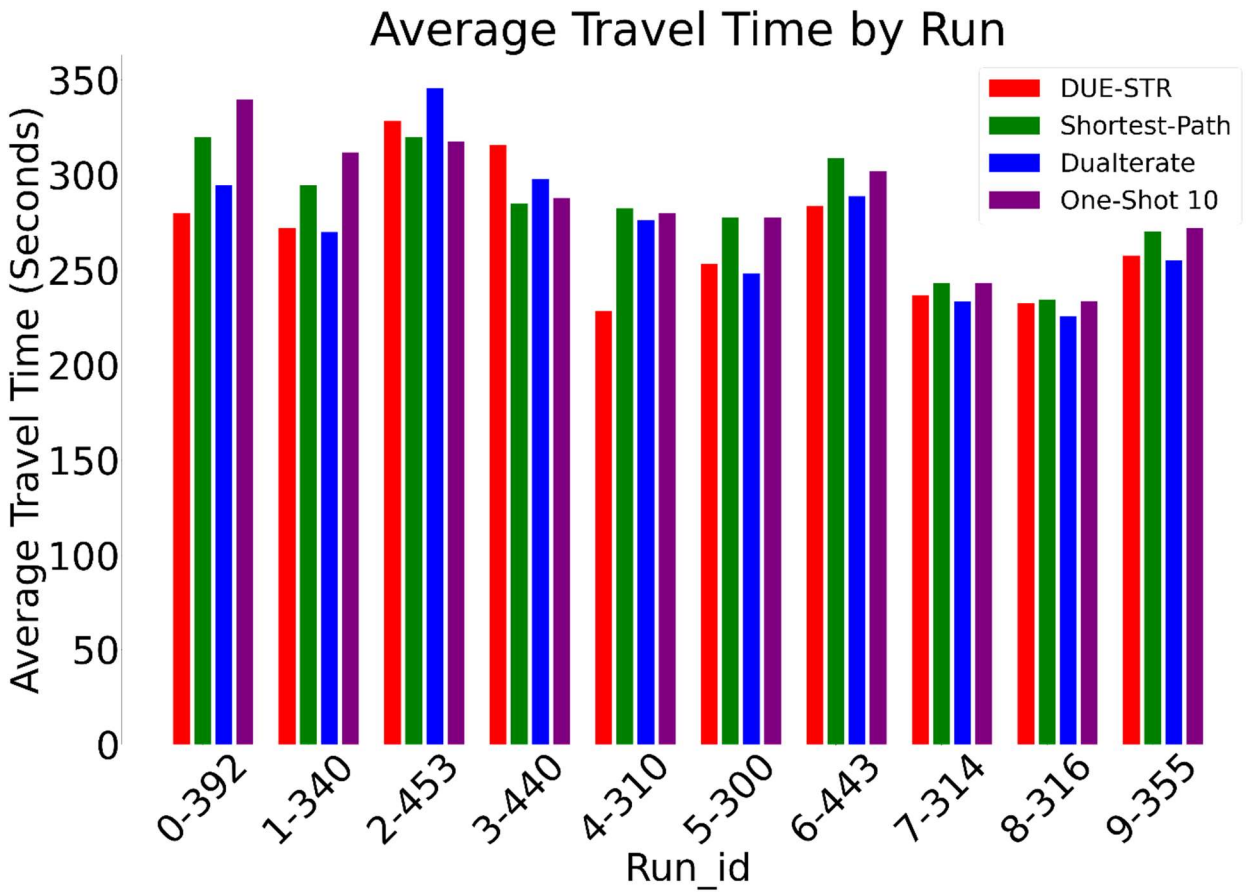

Figure 6: Average Travel Time by Run in Seconds.

3.1. Results of Experiments

We now compare and analyze the performance of the four routing algorithms and find mixed results. In terms of our primary objective deadline misses, DUE-STR outperforms or does as well as the other algorithms in runs 0, 4, 5, 6, 7, and 9, outperforms two algorithms in run 1, at least one algorithm in run 2, and does worse than all other algorithms in runs 3 and 8. With regards to our secondary object average travel time, we see that DUE-STR outperforms or does as well as the other algorithms in runs 0, 4, and 6, outperforms two algorithms in runs 1, 5, 7, 8, and 9 at least one algorithm in run 2, and does worse than all other algorithms in run 3. There are a few reasons

why DUE-STR might perform worse than the other algorithms: the lack of accuracy in the KNN time prediction agent, the strict adherence to DUE, and rerouting before vehicles travel (pre-run). We would like to go into further detail with regards to these issues and potential solutions. Starting from the top: while the KNN time prediction agent is a strong asset, it only predicts travel times before a run, which may cause some error due to not being able to consider the real-time conditions of the network, resulting in subsequent error. Furthermore, as shown earlier, the KNN regression travel time prediction agent is not 100% accurate. Moving on to the issue of strict Adherence to DUE: sometimes only considering the alternative routes created by duaiterates DUE approximation, disallows vehicles that need to swap in order to meet their deadline, to miss their deadline due to a lack of opportunity to reroute. We make a differentiation between this type of lack of opportunity and the lack of opportunity mentioned in Section 1.2 regarding no ability to reroute due to all the routes having the same travel time due to user equilibrium. Rather this lack of opportunity mentioned here occurs when no shared edges are found for a vehicle, thus a vehicle cannot reroute due to the lack of available alternative routes. To be more explicit, we can reference Figure 4 again; the vehicle being referred to here are vehicles like the pink vehicle. As shown in Figure 4, the pink vehicle has no valid shared edges e and thus no valid Vj to swap with. The evidence of these types of interactions is shown in Figure 7 by the red lines representing the vehicles that missed their deadlines.

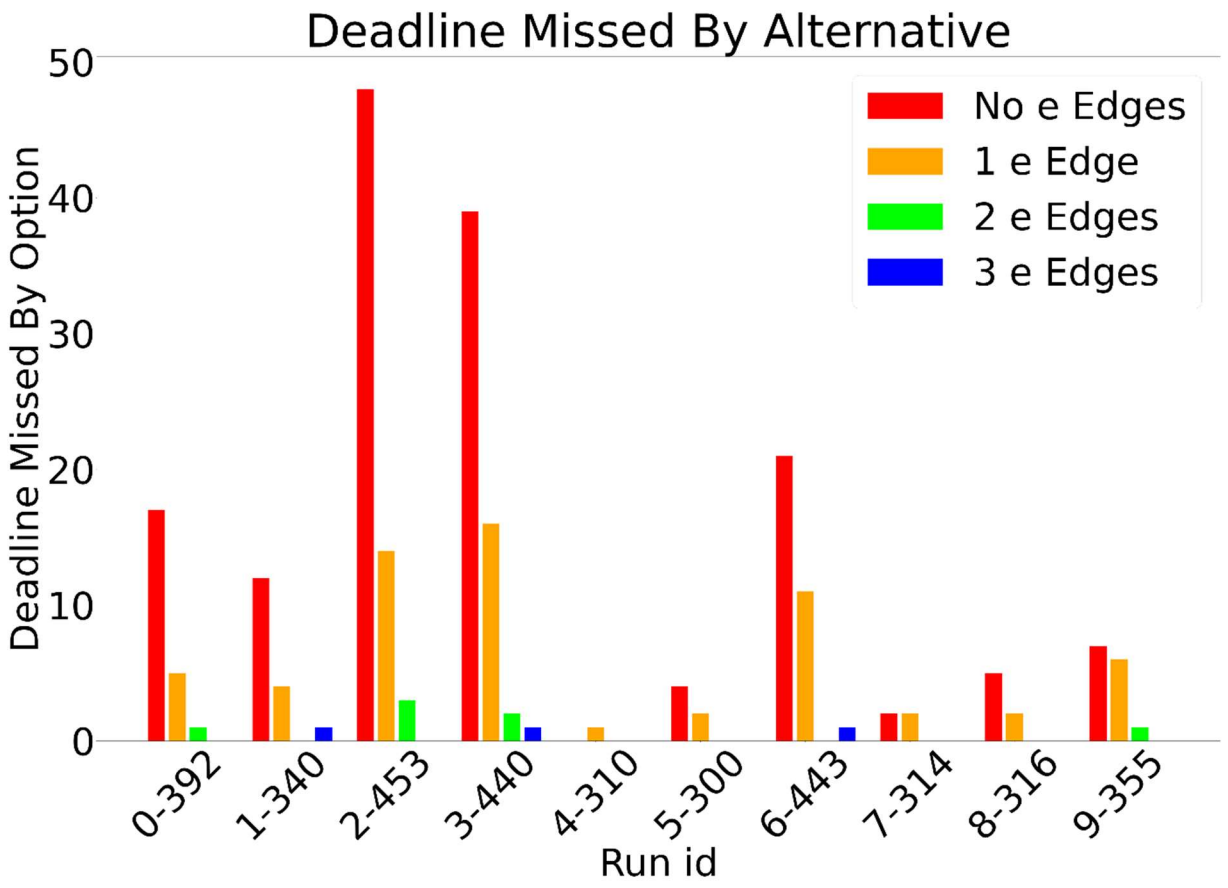

Figure 7: Deadline Miss by Options for DUE-STR

The vehicles with no "edge *e* found" (no shared edges found) aka vehicles like the pink vehicle in Figure 4, are the largest group of vehicles to miss their deadline in almost every run. Furthermore, the rerouting of vehicles can potentially affect the performance of the sub-network, moving it either towards or away from a good approximation of DUE, caused by duaiterate creating a non-optimal DUE. To elaborate, when two vehicles are swapped, we currently do not consider the distance between the two in the network. To be more vivid we can consider Figure 4, as shown we swap the routes of the blue and green vehicles, so let us consider that the green and orange vehicles start moving at time 50 and the blue vehicle starts at time 0. When we swap green and blue this can have an effect on the way green and blue interact with other vehicles. For example,

by swapping green and blue, green may end up cutting off orange at the P1 intersection resulting in a delay for the orange long enough to make it miss its deadline. We thus see that swapping vehicles like this can potentially have a negative impact on other vehicles as well. We can also consider the converse, going back to start times, in this example let blue and orange start at time 0 and green at time 50. If blue takes a right onto route R2 at intersection P1, it delays orange enough for orange to miss its deadline. Subsequently, swapping the routes of green and blue removes this delay for orange causing it to now meet its deadline. As shown in these examples, swapping can have a wide variety of effects on not only the vehicles being swapped but other vehicles throughout the network as well.

This is generally difficult to predict due to the temporal nature of vehicle routing and is the primary reason why duaiterate must run through so many different iterations to attempt to improve on the shortest path routes. Furthermore, as previously mentioned, not considering real-time conditions during swapping has a substantial role in this issue.

We would like to discuss potential solutions to these issues. The KNN time prediction agent could be made more accurate by considering real-time conditions of routes, giving us the potential to consider the number of vehicles ahead, current traffic light conditions, etc. Developing the KNN agent to consider these factors and more in real-time will allow us to experiment with different time periods in which reroutes occur, such as right as a vehicle enters the network (one-shot assignment), right before a vehicle reaches a shared edge, etc. This also extends to the issue of pre-run re-routes, whereby considering the state of the network at different times may help us make better re-routes. This goes back a bit to Chapter 1, where previous solutions [4] [5] considered the local state of the network, not considering more than a few edges ahead of current

vehicle positions. On the other hand, we aim to leverage the state of the overall sub-network to make more informed decisions. Finally, the issue of vehicles with no e edges missing deadlines could be solved by assigning some vehicles with longer deadlines and the same route to other longer routes to "move them out of the way" of the vehicles with closer deadlines. This extension of the "selfless" principle most likely requires another scheme other than DUE for alternative route generation.

While there is much room for improvement, on average DUE-STR combines the strengths of DUE and STR and was able to improve upon the existing algorithms in a single step, similar to one-shot assignment, without iterations unlike duaiterate which can take up to two minutes to get 100 iterations. This is an impressive result that shows the potential of the STR model on a larger network and with further improvement could be shown to be more practical in a real-time environment.

Chapter 4 Conclusion and Future Work [5]

In this thesis, we described and analyzed our implementation of STR with DUE in the form of DUE-STR on a sub-network, highlighting the promise of the STR model as well as current implementation weaknesses. Going forward, we look at potential challenges to overcome, such as ensuring that the travel time gained from selflessly rerouting is guaranteed to be minimized with a rigorous proof as well as, considering vehicles that do not follow the STR controller, or give inaccurate deadlines to gain a better route over other drivers, and how to minimize their effects. Working on some of the issues on the current implementation of DUE-STR as well as potentially looking at moving away from an explicit DUE in favor of other schemes are other areas of approach that seem promising. We plan to evaluate our strategies in a smart scaled city with model autonomous vehicles currently under construction.

4.1. Duckietown Scaled City

We plan to use the Duckietown scaled city [28], [29], consisting of a customizable scaled city complete with autonomous vehicles, traffic lights, and roads. The Duckiebots come complete with the capability to drive autonomously on said roads as well as receive route instructions from a centralized controller, much like the one described in this thesis. Currently, we are in the early stages, testing the capabilities of Duckiebots with one already assembled and operating using a

[5] Content in this chapter has been taken from the previously published work "Work In Progress: A Solution Based on Dynamic User Equilibrium Toward the Selfless Traffic Routing Model" By Thomas Carroll, Albert M. K. Cheng and Guangli Dai [30].

Nvidia Jetson board pictured in Figure 8. An example of a fully assembled scaled city is shown in Figure 9.

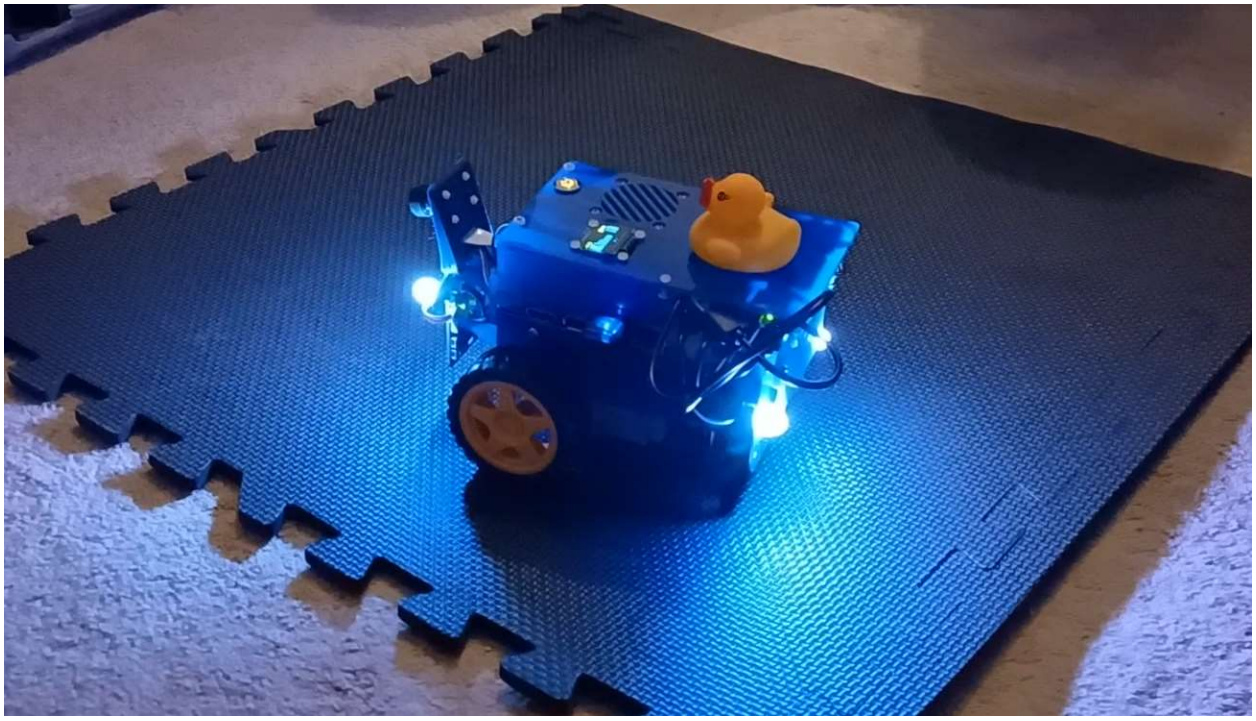

Figure 8: Our First Currently Assembled Duckiebot

Figure 9: A Duckietown Scale City as Shown on the Duckietown Website [29]

Our hope is that with the construction of the scaled city, we will be able to better study and understand routing algorithms at a much more physical level outside of a simulation. With these challenges and more ahead of us, we continue to aim for a more robust system to help reduce the number of people late for work.

Bibliography

[1] C. Burd, M. Burrows and B. Mckenzie, "Travel Time to Work in the United States: 2019," United States Census Bureau, 2021.

[2] Cambridge Systematics, Inc., Texas Transportation Institute, *Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestion Mitigation,* U.S. Department

of transportation, 2005.

[3] T. Yamashita, K. Izumi, K. Kurumatani and H. Nakashima, "Smooth Traffic Flow with a Cooperative Car Navigation System," in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, New York, NY, USA, 2005.

[4] Z. Cao, H. Guo, J. Zhang and U. Fastenrath, "Multiagent-Based Route Guidance for Increasing the Chance of Arrival on Time," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Phoenix, 2016.

[5] Z. Cao, S. Jiang, J. Zhang and H. Guo, "A Unified Framework for Vehicle Rerouting and Traffic Light Control to Reduce Traffic Congestion," *IEEE Trans. Intelligent Transportation Syst.,* vol. 18, pp. 1958-1973, 2017.

[6] Z. Zhou, B. De Schutter, S. Lin and Y. Xi, "Two-Level Hierarchical Model-Based Predictive Control for Large-Scale Urban Traffic Networks," *IEEE Transactions on Control Systems Technology,* vol. 25, pp. 496-508, 2017.

[7] 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," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019.

[8] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner and E. Wiessner, "Microscopic Traffic Simulation using SUMO," in *2018 21st Int. Conf. on Intelligent Transportation Syst. (ITSC)*, 2018.

[9] Waze, "Waze Live-map," [Online]. Available: https://www.waze.com/live-map/. [Accessed 2 April 2022].

[10] Google, "Google Maps," [Online]. Available: https://www.google.com/maps. [Accessed 2 April 2022].

[11] "Houston TranStar Traffic Map," Houston TranStar, [Online]. Available: http://traffic.houstontranstar.org/layers/. [Accessed 2 April 2022].

[12] Tesla, Inc., "Tesla Autopilot," [Online]. Available: https://www.tesla.com/autopilot. [Accessed 2 April 2022].

[13] Waymo LLC, "Waymo," https://waymo.com/, [Online]. Available: https://waymo.com/. [Accessed 10 April 2022].

[14] C. Wu, A. R. Kreidieh, K. Parvate, E. Vinitsky and A. Bayen, "Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control," in *IEEE Transactions on Robotics*, 2021.

[15] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[16] M. Patriksson, The Traffic Assignment Problem : Models & Methods, Mineola, New York: Courier Dover Publications, Inc., 2015.

[17] J. G. Wardrop, "Road Paper. Some Theoretical Aspects of Road Traffic Research.," *Proceedings of the Institution of Civil Engineers,* vol. 1, pp. 325-362, 1952.

[18] A. Pande, "2-route Traffic Assignment Problem," January 2018. [Online]. Available: https://www.youtube.com/watch?v=ZLnrY5jUxBY. [Accessed 2 April 2022].

[19] H. Yuan, G. Li, Z. Bao and L. Feng, "Effective Travel Time Estimation: When Historical

Trajectories over Road Networks Matter," in *SIGMOD/PODS '20: International Conference on Management of Data*, Portland OR USA, 2020.

[20] D. Wang, J. Zhang, W. Cao, J. Li and Z. Yu, "When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks," in *Proceedings of the AAAI Conference on Artificial Intelligence, 32*, New Orleans, Louisiana, USA, 2018.

[21] H. Wang, T. Xianfeng, Y.-H. Kuo, D. Kifer and L. Zhenhui, "A Simple Baseline for Travel Time Estimation Using Large-Scale Trip Data," *Association for Computing Machinery,* vol. 10, no. 2, 2019.

[22] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, S. Waller and J. Hicks, Dynamic Traffic Assignment: A Primer (Transportation Research Circular E-C153), United States of America: Transportation Research Board, 2011.

[23] D. Krajzewicz, M. Behrisch, J. Erdmann and Y.-P. Floetteroed, "Dynamic User Assignment," German Aerospace Center (DLR), February 2008. [Online]. Available: https://sumo.dlr.de/docs/Demand/Dynamic_User_Assignment.html#iterative_assignment_dynamic_user_equilibrium. [Accessed 25 April 2022].

[24] D. Krajzewicz, J. Erdmann, Y.-P. Floetteroed and M. Behrisch, "one-shot.py," German Aerospace Center (DLR), 10 March 2008. [Online]. Available: https://sumo.dlr.de/docs/Tools/Assign.html#one-shotpy. [Accessed 3 April 2022].

[25] J. W. Wedel, B. Schünemann and I. Radusch, "V2X-Based Traffic Congestion Recognition and Avoidance," in *2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, 2009.

[26] "sklearn.neighbors.KNEighborsRegressor," Scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html. [Accessed 01 04 2022].

[27] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing,* vol. 7, pp. 12-18, 2008.

[28] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard and A. Censi, "Duckietown: An Open, Inexpensive and Flexible Platform for Autonomy Education and Research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[29] "Duckietown – Learning Autonomy," Duckietown, [Online]. Available: https://www.duckietown.org/guides. [Accessed 11 April 2022].

[30] T. Carroll, A. M. K. Cheng and G. Dai, "Work In Progress: A Solution Based on Dynamic User Equilibrium Toward the Selfless Traffic Routing Model," in *IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Milano, 2022.