ATHABASCA UNIVERSITY

VEHICLE ROUTING IN A CONGESTED CITY USING A CENTRALIZED REAL-TIME TRAFFIC INFORMATION SYSTEM AND DIRECT EXPERIENCE

BY

ALEXANDER SOLTER

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SYSTEMS

SCHOOL OF COMPUTING AND INFORMATION SYSTEMS
FACULTY OF SCIENCE AND TECHNOLOGY

ATHABASCA UNIVERSITY
MARCH, 2020

Athabasca University
FACULTY OF GRADUATE STUDIES

The future of learning.

Approval of Thesis

The undersigned certify that they have read the thesis entitled

VEHICLE ROUTING IN A CONGESTED CITY USING A CENTRALIZED REAL-TIME TRAFFIC INFORMATION SYSTEM AND DIRECT EXPERIENCE

Submitted by

Alexander Solter

In partial fulfillment of the requirements for the degree of

Master of Science in Information Systems

The thesis examination committee certifies that the thesis
and the oral examination is approved

Supervisor:
Dr. Oscar Lin
Athabasca University

Committee Members:
Dr. Dunwei Wen
Athabasca University

Dr. Henry Leung
University of Calgary

External Examiner:
Dr. Rossitza Marinova
Concordia University

June 11, 2020

1 University Drive, Athabasca, AB, T9S 3A3 Canada
P: 780.675-6821 | Toll-free (CAN/U.S.) 1.800.788.9041 (ext 6821)
fgs@athabascau.ca | fgs.athabascau.ca | athabascau.ca

Dedication

To my family:

Sarah, Warren, and Sonja

Acknowledgements

I would like to thank Dr. Oscar Lin for working as my supervisor and providing guidance throughout my time at Athabasca University. His experience, knowledge, and patience were invaluable in the development and completion of this thesis.

I would like to thank Dr. Dunwei Wen for his guidance during the development of this thesis and for his detailed critique of the results.

I would also like to thank the other members of my committee, Dr. Henry Leung and Dr. Rossitza Marinova, for their feedback. The final thesis is much improved by their advice.

I would like to express my appreciation to Krystal Zahara in the Faculty of Graduate Studies and Linda Gray in the Faculty of Science and Technology for their assistance during my time at Athabasca.

Finally, I would like to thank my wife, Sarah, for her support during my studies and proofreading every line of my thesis.

Abstract

Navigation in a traffic congested city can prove to be a difficult task. Often a path that may appear to be the fastest option is much slower due to congestion. If we are able to predict the effects of congestion, it may be possible to develop a better route that allows us to reach our destination more quickly. This thesis studies the possibility of using a centralized real-time traffic information system containing travel time data collected from each road user. This data is made available to all users, such that they may be able to predict the effects of congestion when building a route.

This method is further enhanced by combining the traffic information system data with previous routing experiences. We test our method using a multi-agent simulation, demonstrating that this method produces a lower total route time for all vehicles than when using either a centralized traffic information system or direct experience alone.

Keywords: multi-agent systems, multiagent systems, reinforcement learning, traffic congestion, pathfinding

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1: Introduction

1.1 Research Motivation

When a driver attempts to navigate in a modern urban environment, they must overcome many obstacles to reach their destination. Poor weather, road construction, and accidents are just a few of these. However, while these problems may occur with varying degrees of frequency, traffic congestion is one which is encountered on a daily basis. Delays on the morning and evening commute to and from work are familiar to many who regularly travel in a city. Indeed, in particularly congested cities with high population densities, the problem of traffic congestion may be a constant condition on many roads.

Delays due to traffic congestion can be the cause of many problems. Drivers experience increased stress as delays may cause them to miss appointments or arrive late for work. Environmental damage is also a concern, as traffic delays require vehicles to operate for longer periods than may otherwise be necessary, resulting in increased pollution due to automotive exhaust. As well, economic damage can occur as worker productivity is reduced due to increased stress and work time lost in travel (Mandayam and Prabhakar, 2014).

When one considers the negative impacts of congestion it is clear that its reduction would be beneficial both to individuals and society. However, reducing congestion is not a simple task. Building additional roads to increase the volume of traffic that may be handled without congestion may not be possible in many locations, due to existing structures or budget constraints. Increased availability of mass transit can be helpful in

reducing the number of road users, but this can be expensive to operate and may not be feasible for users who must travel beyond a short distance to reach their destination.

If we consider that many commuters may opt to drive personal vehicles, either by preference or necessity, it is worthwhile to investigate how they may be better routed to reach their destinations while minimizing the negative effects of congestion.

1.1.1 Individually Optimized Routing to a Destination

The simplest approach to routing is to take the shortest path to one's destination. While this may seem ideal, the shortest path may not be the best option, as the roads selected may have a low speed limit and thus be inherently slow. The roads may also be congested at the time of travel, causing the route to be slower than anticipated.

A more sophisticated approach to routing would involve a consideration of the speed limit on each road taken. By factoring in how fast we can travel on each road in our path, we can calculate how long it would take to travel them. As such, a path that uses faster roads may result in a shorter route time than one that simply selects the shortest path.

These routing methods are examples of *user* or *individually* optimized routing (Bazzan and Chira, 2015). This type of routing focuses on finding the fastest, or optimal, route to the driver's destination. As such, there is little regard for the impact on existing traffic congestion beyond the necessity to limit its effects in delaying the driver.

However, using individually optimized routing can have a detrimental effect when all drivers attempt to use this method. Referred to as the *tragedy of the commons* (Hardin, 1968), this problem appears when all vehicles attempt to use the same roads at the same time. As the road is finite in the number of vehicles that may efficiently use it concurrently,

increasing the number of users will increase the amount of time it takes to travel upon the road.

1.1.2 System Optimization to Improve Routing to a Destination

An alternate method is to build routes that are *system* optimized. System optimization focuses on reducing the total amount of travel time for all vehicles using the road network (Bazzan and Chira, 2015), with the goal of reducing the impact of each vehicle on the congestion problem. As such, many road networks are designed to favour system optimization by operating high capacity and high-speed roads, with the goal of moving the largest number of vehicles possible through the system. System optimization can also be attempted through the use of traffic light systems that prioritize traffic on high capacity roads while also directing vehicles towards them (Bazzan and Chira, 2015).

Unfortunately, system optimization may result in negative effects for some drivers. When a route is built to take advantage of high capacity roads, the driver may not be using the best route to reach their destination – they may be required to take a longer path than otherwise necessary. While this can reduce the total travel time experienced by all drivers – the modified path reduces congestion on some other road – the individual driver does not see a benefit.

1.1.3 Hybrid Routing to a Destination

Ideally, we would like to optimize the route for the individual while also optimizing for the system. A hybrid method, combining both individual and system optimized routing attempts to use the best components of both methods to achieve this (Bazzan and Chira, 2015). By selecting roads that provide for a fast route for the driver but also work to avoid congestion, we can avoid building routes that cost the individual too much time while also

working to reduce the total congestion in the road network. The reduction in congestion may then be enough to offset the extra time that the individual driver must spend reaching their destination.

However, when routing in congestion, one must consider the following question: is it better for me to use a road that is short, but congested; or, is it better for me to take a road that is longer, but uncongested. The answer to this question is not simple when one does not know how badly congested the road is, as a short but congested road may be a better option than a longer, uncongested one if the delay due to congestion is small.

1.2 Research Objectives

The purpose of this thesis is to determine if we can develop routes that are better suited for areas with traffic congestion than existing methods.

When routing in traffic we face the following challenges:

1. Finding the best route while accounting for congestion requires the driver to search through a number of possible alternatives.
2. The specific amount of congestion that will be encountered on any given road segment is unknown to the driver.
3. Avoiding the tragedy of the commons to approach a system optimum state.

1.2.1 Finding the Best Route

Finding the fastest route to one's destination can be accomplished using a number of pathfinding algorithms, such as Dijkstra's Algorithm (Dijkstra, 1959) or the A*

algorithm (Hart et al., 1968). Given a map with sufficient detail of the road segment lengths and permissible road speeds, these algorithms can provide a route that will get the driver to their destination. However, given the unknown details of congestion and its effects on road speed, these algorithms are insufficient by themselves.

If we were to use a pathfinding algorithm to produce a number of possible routes that our driver could choose from, we may be able to find the fastest route. To do this we can use reinforcement learning try to learn which route will be the quickest. A Multi-Armed Bandit algorithm, such as Ɛ-Greedy (Thathachar and Sastry, 1985) or Upper Confidence Bound (UCB) (Auer et al., 2002) may be used to search for the best routing solution and more reliably select it in the future. However, in order to learn the fastest route to take, these methods must first explore the possible solutions, resulting in the driver using some potentially poor routes while trying to find the best one.

This provides us with the first objective for this thesis:

Objective 1: Determine the fastest route for the driver with the least exploration.

1.2.2 The Amount of Congestion is Unknown

The amount of congestion along a given route affects its speed. If a driver would like to select the fastest route with a limited number of routing tries, more detailed information about the congestion on a road segment may be of help.

Software such as Google Traffic (Google, 2017) and Waze (Waze, 2017) offer information about the traffic conditions on a road. Both operate by collecting data from

public traffic sensors and user provided travel times (via smartphone application) and use it to provide routes that account for delays due to congestion.

While a large amount of travel data can be collected from users, we must consider that the congestion problem will change over time. The number of road users may change, and the routes they select can result in roads becoming more or less congested over time. As such, we need a routing method that can adapt to changes in congestion and anticipate what these changes will do to congestion while building a route.

This provides us with our second objective for this thesis:

Objective 2: The routing method must adapt to changes in congestion over successive routing actions.

1.2.3 Avoiding the Tragedy of the Commons

As noted in 1.1.1, the tragedy of the commons occurs when each driver attempts to select the fastest route without regard for the effects of this selection on congestion. Ideally, we would like to avoid this issue and reach a system optimum state where the total travel time is at a minimum while also minimizing the travel time for each driver.

Stackelberg routing (Roughgarden, 2001), where a leader selected to pick an initial route which is then built upon by others, can be used to solve this issue. However, this method requires a central authority to select the leader and assign routes, which may or may not be followed by the other drivers if they consider them to be unfair.

This provides us with our final objective for this thesis:

Objective 3: The routing method must produce routes that are fair for each driver but also minimizes the total travel time for all drivers.

1.3 Research Questions

In addition to the three objectives noted in section 1.2, this research we will attempt to answer the following questions:

1) Will such a multi-agent system achieve user equilibrium with fewer routing episodes than either a centralized real-time traffic information system or direct agent experience?
2) Will re-routing while on route result in lower total route times than when no re-routing is used?
3) Will the weighting factor reach an equilibrium point at which the agent will no longer make adjustments between routing episodes?

1.4 Thesis Contribution and Significance of Research

This research makes multiple contributions towards solving the congestion problem in the urban environment. First, we show that the fastest route in a congested road network can be determined with less exploration than might otherwise be necessary. This is done by combining the data the driver acquires through experience driving a route with the data that is collected by all road users.

Second, we show that this method is adaptable to changes in the congestion problem by using reinforcement learning to teach each driver which travel data best matches the current congestion.

Finally, we demonstrate, through the use of a multi-agent simulation, that the drivers can reach an equilibrium point that approaches the system optimum with being directed through a control mechanism.

1.5 Definitions and Nomenclature Used

The following language is used in this thesis:

1) Agent – The term *agent* will be used to refer to the vehicle's navigation system. The agent develops the routes and decides which one the vehicle will take. The driver of the vehicle managed by the vehicle simulation and will always follow the route provided by the agent.
2) Road Segment – The term *road segment* is used to refer to a section of a road that lies between two intersections. As the routing agent can only make road selection decisions at an intersection, the road segment represents the smallest unit of a road that the agent can perceive.
3) Travel Time – The *travel time* refers to the time, in seconds, required for a vehicle to travel the length of a road segment. The travel time may be that of an individual vehicle, or may be an average of all vehicles for a specified range of times.
4) User Equilibrium (UE) – A state in which no agent perceives a possibly faster route than the one they are currently using. At this point no agent will select a different route than the one they used in their previous routing instance.
5) System Optimum (SO) – The road network is considered to be system optimum when the total amount of time required for all agents to complete their routes is minimized.

6) Centralized Real-Time Traffic Information System (TIS) – A system which collects travel time data from all agents and maintains it in a database. Each connected agent may request and receive travel time data for any road segment.

1.6 Thesis Organization

This thesis consists of seven chapters. Chapter 1 introduces routing in traffic congestion and presents the thesis objectives. Chapter 2 discusses the problem in more detail and reviews the literature on some of the other routing methods used. Chapter 3 discusses the theoretical framework. Chapter 4 covers the design of the solution and the method of experimentation. Chapter 5 presents our experimental results. Chapter 6 discusses the results of this research and future work. Finally, chapter 7 presents the conclusions reached.

Chapter 2: Review of the Literature

2.1 Routing in a City

In this chapter we review the various components involved in finding an efficient route to one's destination in a city. We start with some of the previous and current methods used to build a better path.

2.2 Traffic Planning

Before delving further into the details of building an effective routing solution that accounts for the effects of urban traffic, we must consider the composition of the traffic problem. This consists of two components – congestion and bottlenecking (Kutz, 2011). Congestion is the volume of traffic on a road while bottlenecking is a reduction in road capacity. A road may be considered to be congested when the volume of vehicles per section of roadway exceeds its capacity to handle them efficiently – beyond this volume the average speed of each vehicle is reduced. Bottlenecking can be caused when a road's capacity is reduced by either an event – such as construction or an accident closing a lane – or by slow moving vehicles – such as a bus or truck, or a car attempting a left turn at an intersection. This thesis is concerned with the alleviating the effects of the former issue, and as such we will not discuss bottlenecking further.

For city and traffic planners, the issue of traffic congestion is most often viewed as a network design and management problem. To optimize the road network, they use what is commonly referred to as an Intelligent Transportation System (ITS) (Desjardins, et al., 2009; Bazzan, 2009), and focus on using traffic information to make better use of existing

infrastructure. These methods include radio broadcasts and roadside messages to pass information to drivers, traffic cameras, websites, and variable timed traffic lights (Bazzan, 2009).

Centralized traffic management systems, such as the Sydney Coordinated Adaptive Traffic System (SCATS) (Wang et al., 2016) can attempt to improve traffic flow by further managing traffic lights, such that busier roads are given priority to increase the speed of vehicles travelling along the road when there is high traffic volume. While this technology can greatly improve traffic flow, it is complex and costly to implement. As well, changing traffic light timings to favour certain roads will encourage drivers to prefer using them, resulting in further congestion.

Recent advances in communications technology have also enabled the use of more advanced techniques. Vehicle-to-roadside (V2R) communications technologies allow for greater organization of the network by transmitting information about destination and vehicle status to local traffic management systems (Hong and Cheng, 2016). Intelligent intersections can sense the volume and direction of traffic passing through them and time signal changes to improve flow while also providing information to neighbouring intersections about the volume of traffic headed in their directions (Desjardins et al., 2009; Hong and Cheng, 2016; de Oliveira and Bazzan, 2009).

2.3 Wardrop's Principles

While ITS technologies can increase the overall capacity of the road network and generally work to improve the transit times for users, they do not focus on providing an efficient routing solution for the individual vehicle. This task falls to routing methodology.

Many routing methods attempt to solve this problem by focusing on reaching either a User Equilibrium (UE) or System Optimum (SO) state. These states are defined as two principles of route choice by Wardrop (1952), which can be summarized as:

1) User equilibrium, a state in which the travel time of routes used by all vehicles is equal to or less than any alternative route that could be selected by drivers.
2) System optimum, a state in which the total travel time for all drivers is minimized.

Each of these principles may be understood intuitively as the consequences of either self-interested or altruistic agents (Levy et al., 2017). When self-interested drivers build routes selfishly, that is, attempt to find the fastest route, regardless of the effect that selection has on other drivers, the road network will eventually reach UE. When drivers behave altruistically, they select routes that are perhaps slower, but reduce congestion on a given road, so a SO state may be reached.

However, to understand Wardrop's principles more thoroughly, we must look at the problem of routing in traffic congestion through the use of game theory.

2.4 Routing as Congestion Games

Game theory is the mathematical study of the strategies used when playing games. It's important to note that in our case, games are considered to be any activity where there are multiple players, each of which is attempting to determine the best strategy to use to achieve the largest reward possible (Shoham and Leyton-Brown, 2009). A strategy is the

method that is used to determine which action, or set of actions, the player will take to accomplish this.

While game theory is a broad subject, we are concerned with its use in modelling the problem of traffic congestion in a city. To do this, the problem can be represented as a *congestion game* (Tumer and Proper, 2013; Shoham and Leyton-Brown, 2009). Congestion games study the problem of maximizing an individual player or agent's reward where there are limited resources available. Each player in the game is attempting to own as many resources as possible, while also paying the lowest cost for them, thus maximizing their reward. The focus of the game is to determine which strategy will achieve this and at what point an equilibrium is reached. The game reaches equilibrium when no player perceives a possibility of improving their reward by changing strategy (Shoham and Leyton-Brown, 2009).

The game theory analysis of routing in traffic congestion views the problem as a *nonatomic congestion game* consisting of a large number of players attempting to use a set of roads (Shoham and Leyton-Brown, 2009). As the number of players is very large, although not infinite, the decisions of any individual player on the congestion encountered is very small – essentially, the decision of any one player to use a particular road at a particular time will not make a noticeable difference in how fast a vehicle may travel down that road.

A common example of this class of game is the *El Farol Bar Problem* (Shoham and Leyton-Brown, 2009). In this problem, each player must decide whether to visit the bar or stay at home. If more than 60% of players visit the bar, the player would have more fun staying at home. If less than 60% of players visit the bar, the player has more fun

visiting the bar. As all players must make their decisions at the same time, each must use a strategy to decide whether most of the other players have decided to go to the bar, or stay at home. The player maximizes their reward if they make a decision that is the opposite of that selected by most of the other players.

2.5 The Mathematics of Nonatomic Congestion Games

Mathematically, we can represent the nonatomic congestion game as a tuple, (N, μ, R, A, ρ, c), with the following properties (Shoham and Leyton-Brown, 2009):

$\boldsymbol{N} = \{\boldsymbol{1}, \ldots, \boldsymbol{n}\}$ is a set of players of different types;

$\boldsymbol{\mu} = (\boldsymbol{\mu_1}, \ldots, \boldsymbol{\mu_n})$ represents the players. Where $\boldsymbol{i} \in \boldsymbol{N}$ there is a continuum of players represented by the interval $[\boldsymbol{0}, \boldsymbol{\mu_i}]$;

$\boldsymbol{R}$ is a set of $\boldsymbol{k}$ roads. Each road segment, $\boldsymbol{r}$ is $\boldsymbol{r} \in \boldsymbol{R}$;

$\boldsymbol{A} = \boldsymbol{A_1} \times \ldots \times \boldsymbol{A_n}$, where $\boldsymbol{A_i} \subseteq \boldsymbol{2^R}\{\emptyset\}$ is the set of actions. The action $\boldsymbol{a_i} \in \boldsymbol{A_i}$ is selected by all players of type $\boldsymbol{i}$. The action a_i represents a segment of the player's path $\boldsymbol{A_i}$;

$\boldsymbol{\rho} = (\boldsymbol{\rho_1}, \ldots, \boldsymbol{\rho_n})$, where for each $\boldsymbol{i} \in \boldsymbol{N}$, $\boldsymbol{\rho_i}: \boldsymbol{A_i} \times \boldsymbol{R} \mapsto \mathbb{R}_+$ denotes the amount of congestion contributed to a given road segment $\boldsymbol{r} \in \boldsymbol{R}$ by players of type $\boldsymbol{i}$ selecting an action $\boldsymbol{a_i} \in \boldsymbol{A_i}$;

$\boldsymbol{c} = (\boldsymbol{c_1}, \ldots, \boldsymbol{c_k})$, where $\boldsymbol{c_r}: \mathbb{R}_+ \mapsto \mathbb{R}$ is a cost function for road segment $\boldsymbol{r} \in \boldsymbol{R}$, and $\boldsymbol{c_r}$ is nonnegative, continuous and nondecreasing.

We begin by defining the *action distribution* for the game. An action is the move that each agent makes that puts them on a particular road segment at a particular time. The action distribution, $s \in S$, indicates the number of players that choose each action, and $s(a_i)$ is the element of *s* that corresponds to the set of players of type *i* who select action $a_i \in A_i$.

As the action distribution represents the actions of each player of type *i*, we can arrive at the following:

$$\sum_{a_i \in A_i} s(a_i) = u_i \tag{1}$$

This allows us to determine the amount of congestion, s_r, on a given road segment as a multiplication of the number of players selecting the segment by the amount of congestion each contributes by their actions:

$$s_r = \sum_{i \in N} \sum_{a_i \in A_i} \rho_i(a_i, r) s(a_i) \tag{2}$$

We note from the above formula that, although the effect of any one player on the congestion problem is very small, it is not 0. Thus, the actions of all players result in congestion that can be measured and does affect the flow of traffic.

While we can now formulate the amount of congestion on a road segment, we would still like to know the effect of this on our players. We can calculate the *cost function* due to congestion induced on the a given road segment by the players' actions as:

$$c_{a_i}(s) = \sum_{r \in a_i} \rho(a_i, r) c_r(s_r) \tag{3}$$

The *utility function* for the player may now be determined. The utility function helps the player determine which of a number of possible actions is of the most ber (4) them if selected (Shoham and Leyton-Brown, 2009). In this case, the utility function can be expressed as the cost of selecting a road segment, where the road segment with the least congestion gives the highest utility:

$$u_i(a_i, s) = -c_{a_i}(s)$$

Finally, the *social cost* to all players choosing an action may be found by multiplying the action distribution for all players of type *i* by the cost function due to those players selecting a given road segment:

$$C(s) = \sum_{i \in N} \sum_{a_i \in A_i} s(a_i) c_{a_i}(s) \quad (5)$$

An analysis of the social cost formula reveals that the system optimum is achieved when the players select actions that sum to the lowest social cost. However, because the social cost is the sum of the cost incurred by all players, we can achieve this when some players achieve a lower cost by selecting a given action while other players select an action that is more costly to them individually. Under these conditions, the player with the more costly action will select a less costly action to increase their utility. As such, by selecting an action that is less costly, they increase the congestion on the road segment, thus increasing the cost for all players selecting the same action. Since the cost increases, this increases the social cost to all players, although at this point an equilibrium may have been reached.

Given the effects of selfish selection of routes, the user equilibrium can, at best, reach a social optimum. However, often this may not be the case, as the effects of selfish actions work to increase the impact of congestion for all (Levy et al., 2017).

2.6 Approaches to Routing

There have been a number of different approaches that have been applied to the routing problem. As noted in chapter 1, these approaches can be separated into attempting to solve for either of Wardrop's Principles or both, with varying levels of sophistication.

2.6.1 Dijkstra's Algorithm and A* Routing

Perhaps the most direct method of routing is to select a path that combines the fastest allowable road speeds with the shortest possible distance. Provided a map of the city in which we would like to navigate, we can use a pathfinding algorithm such as Dijkstra's Algorithm or A*.

Treating the map as a graph, where each intersection is a node and each road segment is an edge, Dijkstra's Algorithm (Dijkstra, 1959) performs a best-first search to find the shortest path. Starting at an origin point, the algorithm calculates the travel time to each unvisited node that can be reached directly and selects the fastest one. In successive iterations, the algorithm selects the closest node to the origin that has not yet been visited. Given sufficient iterations, this algorithm will provide the user with the fastest route to their destination.

The A* algorithm (Hart et al., 1968) is a modified version of Dijkstra's Algorithm that attempts to improve upon the speed at which an optimal path is determined. Rather

than selecting the edge that arrives at a neighbouring node the fastest, the algorithm also considers whether the edge will bring the driver closer to their destination.

Using a heuristic value, such as the Euclidian distance from the edge endpoint node to the driver's destination, the algorithm selects the next edge in its path as the shortest sum of distance travelled so far and the heuristic distance to the destination. This method allows the algorithm to focus on trying paths that take the driver closer to their destination, and thus typically arrives at an optimal solution more quickly than Dijkstra's algorithm.

2.6.2 Route Information Sharing

Route Information Sharing (RIS) (Yamashita et al., 2005) represents one method to reach a system optimum. By sharing information on the routes chosen by drivers, this method seeks to enable them to avoid congestion. The method is implemented as follows:

1) Each driver builds a shortest route to the destination. The information is then transmitted to a server.
2) The server collects route information from all drivers and assigns a weight value to each road segment for each driver.
3) The weight of each driver on each segment is summed, producing a total weigh for each segment.
4) The total weight is used to calculate the expected traffic on each segment and, thus, the expected travel time.
5) The expected travel time is transmitted to each driver for their prospective routes. The drivers may then revise their routes based on this information.
6) Steps 2 through 5 are repeated until each driver has selected the route they will use.

7) The drivers travel their routes. At each intersection, the driver transmits their location and updates their information. If the vehicle encounters congestion the route is revised.
8) Once at their destination, the vehicle is removed from the problem.

Route Information Sharing has been shown to provide an improvement in the average travel time for drivers using this method over the average times for drivers routing using a shortest distance method. This difference was found to increase as the percentage of vehicles using RIS increased in the road network.

However, while an improvement was found, the number of cycles of route, transmit, receive, and re-route that must be accomplished before a final set of routes is reached can be large. As well, it has been noted that, in large cities where there may be millions of vehicles, such a system may be impractical, as a typical communications system, such as a cellular network, may not capable of handling the number of connections required.

2.6.3 Sampling and Weighting Algorithm

While a system optimal method of routing results in faster overall route times, preventing the system from reverting to a user equilibrium state that is less efficient is difficult. Route Information Sharing approaches this, but does so at the price of potentially delayed routing results and a large communications infrastructure cost.

If drivers are selfish in their behaviours – that is, always attempting to achieve the fastest route for themselves, regardless of the cost to others – perhaps a better approach

would be to achieve a user equilibrium state that is as close as possible to the social optimum. Levy, Klein and Ben-Elia (2017), and Levy and Ben-Elia (2016) investigate the possibility of system optimum being an emergent property of a multi-agent system.

In their research, a group of agents, representing drivers, are given a choice between two routes of equal length. As both routes are equal, the only factor that affects the route completion time is the number of agents that select the route. Each agent has the goal of reaching the end of the route in the shortest time possible and must decide which route is the best choice to achieve this.

The authors solve this problem applying the agents' previous routing experience on each route. A routing simulation was built for the agents with both available routes. As successive simulations are run, the agents acquired more information as to the amount of time required to complete each route, which is used to inform the agent's route selection in the next simulation.

To determine which route is likely to be the least congested, each agent uses the Sampling And Weighting (SAW) formula (Levy et al., 2017; Levy and Ben-Elia 2016), which is an adaptation of the formula used by Erev et al. (2010):

$$EST_{j,k} = w * \frac{\sum_{i=0}^{k} RouteTime_j(i)}{DaysOnRoute} + (1 - w) * \frac{\sum_{i=k-\delta}^{k} RouteTime_j(i)}{\delta} \quad (6)$$

Where *j* is the potential route, *k* is the index of simulations where the agent selected route *j*, δ is the number of recent simulations used, and *w* is a unitary weight determining the agent's long-term memory. A weight value of 1 results in the agent only using long-term travel data, while a weight of 0 causes the agent to use only recent travel data. The

weight value can take any value between and including 0 and 1, giving the agent the ability to consider both long-term and recent travel data as well.

The experiment was initially performed using a group of agents selecting routes selfishly – the agents use a utility function that is maximized when their route is the shortest possible. It was found that the agents converged on a user equilibrium most quickly when $w = 0$ and long-term memory was not employed.

After 15000 runs, the agents were modified to become altruistic, where each agent's utility function is maximized by the total route time of all agents being minimized. The researchers found that the agents were able to achieve a SO state without a control mechanism and that this occurred most quickly when $w = 0$.

While this work shows that it is possible to reach a user equilibrium and a social optimum state using a deterministic algorithm, there are some practical limitations. First, the agents required almost 2000 runs to reach a user equilibrium, and reaching a social optimum required approximately 500 runs. If such a system were attempted for a group of drivers, it would require a large number of tries before they saw a significant improvement in their route times.

Secondly, the experiments gave the agents the option of two routes, rather than the hundreds that may be possible in an urban road network. Given the larger number of options, a weight value that accounts for long-term memory may result in a better result that was found when only two routes were allowed.

2.7 Summary

While the methods discussed above may assist in improving travel in a congested city, there are costs or limitations to each. In chapter 3 an alternative routing method is introduced that may better manage these issues.

Chapter 3: The Theoretical Framework

3.1 Multi-Agent Routing Method

In Chapter 1 we noted the following three objectives for this thesis:

1) To determine the fastest route for the driver with the least exploration.
2) To find a routing method to adapt to changes in congestion over successive routing actions.
3) To find a routing method to produce routes that are fair for each driver but also minimize the total travel time for all drivers.

We present a multi-agent routing methodology that addresses each of these objectives when routing in an urban environment. To develop their route, each agent uses the following steps, which we expand upon later in this chapter.

1) **Build a list of routes -** The agent builds a list of potential routes to its destination. The list is built using a modified version of the A* algorithm, which returns the fastest routes possible based on the map data available to the agent.
2) **Estimate road segment start times -** For all potential routes in the list, the agent estimates the time at which they will start each road segment.
3) **Request travel time data from TIS -** The agent requests travel time data from the centralized real-time traffic information system (TIS) for each road segment in each route. By using TIS, the agent avoids the need to try each route to learn

the amount of congestion first hand, thus minimizing the exploration required to estimate its effects.

4) **TIS returns travel times -** The TIS returns two average travel times for each road segment requested, adjusted for the time at which the agent estimates it will reach the segment. First is the long-term average, which is an average of all vehicle travel times on the given road segment for all available routing episodes. Second, the short-term average, comprised of the average travel time for all vehicles in the 5 most recent routing episodes.
5) **Estimate potential route times -** The agent applies the averages to each potential route using the SAW formula. The agent selects the route with the fastest estimated time. As the estimates are developed using previous travel time data, our routing method is able to adapt to changes in congestion – the effects of any changes will be reflected in the travel times used.
6) **Apply previous routing experience -** The agent compares the selected route to a list of routes it has travelled previously. If the route is the same as the route used by the agent in the previous routing episode, the agent will select the route. If it is different, the agent searches the list to determine if it has used the route before. If not, the agent will use the route. If the agent has used the route previously, it will only select the new route if it was significantly faster than the route used in the previous routing episode.
7) **Drive selected route -** The agent travels its route. As the agent travels, it transmits the time required to complete each road segment to the TIS. If re-routing is enabled for the agent, it compares the time on route to its estimated

route time to that point. If the route time is significantly higher than the estimated time it will re-route.

8) **Apply learning algorithm -** After completing its route, the agent reviews the route times for each of its potential routes by again querying the TIS to apply the most recent travel time data to each one. If the fastest route was different than the one the agent selected, they adjust their selection algorithm to better reflect the effects of congestion on its routes. As each agent acts to improve its route selection, the amount of time to complete its routes are reduced and works to minimize the total travel time for all drivers, while also providing a fair route for the agent.

Figure 1 presents a flowchart depicting the agent's routing actions. The algorithm *Agent Routing Process* provides an overview of the agent routing steps and calls the algorithms presented in the following sections.

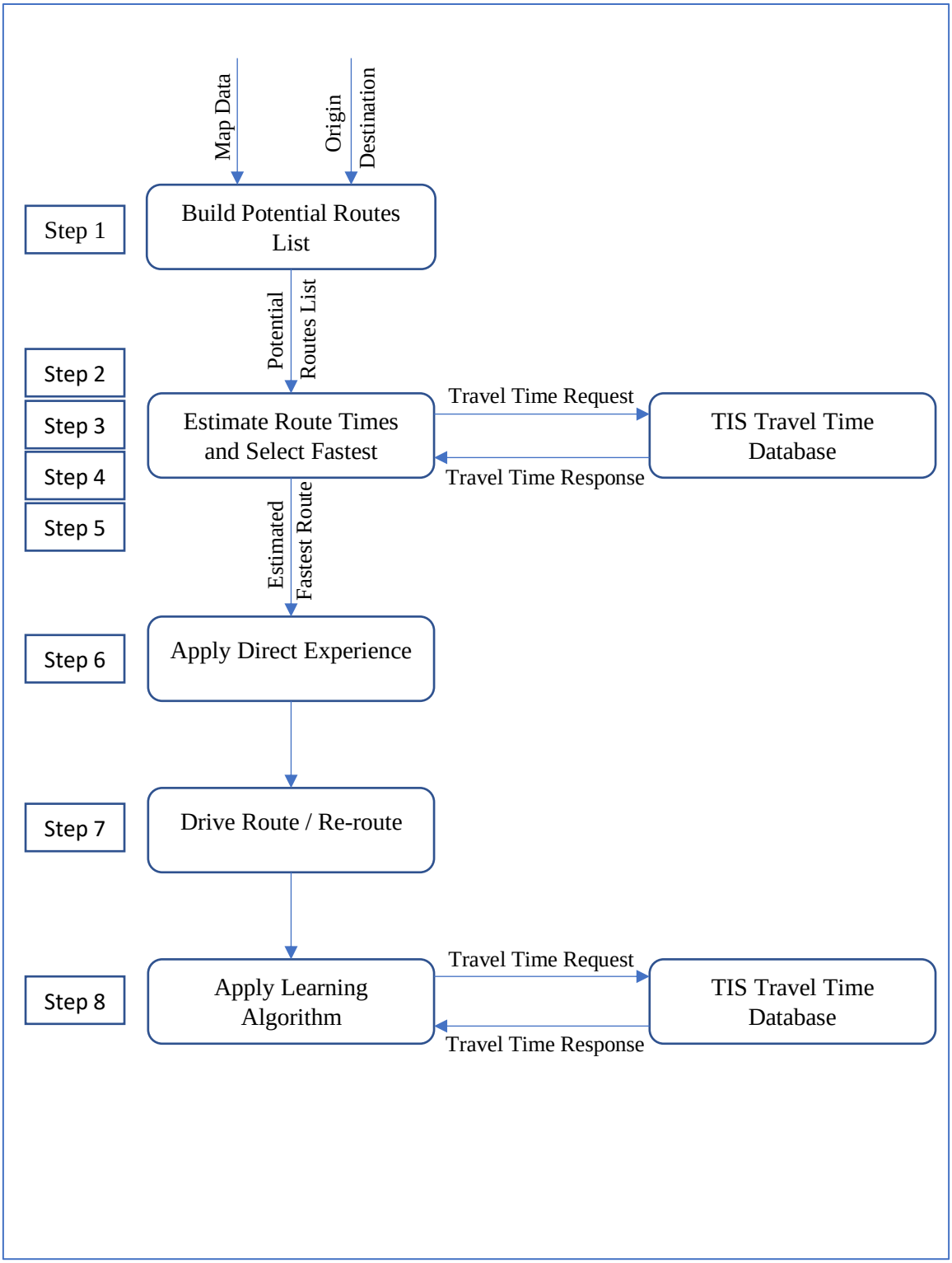

Figure 1. Agent Routing Process

Algorithm Agent Routing Process

Input: Map data, agent origin, agent destination.

Output: Updated SAW weight.

```
// Get the list of potential routes (Step 1).
routeList = call ModA*(map data, origin, destination, number of routes requested);

// Estimate fastest route (Steps 2, 3, 4, 5).
routeList[fastestRoute] = call RouteSelector(routeList, SAW Weight);

// Find route with fastest estimate (Step 6).
fastestRoute = call RouteMemory(routeList[fastestRoute], last completed route,
                                exploration factor);

// Agent drives route (Step 7).

// If re-routing is enabled, determine a new route if current route is too slow at
// end of each road segment (Step 7).
fastestRoute = call ReRouting(fastestRoute, actual route time, performance factor);

// Apply the learning algorithm to find new SAW weight (Step 8).
newSAWWeight = call LearnSAWWeight(routeList, actual route time,
                                previous SAW weight, weight change factor);
```

Algorithm 1. Agent Routing Process

3.2 Building the Potential Routes List (Step 1)

Before selecting the fastest path to its destination, the agent must first develop a list of viable routes. In many cities the number of possible routes can be quite large, given a multitude of roads to choose from. However, many routes are not good options due to a combination of allowable road speed and distance.

To accomplish this task, the agent uses a modified version of the A* algorithm (Hart et al., 1968). While the standard A* algorithm finds the fastest route to a destination, based on the path length and allowable road speed, this only provides one route. Our modified version, *Mod A**, will continue to build routes until a pre-determined number of routes has been reached, providing the agent with a list of routes to choose from.

Algorithm ModA*

Input: Map data, agent origin, agent destination, number of routes requested.

Output: A list of the fastest routes by distance and road speed.

```
mapData = Map data;
originNode = The origin node location on the map;
destinationNode = The destination node location on the map;
edgeList = null;                                    // The list of edges to search.
numberOfRoutes = The number of routes requested;
currentEdgeIndex = 0;                               // The index of the edge being
                                                    // searched by the algorithm.
destinationFound = false;                           // Boolean indicating the
                                                    // destination node was found.
routeList = null;                                   // The list of potential routes.
routesFound = 0;                                    // Counts the number of routes.

// Find every edge from the origin node.
for (each edge from originNode) {
   edge.available = true;
   edge.g = mapData.edgeLength / mapData.roadSpeed;
   edge.h = destinationNode.GPS – edge.endNode.GPS;
   edgeList.add(edge);
}
```

Algorithm 2. ModA*

Algorithm ModA* ***continued***

```
// Loop through the list of available edges until the requested number of routes
// are found.
While (routesFound < numberOfRoutes) {
   // Find the available edge with the lowest f(x) = g(x) + h(x).
   shortestEdge = 1000000;

   for (each edge in edgeList) {
      if (edge.available) {
         if (edge.endNode == destinationNode) {
            destinationFound = true;
            currentEdgeIndex = index of edge;
         }
         edge.f = edge.g + edge.h;
         if (edge.f < shortestEdge && !destinationFound) {
            shortestEdge = edge.f;
            currentEdgeIndex = index of edge;
         }
      }
   }

   // We now have the available edge with the shortest f(x).
   edgeList[currentEdgeIndex].available = false;
```

Algorithm 2. ModA*

Algorithm ModA* ***continued***

```
    // If the destination wasn't found, retrieve all connected edges.
    if (!destinationFound) {
        for (each childEdge in edgeList[currentEdgeIndex]) {
            childEdge.available = true;
            childEdge.g = mapData.edgeLength / mapData.roadSpeed
                                + edgeList[currentEdgeIndex].g;
            childEdge.h = destinationGPS – childEdge.endNode.GPS;
            childEdge.parentEdge = currentEdgeIndex;
            edgeList.add(childEdge);
        }
    }

    // If the destination was found, add the route to the route list.
    if (destinationFound) {
        route = Route built by tracing back from edgeList[currentEdgeIndex];
        routeList.add(route);
        routesFound = routesFound + 1;
        destinationFound = false;
    }
}

// Return the list of potential routes.
return routeList;
```

Algorithm 2. ModA*

3.3 Estimate Route Times (Steps 2, 3, 4, 5)

Given a list of potential routes, the agent must now determine which one to use while also accounting for the effects of congestion. As shown by Levy et al. (2017), determining the impact congestion has on a given route can require a large number of tries before we can be sure we have found the fastest one.

To reduce the amount of searching required to find the fastest route with congestion, we use a TIS containing travel time data collected from each vehicle. When a vehicle passes through an intersection, the amount of time required to traverse the road segment is transmitted to the database, along with the start and completion times on the segment. As the amount of information is small and only transmitted at the intersection, we can avoid the RIS communications capacity limitations noted by Yamashita et al. (2005).

The collected travel time data is used by each agent as a substitute for direct experience that would have been gathered through the exploration of different route options. This allows our routing method to be flexible as to the origin and destination of the agent – they need not have travelled to a destination previously to be able to select a route that will account for traffic congestion.

3.3.1 Requesting Travel Times (Step 2)

Although the agents have travel time data available to them for any road segment they may wish to select, they are faced with a problem. The amount of travel time data can be very large, as it may be collected from many vehicles over a long period of time, and the transfer of such a large volume of data would be impractical when a driver is waiting for their route.

We manage this issue by limiting the amount of data that is required by the agent to make its decision. As the agent has a list of potential routes to choose from, they only require the travel time data relevant to each route. The agent further reduces the required data by estimating the time at which its vehicle will reach a given road segment, thus only requesting the travel time data for a limited time frame.

The specification a time frame further aids the agent by accounting for changes in congestion that may occur while travelling a route. For instance, a road segment may not have much congestion at the time when the agent begins its route, but a number of employers located along the route may begin their day as the agent's vehicle travels, producing congestion that didn't exist earlier. Having data that indicates that a given road segment will become more congested by the time its vehicle reaches it helps the agent determine if selecting a route with that segment is good option.

3.3.2 Retrieving Travel Times (Steps 3, 4)

The traffic information system, upon receiving a request, must retrieve the data and format it to send to the requesting agent. However, there are variations in the data that must first be managed. The travel time data will vary over days and months – a road segment may have little congestion on a Sunday afternoon, but be very congested on Monday morning when a large number of drivers are travelling to work. Additionally, seasonal changes may be expected, such as higher congestion on road segments near shopping malls in the month of December.

The database accounts for these changes by only returning travel times for the same day of the week as the day being routed. Thus, if the current date is a Monday, the database will only retrieve data from previous Mondays.

Another issue that may arise is travel time variation due to unforeseen conditions. A driver may encounter a slow-moving vehicle or bottlenecking due to construction, which may temporarily slow traffic. While these incidents will show in the data as an increased amount of time required to traverse a road segment, they aren't representative of the day-to-day congestion that the driver may be expected to encounter.

The database manages this issue by averaging the travel time data returned. Upon retrieval, the database will construct two values for the road segment – the long-term and short-term average. The long-term average consists of the average time required to complete the segment over all dates available, while the short-term average is that over the most recent days. The number of recent days used is configured to be consistent for all data requests.

Finally, when data is returned to the agent, it consists of a long-term and short-term average for each road segment requested, adjusted for the estimated time of arrival at the segment.

3.3.3 Building the Route Estimate (Step 5)

Given the list of potential routes and both long and short-term travel time averages for each road segment, the agent must now make a routing decision. To do this we use the Sampling And Weighting (SAW) formula from Levy et al. (2017):

$$EST_{j,k} = w * \frac{\sum_{i=0}^{k} RouteTime_j(i)}{DaysOnRoute} + (1-w) * \frac{\sum_{i-\delta}^{k} RouteTime_j(i)}{\delta} \qquad (6)$$

Rewritten, we use the formula as (7):

$$EST_{route} = w * \sum_{i=0}^{r} Long-Term\ Avg(i) + (1-w) * \sum_{i=0}^{r} Short-Term\ Avg(i)$$

Where EST_{route} is the estimated total route time, r is the number of segments on the route, and w is a weighting factor. The weighting factor allows the agent to choose which set of averages will have more value in the routing decision – long-term or short-term.

The SAW formula (6) was selected for the route estimation task as it allows the agents to easily apply the large amount of travel time data available to them while also accounting for changes to the congestion problem that will occur over time. The weighting factor is not a fixed value, but rather is changed by the agent over time as the congestion problem changes. The selection of the weighting factor is explained further in section 6 of this chapter.

Once the agent has estimated the travel time for each potential route it selects the one with the lowest estimate.

Steps 2, 3, 4, and 5 are implemented by the *Route Selector* algorithm.

```
Algorithm RouteSelector
    Input: List of potential routes, SAW weight.
    Output: A route with the fastest estimated time.

    routeList = List of routes from ModA*;        // Algorithm 2: ModA*
    weight = The SAW weight;

    // Calculate the estimated time for each route.
    for (each route in routeList) {
       for (each roadSegment in route) {
          // Calculate estimated start time at road segment.
          estStartTime = routeSegment.EstEndTime – routeSegment.EstTravelTime;
          // Request travel times from database for segment.
          roadSegment.LongTermTime = Get Long-term average by estStartTime;
          roadSegment.ShortTermTime = Get Short-term average by estStartTime;
          // Add the segment times to the route times.
          route.LongTermTime = route.LongTermTime + roadSegment.LongTermTime;
          route.ShortTermTime = route.ShortTermTime + roadSegment.ShortTermTime;
       }
       // Calculate SAW estimated route time.  Formula (7).
       route.SAWEst = weight * route.LongTermTime
                              + (1 – weight) * route.ShortTermTime;
    }
```

Algorithm 3. Route Selector

Algorithm RouteSelector ***continued***

```
// Select the route with the fastest estimate.
fastestSAW = 10000000;                  // Stores the fastest estimate.
fastestIndex = 0;                       // Stores the route list index of fastest.
for (each route in routeList) {
   if (route.SAWEst < fastestSAW) {
      fastestSAW = route.SAWEst;
      fastestIndex = routeList.CurrentIndex;
   }
}
// Return the estimated fastest route.
return routeList[fastestIndex];
```

Algorithm 3. Route Selector

3.4 Apply Direct Experience (Step 6)

Although the agent has made a decision as to the best route to use, we are now faced with a potential problem. In their research, Levy et al. (2017) required a large number of routing runs before the agents reached equilibrium. This is partly due to the agents having to guess the amount of congestion on a given road – an issue which we are addressing with the use of a traffic information system. However, another issue is the amount of route switching the agents perform while trying to settle on the fastest one. For our routing method to produce a good route for each driver, the agents must reach an

equilibrium where there is no incentive for them to switch routes in consecutive routing tries.

To reach equilibrium the agents must avoid selecting different routes that have slightly faster estimated times than the route they most recently completed. For instance, if an agent were to select a route that is estimated to be one second faster than its previous route, the agent may find that the new route is not as fast as expected, due to unanticipated delays, such as a slower moving vehicle. In the next routing instance, the agent would switch back to its first route, only to find the previous route may have been faster. This cycle may continue many times before the agent reaches settles on a route.

We manage this issue by including previous experience on the route as the final step of the routing process. If the agent has never used the estimated fastest route before, it will always select it, allowing it to explore an option that may well prove be the best available under current congestion. If the agent has used the route previously it then compares it to the route it has used most recently for the same origin and destination. If the route is the same it will continue to use it. If the route is different it must decide if selecting the new route will be faster than the last used route.

The agent will now retrieve the average completion time for the new route and the actual completion time for the most recently used route. This time is the agent's own experience on the route. If the new route is faster, the agent will select it, as both the estimated route time and previous experience indicate this is likely to be a good choice. If the new route's previous times are slower, the agent will compare the new route's estimated time with the previous route's average multiplied by an exploration factor. The exploration factor represents the agent's willingness to switch to a different route rather

than continue exploiting the one they have most recently used. If the new route's estimated time is faster, it will select it, otherwise the agent will stay with the previous route. The exploration factor static value used by the agent in all routing attempts.

The algorithm, *Route Memory*, implements step 6 and applies the agent's direct experience to the routing problem.

```
Algorithm RouteMemory
    Input: Estimated fastest route, last completed route, exploration factor.
    Output: Selected route.

    route = Estimated fastest route;
    prevRoute = Last completed route;
    explorationFactor = Exploration factor;

    // Check if the agent has used the route before.  The agent always uses a new route.
    if (route is not new) {
       // Compare the segment list of the new route and previous route.
       if (route.segmentList != prevRoute.segmentList) {
          // The routes are different.  Get the new route's average route time in the past.
          route.avgPastTime = The average of route time in past routing tries.
          // Compare the new route average time to the previous route time.
          if (route.avgPastTime > prevRoute.actualTime) {
             // The new route average time is higher than the previous route time.
             // Compare new route estimated time to previous route time with exploration
             // factor.
             if (route.SAWEst > (prevRoute.actualTime * explorationFactor)) {
                // The agent will use the previous route.  Set the selected route to previous.
                route = prevRoute;
             }
          }
       }
    }
    return route;
```

Algorithm 4. Route Memory

3.5 Driving the Route and Re-routing (Step 7)

While its vehicle is travelling its selected route, the agent evaluates its performance at each intersection. As the vehicle approaches an intersection, the agent compares the amount of time the vehicle has taken to reach this point in its route and compares it to the estimated time, multiplied by a performance factor. The performance factor is used to prevent the agent from evaluating the route time as simply slower than estimated, the difference between the two must be large enough that the agent has reason to believe that its route selection was a bad decision.

If the agent determines that its route is not performing as expected, they will re-route, using the next intersection as its origin point and the same method as described in sections 2 and 3. The next intersection is used to allow the agent time to develop a new route and position the vehicle to execute it appropriately. The vehicle will then continue travel using the new route selected.

The agent will only re-route once while on route. This limitation is set to prevent the agents from attempting to re-route at each intersection, thus reducing the amount of communication required. This limitation will also aid the agents in reaching an equilibrium, avoiding large changes in road congestion that may occur if each agent is constantly changing its routes.

Re-routing is implemented using the *ReRouting* algorithm.

Algorithm ReRouting

Input: Agent's route, Actual route time, Performance factor.

Output: Original route or new route.

```
route = The agent's current route;
performanceFactor = Performance factor;
actualTime = Total time the agent's vehicle has spent on route;
potentialRouteList = null;                          // Empty list of potential routes if
                                                    // re-routing.

// Check if the actual route time to the end of the road segment is greater than
// the estimated route time to this point multiplied by the performance factor.
if (actualTime > (route.roadSegment.EstTime * performanceFactor)) {
   // The agent will re-route from the next intersection of the current route.
   if (route.nextRoadSegment != route.lastRoadSegment) {
      potentialRouteList = Call ModA*;              // Algorithm 2: ModA*
      route = Call RouteSelector(potentialRouteList);
   }
}

return route;
```

Algorithm 5. ReRouting

3.6 Applying the Learning Algorithm (Step 8)

An agent's route selection is affected by the weighting value they use in the SAW formula. This weight must be learned by the agent, as its ideal value may change over time, as traffic congestion along routes change. The learning process will utilize the following steps:

1) After the agent's vehicle completes a route, the agent will request the actual travel times for its alternate routes from the database. The times used will be the most recent road segment completion time averages, providing the agent with the travel time they would likely have achieved if they had selected a given alternate route.

2) The agent selects the route with the fastest actual travel time - this list includes the route they just completed - and uses the SAW formula (6) to find the new weight, w_{new}:

$$ActualTravelTime = w_{new} * \frac{\sum_{i=0}^{k} RouteTime_j(i)}{SimulationDays} + (1 - w_{new}) * \frac{\sum_{i-\delta}^{k} RouteTime_j(i)}{\delta}$$

Rewritten, we use the formula as:

$$w_{new} = \frac{(FastestActualTravelTime - ShortTermAverage)}{(LongTermAverage - ShortTermAverage)} \qquad (8)$$

3) The newly calculated weight represents the weighting value that would have allowed the agent to select the fastest route, given the previously available travel time data. However, the agent may choose to adjust it, depending on its learning strategy.

The agent's learning strategy will determine how aggressively it will change its routing weight. An exploratory strategy (Sutton and Barto, 2014) would cause the agent to accept the newly calculated weight and use it in their next routing problem. However, this strategy may not be advisable if patterns of congestion change rapidly from one routing period to the next.

The agent may also use a strategy of exploitation (Sutton and Barto, 2014), in which the weight changes very little from one route to the next, hoping to ride out any fluctuations in road congestion in favour of long-term route stability.

To resolve this issue, the agent will treat the selection of a new weight as a multi-armed bandit (MAB) problem, where the arms to be chosen are the existing weight being used by the agent and the new weight calculated. The selection method is similar to that used by the Upper Confidence Bound 1 (UCB1) MAB algorithm (Sutton and Barto, 2014), in that it is deterministic in how it selects arms, rather than stochastic.

A deterministic method was chosen due to the large amount of actual travel time data available to the agent. As the agent has the near perfect ability to determine which weight would have given the best route in the previous routing episode, the randomness associated with stochastic methods would not be helpful.

We use a maximum weight change factor to determine whether the agent's strategy is one of exploration (a high change) or exploitation (a low change). The weight change factor remains fixed for the agent over successive routing tries. If the difference between the new weight and the previous weight is greater than the weight change factor, then the agent will use the previous weight adjusted by the weight change factor.

The weight change is adjusted as follows:

$$w_{used} = \begin{cases} \max\{w_{new}, (w_{prev} - wcf)\}, & w_{new} < w_{prev} \\ \min\{w_{new}, (w_{prev} + wcf)\}, & w_{new} \geq w_{prev} \end{cases} \quad (9)$$

Where wcf is the weight change factor, w_{new} is the newly calculated weight, w_{prev} is the weight used in the recently completed routing episode, and w_{used} is the adjusted weight that will be used by the agent.

Algorithm *LearnSAWWeight* performs the agent SAW weight update.

Algorithm LearnSAWWeight

Input: Potential route list, Actual route time, Previous SAW weight, Weight change Factor.

Output: Newly calculated SAW weight or previous SAW weight with adjustment.

potentialRouteList = List of potential routes from ModA* algorithm;

routeTime = The total time to complete agent's current route;

prevWeight = The SAW weight the agent used to determine its recent route;

totalTimeList = List of actual total times for each potential route;

wcf = Weight change factor;

Algorithm 6. LearnSAWWeight

Algorithm LearnSAWWeight ***continued***

```
// Build a list of actual total times for each potential route.
for (each potentialRoute in potentialRouteList) {
   for (each roadSegment in potentialRoute) {
      // Accumulate the average travel time for the road segment in the most recent
      // simulation, adjusted for the time the agent's vehicle would be travelling on it.
      totalCurrentTime = totalCurrentTime + averageRoadSegmentTime;
   }
   totalTimeList.add(totalCurrentTime);
}

// Select the route with the fastest time.
fastestRouteTime = 10000000;              // Stores the fastest time.
fastestIndex = 0;                         // Stores the route list index of fastest.
for (each time in totalTimeList) {
   if (time < fastestRouteTime) {
      fastestRouteTime = time;
      fastestIndex = totalTimeList.CurrentIndex;
   }
}
```

Algorithm 6. LearnSAWWeight

Algorithm LearnSAWWeight ***continued***

```
// Retrieve the average long and short-term travel times for the fastest route.
for (each roadSegment in potentialRouteList[fastestIndex]) {
   // Calculate estimated start time at road segment.
   estStartTime = routeSegment.EstEndTime – routeSegment.EstTravelTime;
   // Request travel times from database for segment.
   roadSegment.LongTermTime = Get Long-term average by estStartTime;
   roadSegment.ShortTermTime = Get Short-term average by estStartTime;
   // Add the segment times to the route times.
   longTermTime = longTermTime + roadSegment.LongTermTime;
   shortTermTime = shortTermTime + roadSegment.ShortTermTime;
}

// Calculate the weight that would have allowed the agent to select the route.
newWeight = (fastestRouteTime – shortTermTime)
                              / (longTermTime – shortTermTime);          // Formula 8.

// Adjust the newWeight by the cut-off factor if the difference is too large.  Formula 9.
if (|newWeight – prevWeight| > wcf) {
  if (newWeight >= prevWeight) {
    newWeight = prevWeight + wcf;
  }
  else {
    newWeight = prevWeight – wcf;
  }
}
return newWeight;
```

Algorithm 6. LearnSAWWeight

Chapter 4: Experimental Design

4.1 Hypothesis and Research Questions

We propose the following hypothesis:

A multi-agent system using a combination of centralized real-time traffic information system and direct agent experience will achieve a user equilibrium with a lower total route time than is possible using either method alone.

Additionally, we would like to answer the following questions:

4) Will such a multi-agent system achieve user equilibrium with fewer routing episodes than either a centralized real-time traffic information system or direct agent experience?
5) Will re-routing while on route result in lower total route times than when no re-routing is used?
6) Will the weighting factor reach an equilibrium point at which the agent will no longer make adjustments between routing episodes?

4.2 Experimental Design

We tested our hypothesis through the use of simulation. The simulations were run using a variety of parameters to determine the conditions under which routing would be most effective at reducing delays due to congestion. As a control, simulations were also run in which the agents were limited to using only the travel time information they were able to collect through direct experience, as a typical driver would. The resulting route

times were then compared to determine if the agents saw an improvement by using either the TIS or TIS/direct experience method.

4.2.1 Simulation Hardware and Software

The agent software, including the route building and learning components, was developed in Java 1.8.0_121 using the NetBeans IDE, version 8.2. Individual agent configurations and data collection were performed using MySQL 8.0. The road simulations were run using SUMO 0.27.1 (DLR, 2017), an open source traffic simulator. All simulations were executed on a laptop using four Intel Core i7-7500U CPUs at 2.7Ghz with 8GB of RAM.

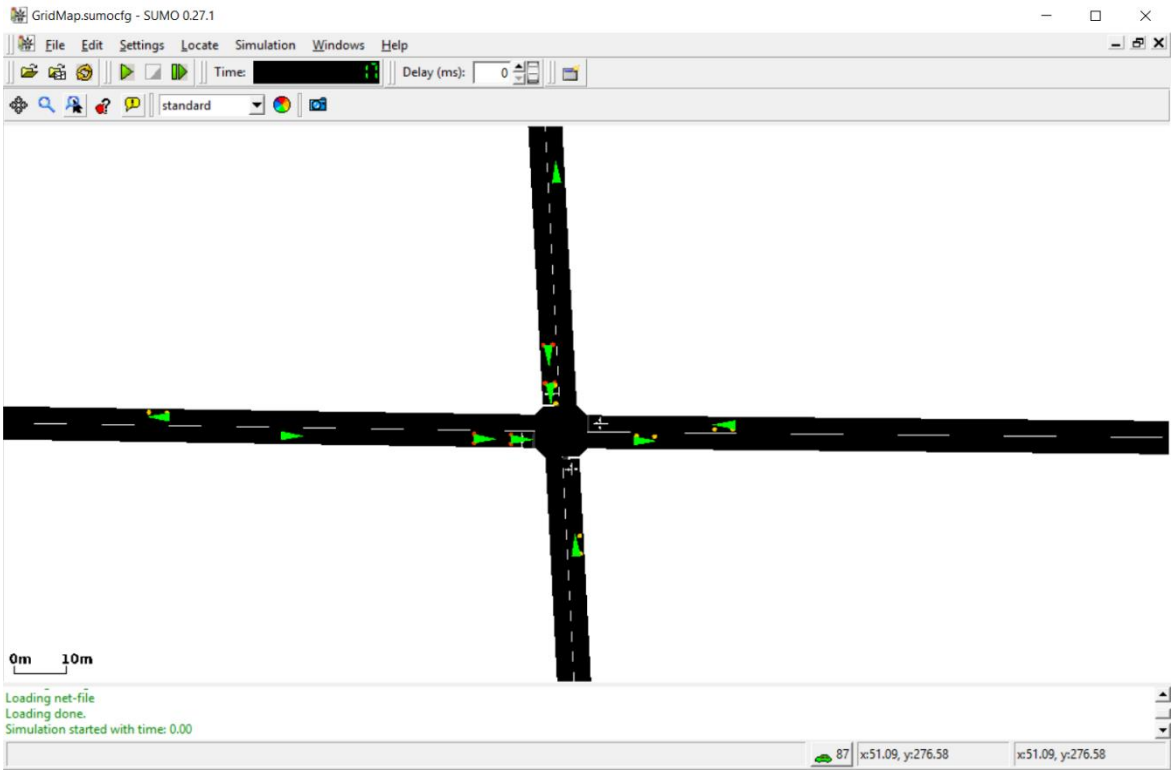

Figure 2. SUMO Traffic Simulator

4.2.2 Simulation Configurations

All simulations use the same map lattice – 25 standard city blocks of 100 metres to a side, arranged in a 5x5 grid. A grid was selected to provide a consistent distance between all intersections, allowing the agent a choice of paths that may have varying amounts of congestion, but not a significant difference in length. As the agent has several possible equal length routes available to it, the selection of a route becomes one of how much congestion is acceptable, rather than distance. A diagram of the map grid is presented in *Figure 3*.

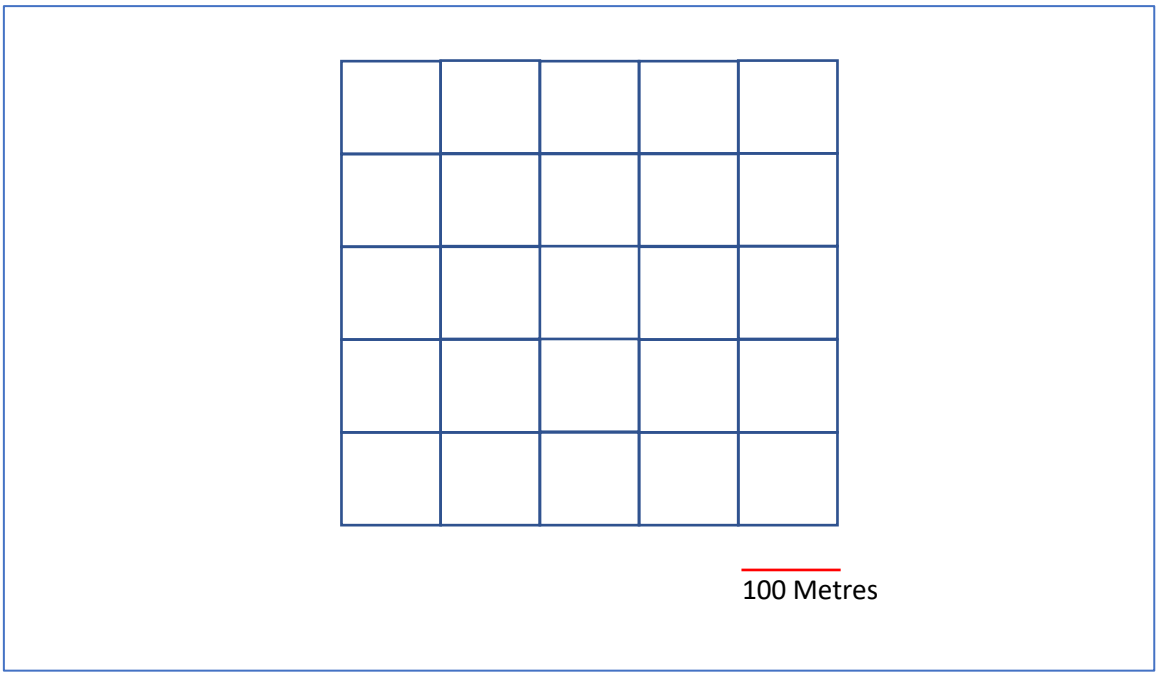

Figure 3. 5x5 Lattice Map

Each simulation, with the exception of the first set below, simulates 100 agents using the map simultaneously. Although configuration parameters are changed between simulations, all simulations use the same set of origin and destination locations for the

agents. As well, each agent is limited to starting and ending their route at an intersection, rather than in the middle of a road segment.

We limit the number of simulation runs to 60 per set of parameters in each scenario, with the exception of the first. This limit is selected as each simulation run represents the same day in repeated weeks. As such, for the routing method to be of value to a driver, it must produce improved routing results over a small number of attempts, leading us to use a limited number of runs for each scenario.

The following six sets of simulation scenarios are used:

1) Each agent simulated individually

Each agent is provided with the five fastest routes from the modified A* algorithm and allowed to run through each as the sole agent in the simulation. The fastest of the five is then selected as the fastest possible route time for the agent to travel from its origin to its destination without delays due to traffic congestion.

2) Run the simulation with direct experience but no re-routing

The simulation is run using 100 agents that are limited to using only the travel time data they can collect directly. The agents start by exploring the five fastest potential routes from the modified A* algorithm to determine the fastest one and then select a route as they gain further experience. The SAW formula is used to estimate the fastest route, but each agent uses an epsilon-greedy algorithm (Sutton and Barto, 2014) to make their selection, with the epsilon value varied as one of the simulation parameters.

The simulations are run 60 times with the SAW weight fixed such that the same weight value is used for all simulations for a given set of parameters.

3) Run the simulation with a TIS but no re-routing

The simulation is run using 100 agents that are allowed to learn a new SAW weight at varying rates using all travel time data available from the TIS. The SAW formula is used to estimate the fastest routes from a list of potential routes and the agents select the fastest estimate. After each simulation, the agent reviews the actual travel time for each potential route and determines what the SAW weight would need to be for the agent to have selected the fastest route.

The simulations are run 60 times for each set of parameters.

4) Run the simulation with direct experience and re-routing

The simulation is run using 100 agents that are limited to using only the travel time data they can collect directly. This set of simulations is identical to the simulations in method 2, with the exception that a route performance factor of 1.5 is set for each set of simulations. The performance factor is a setting that allows the agent to calculate a new route from the next intersection they will occupy, to their destination. In the case of these simulations, the agent will only attempt to re-route if the total time they've experienced on a route is greater than 1.5 times the expected route time to that point. While an agent can consider re-routing at each intersection, each agent is only allowed to select a re-route once in each simulation.

The simulations are run 60 times for each set of parameters.

5) Run the simulation with a TIS and re-routing

The simulation is run using 100 agents that are allowed to learn a new SAW weight at varying rates using all travel time data available from the TIS. This set of simulations is identical to simulations in method 3, with the exception that a route performance factor of 1.5 is set for each set of simulations.

The simulations are run 60 times for each set of parameters.

6) Run the simulation with a combination of a TIS and direct experience

The simulation is run using 100 agents that are allowed to learn a new SAW weight at varying rates using all travel time data available from the TIS. Re-routing is not allowed for the agents.

This set of simulations differs from method 3 in that agents are also able to learn from direct experience. After an agent is provided a list of potential routes with estimated route times from applying the SAW formula, it reviews its previous route experience. If the route with the fastest estimated route time has not been used before, the agent will always select it. If the best estimated route is the same as the route used in the previous simulation, the agent selects the same route again. If the best estimated route is different from the route used in the previous simulation, the agent compares the estimated route time to the actual route time from the previous simulation. The previous route's travel time is modified by an exploration factor of 0.5. If the estimated route is faster than the adjusted previous route time, the agent selects the new route.

The simulations are run 60 times for each set of parameters.

4.3 Experimental Analysis

To measure the effectiveness of routing with a TIS we measure the *price of anarchy* (Shoham and Leyton-Brown, 2009). The price of anarchy is the social cost due to congestion. In the case of vehicle routing it can be measured as the increase in route times that wouldn't otherwise be experienced if congestion was non-existent.

The price of anarchy is calculated as the ratio of the social cost of congestion to the social cost at a minimizing action distribution s^* (Shoham and Leyton-Brown, 2009). From chapter 2, the social cost is measured as:

$$C(s) = \sum_{i \in N} \sum_{a_i \in A_i} s(a_i) c_{a_i}(s)$$

Where:

$\boldsymbol{N = \{1, \ldots, n\}}$ is a set of players of different types;

$\boldsymbol{A = A_1 \times \ldots \times A_n}$, where $\boldsymbol{A_i \subseteq 2^R\{\emptyset\}}$ is the set of actions. The action $\boldsymbol{a_i \in A_i}$ is selected by all players of type $\boldsymbol{i}$. The action a_i represents a segment of the player's path $\boldsymbol{A_i}$;

$\boldsymbol{c = (c_1, \ldots, c_k)}$, where $\boldsymbol{c_r}$: $\mathbb{R}_+ \longmapsto \mathbb{R}$ is a cost function for road segment $\boldsymbol{r \in R}$, and $\boldsymbol{c_r}$ is nonnegative, continuous and nondecreasing;

$\boldsymbol{s(a_i)}$ is the element of $\boldsymbol{s}$ that corresponds to the set of players of type $\boldsymbol{i}$ who select action $\boldsymbol{a_i \in A_i}$.

The price of anarchy is thus:

$$Price\ of\ Anarchy = \frac{C(s)}{C(s^*)}$$

In our research, the value of $C(s^*)$ is calculated by summing the fastest route time for each agent when no other agents are being simulated. This data is collected using simulation scenario 1. As such, the minimized action distribution represents the fastest route time possible, given the list of origins and destinations being used.

The value of $C(s)$ is calculated as the sum of the route times when all agents are simulated simultaneously. The price of anarchy ratio will always give a value greater than or equal to 1, where 1 indicates the agents have found a set of routes that provides the fastest possible route times. The method with the lowest price of anarchy for a given set of parameters at user equilibrium will be considered to provide the fastest routing solutions for all agents.

Chapter 5: Results

5.1 Results Measured

For all simulation methods, with the exception of the first, the following data are presented for each set of parameters:

1) The total route time for all agents on the 60th simulation. As this is the final simulation run for a given set of parameters, it represents the point at which the agents will no longer be able to modify their routes.
2) The price of anarchy at the 60th simulation.
3) The minimum price of anarchy across all simulations.
4) The mean price of anarchy. This value is presented to show the difference between the final simulation results and the average for the method.
5) The median price of anarchy. This value is presented to show the overall effectiveness of the method across all simulations.
6) The number of times user equilibrium was achieved. Equilibrium may last for a single pair of simulations, or may be repeated across multiple simulations.
7) Where re-routing is used, the minimum number of re-routes across all simulations.

5.2 Each agent simulated individually

The total travel time for all agents using their best route is: 3431.4 seconds. This number is used as the $C(s^*)$ value when calculating the price of anarchy.

5.3 Run the simulation with direct experience but no re-routing

Table 1 presents the simulation results for direct experience with no re-routing. Each set of simulations is run 60 times and has a fixed SAW weight, such that the agent does not change the weight between simulations. The mean and median price of anarchy is calculated using only the 6th through 60th simulations as the agents are still exploring potential routes in the 1st through 5th simulations, which would skew the values.

The epsilon values determine the percentage chance that the epsilon-greedy algorithm will select a route other than the fastest provided by the *ModA** algorithm and their previous experience. Thus, an epsilon of 10 represents a 10 percent chance that the agent will select a random alternate route in an attempt to find a faster route along traffic congested roads.

Table 1.

Direct Experience w No Re-routing, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Min. Price of Anarchy	Mean Price of Anarchy	Median Price of Anarchy	No. Times at Equilibrium
SAW weight=1 Epsilon=0	4190.8	1.221	1.161	1.203	1.207	2
SAW weight=1 Epsilon=5	4088.9	1.192	1.152	1.214	1.211	0
SAW weight=1 Epsilon=10	4306.3	1.255	1.152	1.22	1.214	0
SAW weight=1 Epsilon=15	4161.3	1.213	1.17	1.268	1.268	0
SAW weight=1 Epsilon=20	4327.7	1.261	1.226	1.306	1.299	0
SAW weight=0.75 Epsilon=0	4046.6	1.179	1.169	1.186	1.182	8
SAW weight=0.75 Epsilon=5	3983.3	1.161	1.161	1.209	1.206	0
SAW weight=0.75 Epsilon=10	4337.3	1.264	1.168	1.242	1.242	0
SAW weight=0.75 Epsilon=15	4488.8	1.308	1.205	1.287	1.281	0

SAW weight=0.75 Epsilon=20	4478.3	1.305	1.196	1.301	1.297	0
SAW weight=0.5 Epsilon=0	4083.5	1.190	1.175	1.199	1.194	6
SAW weight=0.5 Epsilon=5	4219	1.23	1.165	1.216	1.214	0
SAW weight=0.5 Epsilon=10	3980.9	1.160	1.16	1.236	1.227	0
SAW weight=0.5 Epsilon=15	4613.8	1.345	1.158	1.267	1.274	0
SAW weight=0.5 Epsilon=20	4367	1.273	1.196	1.287	1.289	0
SAW weight=0.25 Epsilon=0	4014.7	1.17	1.149	1.182	1.179	8
SAW weight=0.25 Epsilon=5	4014.5	1.17	1.15	1.214	1.211	0
SAW weight=0.25 Epsilon=10	4106.7	1.197	1.179	1.256	1.250	0
SAW weight=0.25 Epsilon=15	4140.8	1.207	1.183	1.253	1.250	0
SAW weight=0.25 Epsilon=20	4272.2	1.245	1.195	1.294	1.294	0
SAW weight=0 Epsilon=0	4019.4	1.171	1.169	1.196	1.187	5
SAW weight=0 Epsilon=5	4250.2	1.239	1.163	1.206	1.195	0
SAW weight=0 Epsilon=10	4563.8	1.330	1.169	1.242	1.242	0
SAW weight=0 Epsilon=15	4335.9	1.264	1.185	1.264	1.253	0
SAW weight=0 Epsilon=20	4342.8	1.266	1.195	1.288	1.285	0

5.3.1 Minimum and Median Price of Anarchy

Figure 4 displays the minimum and median price of anarchy for each set of parameters. The minimum price of anarchy was lowest when epsilon was set to 5 for all SAW weights, with the exception of w=0.5, where an epsilon of 15 provided the lowest value. The lowest median occurred where epsilon was 0, regardless of the weight used.

That the median price of anarchy was consistently lowest when epsilon is 0, while also being highest at an epsilon of 20, shows us that agents are able to exploit their known routes more effectively when other agents are less likely to explore novel routes.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.013 and 0.116, with the largest differences occurring where higher epsilon values were used.

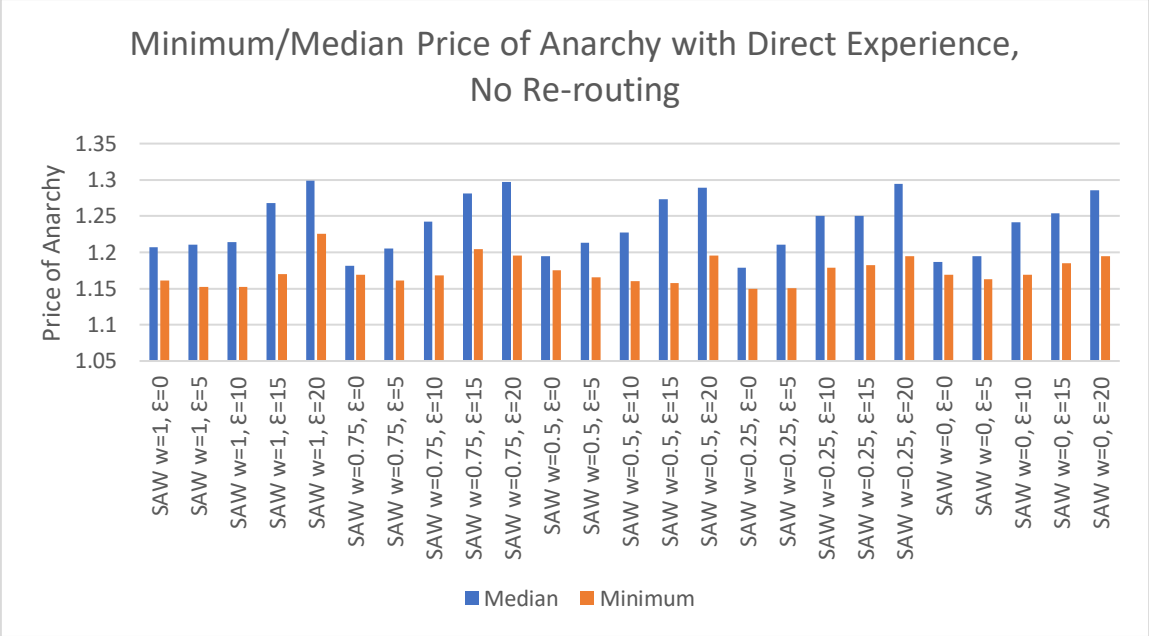

Figure 4. Minimum/Median Price of Anarchy w Direct Experience, No Re-routing

5.3.2 User Equilibrium Points

Figure 5 displays the number of occurrences of user equilibrium for each set of parameters. Equilibrium occurred only where the epsilon value was set to 0 as the agents would randomly select alternative routes when using higher epsilon values, regardless of their perceived likelihood of producing a better route time, preventing equilibrium from being achieved.

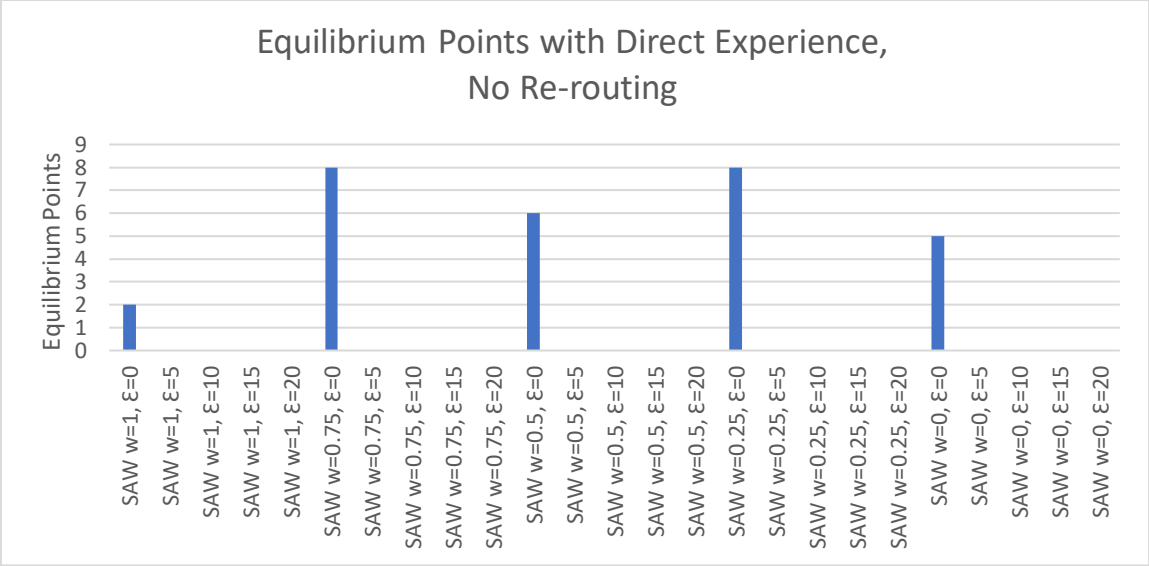

Figure 5. Equilibrium Points w Direct Experience, No Re-routing

5.4 Run the simulation with a TIS but no re-routing

Table 2 presents the simulation results for a TIS with no re-routing. Each set of simulations is run 60 times. The first five sets of simulations use a fixed SAW weight, the remaining sets allow the agents to change the weight by an amount ranging from 0.1 to 1.0 between each simulation.

The mean and median price of anarchy is calculated using only the 6th through 60th simulations as the agents have not yet produced sufficient travel time data to avoid using estimated times in the 1st through 5th simulations. Where travel time data is not available, the agent calculates an estimate using the allowed road speed and road segment length.

Table 2.

TIS w No Re-routing, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Min. Price of Anarchy	Mean Price of Anarchy	Median Price of Anarchy	No. Times at Equilibrium
SAW weight=1 Weight fixed	3920.8	1.143	1.089	1.123	1.123	0
SAW weight=0.75 Weight fixed	3897.7	1.136	1.097	1.122	1.118	0
SAW weight=0.5 Weight fixed	3932.2	1.146	1.098	1.120	1.117	0
SAW weight=0.25 Weight fixed	3920.5	1.143	1.092	1.127	1.126	0
SAW weight=0 Weight fixed	3848	1.121	1.11	1.144	1.141	0
SAW weight=0.5 Weight step=0.1	3824.9	1.115	1.089	1.118	1.114	0
SAW weight=0.5 Weight step=0.2	3788	1.104	1.094	1.121	1.121	0
SAW weight=0.5 Weight step=0.3	3784.4	1.103	1.079	1.118	1.117	0
SAW weight=0.5 Weight step=0.4	3893.4	1.135	1.096	1.122	1.122	0
SAW weight=0.5 Weight step=0.5	3911.3	1.14	1.093	1.125	1.125	0
SAW weight=0.5 Weight step=0.6	3847.2	1.121	1.099	1.128	1.128	0
SAW weight=0.5 Weight step=0.7	3777.8	1.101	1.093	1.123	1.12	0
SAW weight=0.5 Weight step=0.8	3856.5	1.124	1.098	1.123	1.122	0
SAW weight=0.5 Weight step=0.9	3939.5	1.148	1.097	1.122	1.119	0
SAW weight=0.5 Weight step=1.0	3910.5	1.14	1.089	1.120	1.120	0

5.4.1 Minimum and Median Price of Anarchy

Figure 6 displays the minimum and median price of anarchy for each set of parameters. The minimum price of anarchy was lowest when the agents could change their SAW weights by up to 0.3 between routing episodes. The median price of anarchy was

lowest where the agents were allowed to change their SAW weights by up to 0.1 between routing episodes.

The difference between the lowest median price of anarchy and the highest was 0.027. However, when fixed SAW weights are not included the difference drops to 0.014, indicating that there is greater similarity in the median price of anarchy when the agents are able to change their weight values, regardless of the amount of change allowed.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.019 and 0.038. This is a smaller range of differences than was found using direct experience only. As well, the highest median price of anarchy, 1.141, was found be lower than the lowest median value when using direct experience, 1.179.

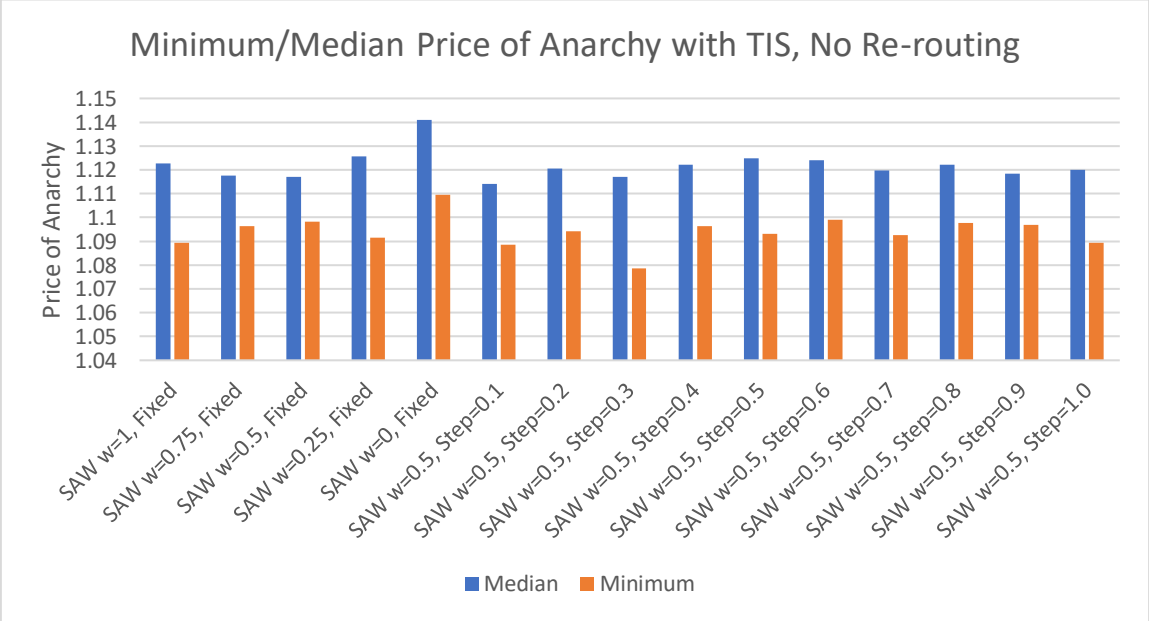

Figure 6. Minimum/Median Price of Anarchy w TIS, No Re-routing

5.5 Run the simulation with direct experience and re-routing

Table 3 presents the simulation results for direct experience with re-routing. Each set of simulations is run 60 times and has a fixed SAW weight, such that the agent does not change the weight between simulations. The mean and median price of anarchy is calculated using only the 6th through 60th simulations as the agents are still exploring potential routes in the 1st through 5th simulations, which would skew the values.

Each agent is allowed to re-route a maximum of 1 time per simulation with a performance factor of 1.5. The re-route decision is made at the end of each road segment by multiplying the estimated route time to the end of the segment by 1.5 and comparing the actual route time to that point. If the actual time is greater than this value and the agent has not re-routed, the agent will re-route.

Table 3.

Direct Experience w Re-routing, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Min. Price of Anarchy	Mean Price of Anarchy	Median Price of Anarchy	No. Times at Equilibrium	Min. No. Re-routes
SAW weight=1 Epsilon=0	4083.5	1.190	1.18	1.201	1.195	1	17
SAW weight=1 Epsilon=5	4204.5	1.225	1.164	1.232	1.230	0	18
SAW weight=1 Epsilon=10	4378.4	1.276	1.167	1.239	1.233	0	16
SAW weight=1 Epsilon=15	4370.9	1.274	1.196	1.273	1.272	0	18
SAW weight=1 Epsilon=20	4226.8	1.232	1.176	1.289	1.29	0	17
SAW weight=0.75 Epsilon=0	4093.8	1.193	1.188	1.2	1.193	0	15
SAW weight=0.75 Epsilon=5	4096.7	1.194	1.153	1.222	1.217	0	18

SAW weight=0.75 Epsilon=10	4080.8	1.189	1.162	1.241	1.243	0	16
SAW weight=0.75 Epsilon=15	4621.7	1.347	1.18	1.282	1.278	0	18
SAW weight=0.75 Epsilon=20	5041.5	1.469	1.202	1.299	1.288	0	18
SAW weight=0.5 Epsilon=0	3934.4	1.147	1.14	1.162	1.149	6	13
SAW weight=0.5 Epsilon=5	4298.4	1.253	1.161	1.209	1.194	0	12
SAW weight=0.5 Epsilon=10	4322.6	1.26	1.166	1.247	1.243	0	14
SAW weight=0.5 Epsilon=15	4261.1	1.242	1.18	1.254	1.245	0	16
SAW weight=0.5 Epsilon=20	4229.1	1.233	1.199	1.294	1.284	0	16
SAW weight=0.25 Epsilon=0	4001.1	1.166	1.155	1.192	1.187	1	12
SAW weight=0.25 Epsilon=5	4092.6	1.193	1.156	1.204	1.204	0	12
SAW weight=0.25 Epsilon=10	4088.2	1.191	1.165	1.24	1.235	0	13
SAW weight=0.25 Epsilon=15	4305.2	1.255	1.198	1.283	1.275	0	15
SAW weight=0.25 Epsilon=20	4313.9	1.257	1.194	1.287	1.295	0	16
SAW weight=0 Epsilon=0	4084.8	1.190	1.147	1.177	1.165	4	9
SAW weight=0 Epsilon=5	4026.6	1.174	1.137	1.201	1.191	0	12
SAW weight=0 Epsilon=10	4149.1	1.209	1.156	1.238	1.231	0	12
SAW weight=0 Epsilon=15	4371.7	1.274	1.199	1.289	1.285	0	15
SAW weight=0 Epsilon=20	4420.7	1.288	1.208	1.298	1.292	0	17

5.5.1 Minimum and Median Price of Anarchy

Figure 7 displays the minimum and median price of anarchy for each set of parameters. The minimum price of anarchy was lowest when epsilon was set to 5 for all SAW weights, with the exception of w=0.5, where an epsilon of 0 provided the lowest value. The lowest median occurred where epsilon was 0, regardless of the weight used.

As with direct experience with no re-routing, the highest median price of anarchy was found when an epsilon of 20 was used. The highest median value with re-routing, 1.29, is comparable to the highest median with no re-routing, 1.299. However, the lowest median with re-routing, 1.149, is lower than the lowest median with no re-routing, 1.179, suggesting that there is a small benefit to using re-routing.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.009 and 0.113. This is comparable to the difference found when no re-routing was used, 0.013 and 0.116, indicating that re-routing had little effect in reducing the gap between the minimum and median.

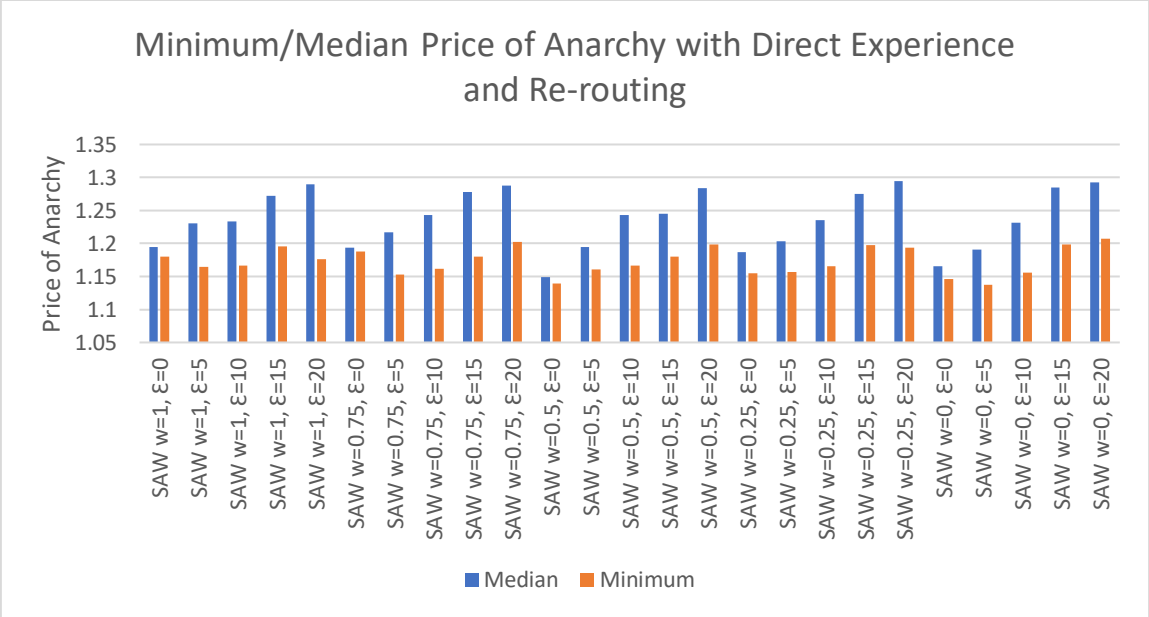

Figure 7. Minimum/Median Price of Anarchy w Direct Experience, Re-routing

5.5.2 User Equilibrium Points and Minimum Re-routes

Figure 8 displays the number of occurrences of user equilibrium and minimum re-routes for each set of parameters. Equilibrium occurred only where the epsilon value was set to 0, with the highest number of equilibrium points being 6 for w=0.5.

A comparison to direct experience with no re-routing indicates that the use of re-routing reduces the frequency of equilibrium.

The minimum number of re-routes was found to be lowest where the SAW weight was 0 and epsilon was 0, while the trend across all sets of parameters showed that the minimum number of re-routes was highest with larger epsilon values. This is consistent with the equilibrium point data in that lower epsilon values tended to result in fewer poor route selections that would require correction through re-routing.

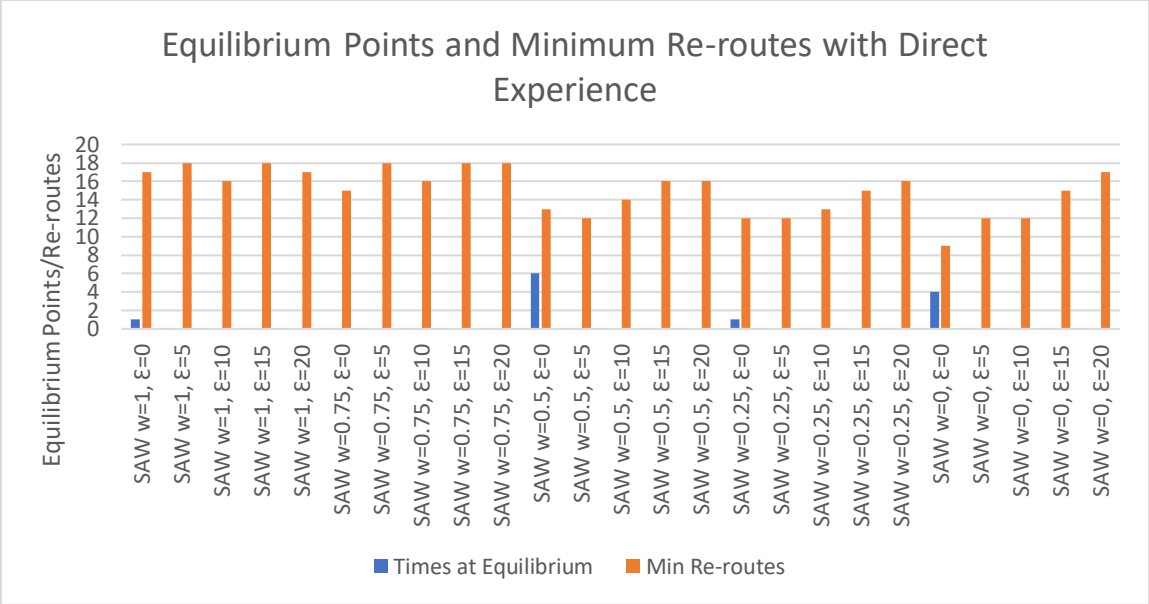

Figure 8. Equilibrium Points and Re-routes w Direct Experience

5.6 Run the simulation with a TIS and re-routing

Table 4 presents the simulation results for a TIS with re-routing. Each set of simulations is run 60 times. The first five sets of simulations use a fixed SAW weight, the remaining sets allow the agents to change the weight by an amount ranging from 0.1 to 1.0 between each simulation. Each agent is allowed to re-route a maximum of 1 time per simulation with a performance factor of 1.5.

The mean and median price of anarchy is calculated using only the 6th through 60th simulations as the agents have not yet produced sufficient travel time data to avoid using estimated times in the 1st through 5th simulations. Where travel time data is not available, the agent calculates an estimate using the allowed road speed and road segment length.

Table 4.

TIS w Re-routing, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Min. Price of Anarchy	Mean Price of Anarchy	Median Price of Anarchy	No. Times at Equilibrium	Min. No. Re-routes
SAW weight=1 Weight fixed	3899	1.136	1.106	1.131	1.127	0	12
SAW weight=0.75 Weight fixed	3814.9	1.112	1.104	1.127	1.124	0	9
SAW weight=0.5 Weight fixed	3924.7	1.144	1.1	1.128	1.127	0	7
SAW weight=0.25 Weight fixed	3862.2	1.126	1.106	1.133	1.131	0	6
SAW weight=0 Weight fixed	4136.9	1.206	1.107	1.144	1.147	0	7
SAW weight=0.5 Weight step=0.1	3867.3	1.127	1.087	1.118	1.116	0	7
SAW weight=0.5 Weight step=0.2	3886.3	1.133	1.094	1.13	1.129	0	18
SAW weight=0.5 Weight step=0.3	3880.5	1.131	1.101	1.124	1.122	0	7
SAW weight=0.5 Weight step=0.4	3955	1.153	1.093	1.129	1.127	0	7
SAW weight=0.5 Weight step=0.5	3889.8	1.134	1.094	1.124	1.123	0	8
SAW weight=0.5 Weight step=0.6	3947.9	1.151	1.091	1.124	1.122	0	8
SAW weight=0.5 Weight step=0.7	3803	1.108	1.094	1.124	1.121	0	7
SAW weight=0.5 Weight step=0.8	3819.9	1.113	1.094	1.121	1.118	0	8
SAW weight=0.5 Weight step=0.9	3801.9	1.108	1.088	1.122	1.123	0	8
SAW weight=0.5 Weight step=1.0	3853.2	1.123	1.088	1.123	1.122	0	7

5.6.1 Minimum and Median Price of Anarchy

Figure 9 displays the minimum and median price of anarchy for each set of parameters. The minimum price of anarchy was lowest when the agents could change their SAW weights by up to 0.3 between routing episodes. The median price of anarchy was

lowest where the agents were allowed to change their SAW weights by up to 0.1 between routing episodes.

The difference between the lowest median price of anarchy and the highest was 0.031. However, when fixed SAW weights are not included the difference drops to 0.012, indicating that, as with using a TIS and no re-routing, there is little difference between median price of anarchy when the agents are able to change their weight values.

The highest median price of anarchy with re-routing, 1.147, is comparable to that found with no re-routing, 1.141. When fixed weight routes are not considered, the highest median values are almost identical, at 1.129 with re-routing and 1.128 without. These results suggest that there is no advantage to using re-routing when the agents are able to make use of a TIS for travel time data.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.019 and 0.04. The highest median price of anarchy using a TIS and re-routing, 1.147, was found to be lower than the lowest median value when using direct experience with re-routing, 1.149.

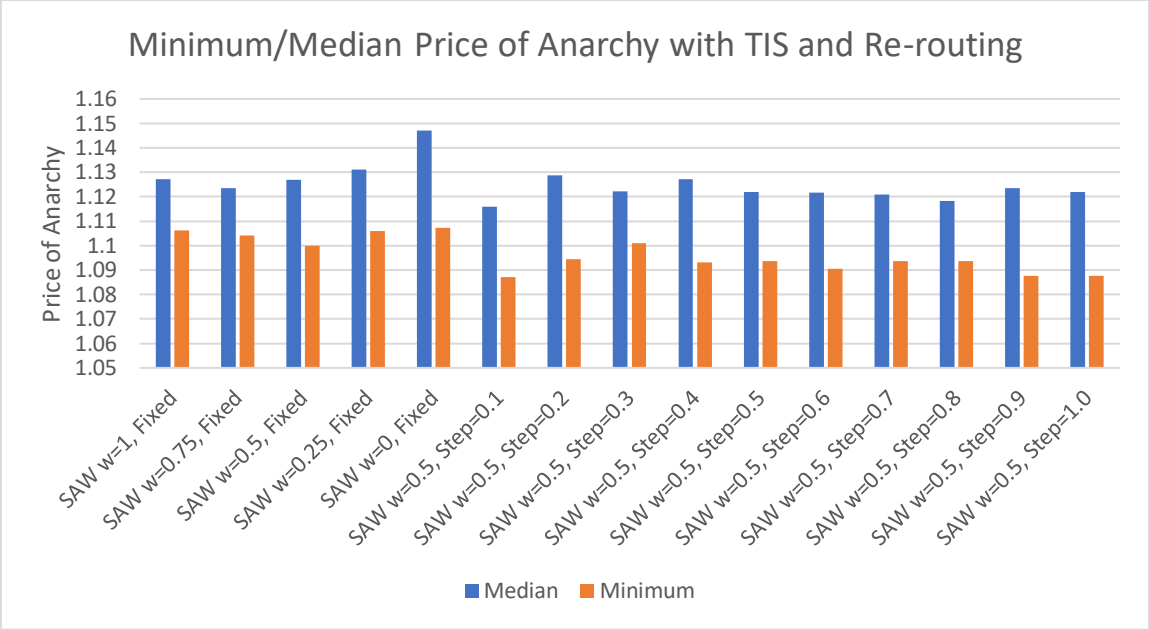

Figure 9. Minimum/Median Price of Anarchy w TIS, Re-routing

5.6.2 User Equilibrium Points and Minimum Re-routes

Figure 10 displays the number of occurrences of user equilibrium and re-routing for each set of parameters. No instances of equilibrium occurred.

The minimum number of re-routes was found to be lowest where the SAW weight was fixed at 0.25. The highest number of re-routes occurred where the agents could adjust their SAW weight by 0.2 between routing episodes. However, for other simulations where the agents were allowed to adjust their weights, the number of minimum re-routes varied from 7 to 8.

A comparison of the average minimum number of re-routes shows that direct experience routing used 15, while TIS routing used 8.4, indicating that using a TIS reduced the need to re-route while on route.

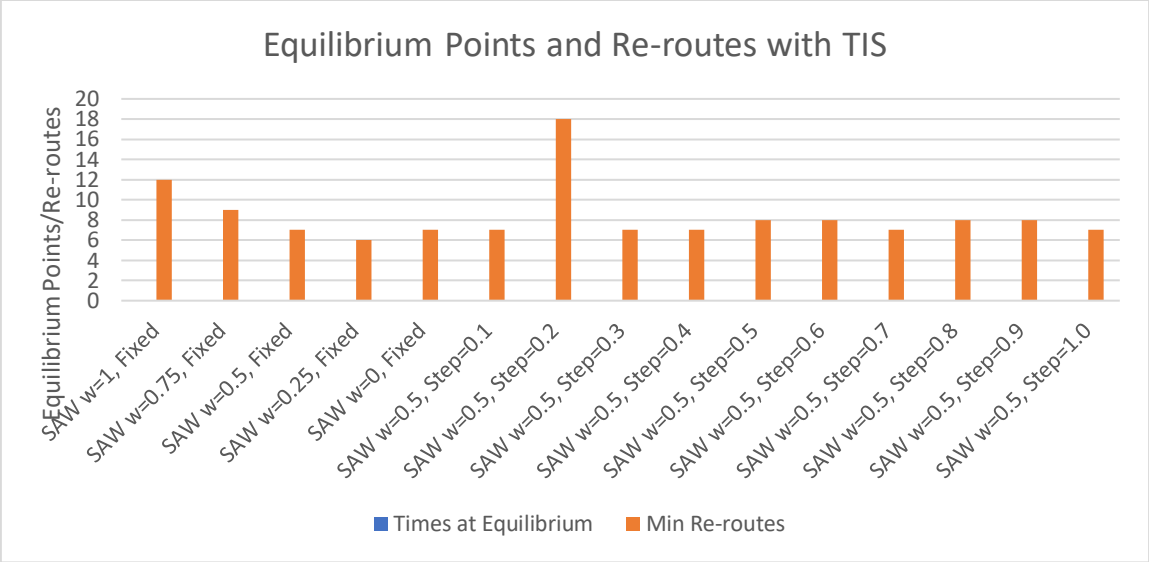

Figure 10. Equilibrium Points and Re-routes w TIS

5.7 Run the simulation with a combination of a TIS and direct experience

Table 5 presents the simulation results for a TIS and direct experience with no re-routing. Each set of simulations is run 60 times. The first five sets of simulations use a fixed SAW weight, the remaining sets start with a weight of 0.5 but allow the agents to change the weight by an amount ranging from 0.1 to 1.0 between each simulation. Each agent also uses its direct routing experience to determine whether to select a different route than was used in the previous simulation.

The mean and median price of anarchy is calculated using only the 6th through 60th simulations as the agents have not yet produced sufficient travel time data to avoid using estimated times in the 1st through 5th simulations. Where travel time data is not available, the agent calculates an estimate using the allowed road speed and road segment length.

Table 5.

TIS and Direct Experience, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Min. Price of Anarchy	Mean Price of Anarchy	Median Price of Anarchy	No. Times at Equilibrium
SAW weight=1 Weight fixed	3783	1.103	1.102	1.117	1.120	36
SAW weight=0.75 Weight fixed	3777.399	1.101	1.088	1.103	1.101	33
SAW weight=0.5 Weight fixed	3834.099	1.117	1.095	1.107	1.103	37
SAW weight=0.25 Weight fixed	3757.699	1.095	1.076	1.096	1.096	41
SAW weight=0 Weight fixed	3848	1.121	1.089	1.117	1.112	8
SAW weight=0.5 Weight step=0.1	3795.1	1.106	1.09	1.107	1.106	10
SAW weight=0.5 Weight step=0.2	3836.3	1.118	1.081	1.114	1.118	22
SAW weight=0.5 Weight step=0.3	3769.6	1.099	1.096	1.107	1.099	26
SAW weight=0.5 Weight step=0.4	3770.2	1.099	1.083	1.102	1.099	17
SAW weight=0.5 Weight step=0.5	3738.6	1.09	1.09	1.104	1.102	19
SAW weight=0.5 Weight step=0.6	3738.6	1.09	1.09	1.104	1.102	19
SAW weight=0.5 Weight step=0.7	3738.6	1.09	1.09	1.104	1.102	19
SAW weight=0.5 Weight step=0.8	3773.3	1.1	1.1	1.105	1.098	21
SAW weight=0.5 Weight step=0.9	3722.4	1.085	1.073	1.092	1.085	34
SAW weight=0.5 Weight step=1.0	3730.1	1.087	1.071	1.094	1.087	30

5.7.1 Minimum and Median Price of Anarchy

Figure 11 presents the median and minimum price of anarchy for each set of parameters. The median price of anarchy was found to be lowest where the agents were allowed to change their SAW weights by up to 0.9 per routing episode. The lowest minimum price of anarchy was found when the agents could change their SAW weights by up to 1.0 per routing episode.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.003 and 0.037. The highest median using a TIS with individual experience, 1.120, was found to be less than the lowest median when using individual experience with re-routing, 1.149, but slightly higher than the lowest median using a TIS with re-routing, at 1.116.

The difference between the minimum and median price of anarchy for each set of routing parameters was found to vary between 0.003 and 0.037. This is a smaller variation than that found for direct experience with re-routing, 0.009 to 0.113, but larger than when using a TIS with re-routing, 0.019 to 0.04.

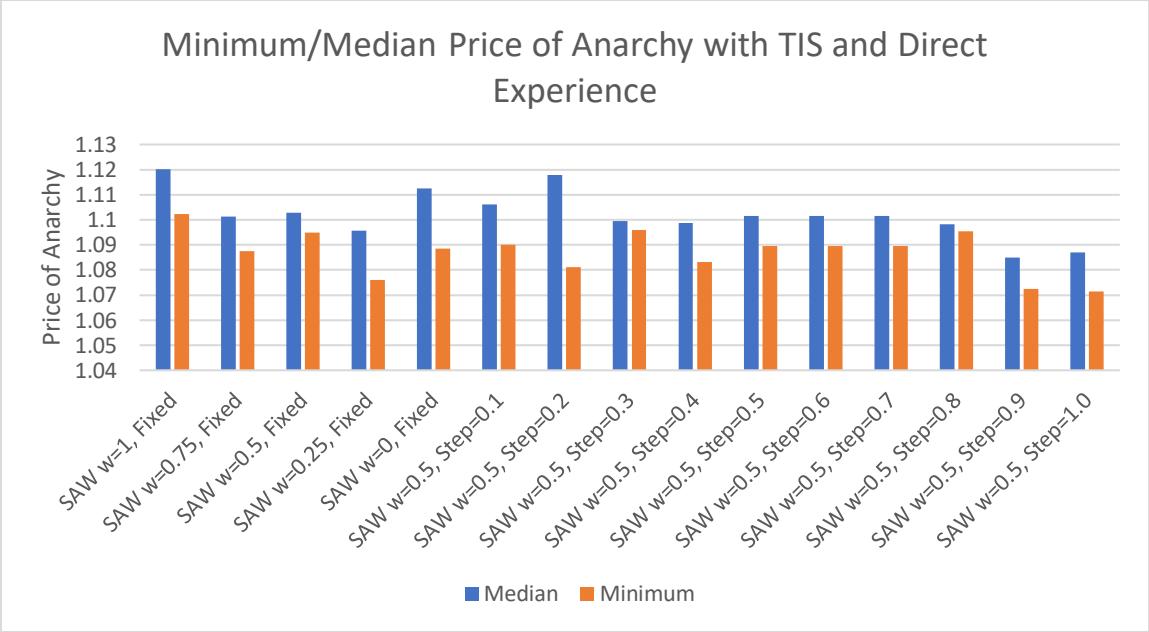

Figure 11. Minimum/Median Price of Anarchy w TIS and Direct Experience

5.7.2 User Equilibrium Points

Figure 12 presents the number of occurrences of user for each set of parameters. Equilibrium occurred with all sets of parameters, but was highest where the SAW weight was fixed and weight was not equal to 0. This method was the only one found to have reached equilibrium regardless of the parameters used.

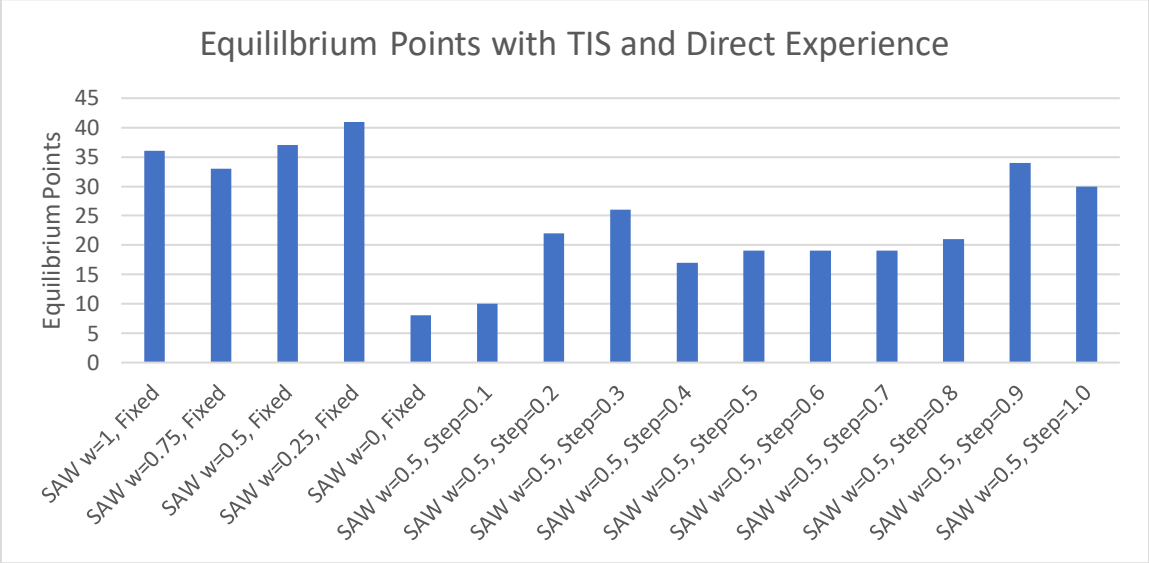

Figure 12. Equilibrium Points w TIS and Direct Experience

Chapter 6: Discussion

6.1 Parameters Discussed

In chapter 1 we outlined three objectives for our research, while in chapter 4 we presented our hypothesis and three questions to study. In this chapter we analyze our research in light of these and discuss its importance.

We will reference *Table 6*, which presents the price of anarchy and total travel time data for the set of simulation parameters that produced the lowest median price of anarchy for each of our routing methods. *Figure 13* and *Figure 14* display the simulation results for each of the parameter sets in *Table 6*.

Table 6.

Parameters by Method w Lowest Median Price of Anarchy, Results

Simulation Parameters	Total Time on 60th Sim	Price of Anarchy 60th Sim	Mean Price of Anarchy	Median Price of Anarchy
Direct exp., no re-routing SAW weight=0.25 Epsilon=0	4014.7	1.17	1.182	1.179
TIS, no re-routing SAW weight=0.5 Weight step=0.1	3824.9	1.115	1.118	1.114
Direct exp., with re-routing SAW weight=0.5 Epsilon=0	3934.4	1.147	1.162	1.149
TIS, with re-routing SAW weight=0.5 Weight step=0.8	3819.9	1.113	1.121	1.118
TIS with Direct exp. SAW weight=0.5 Weight step=0.9	3722.4	1.085	1.092	1.085

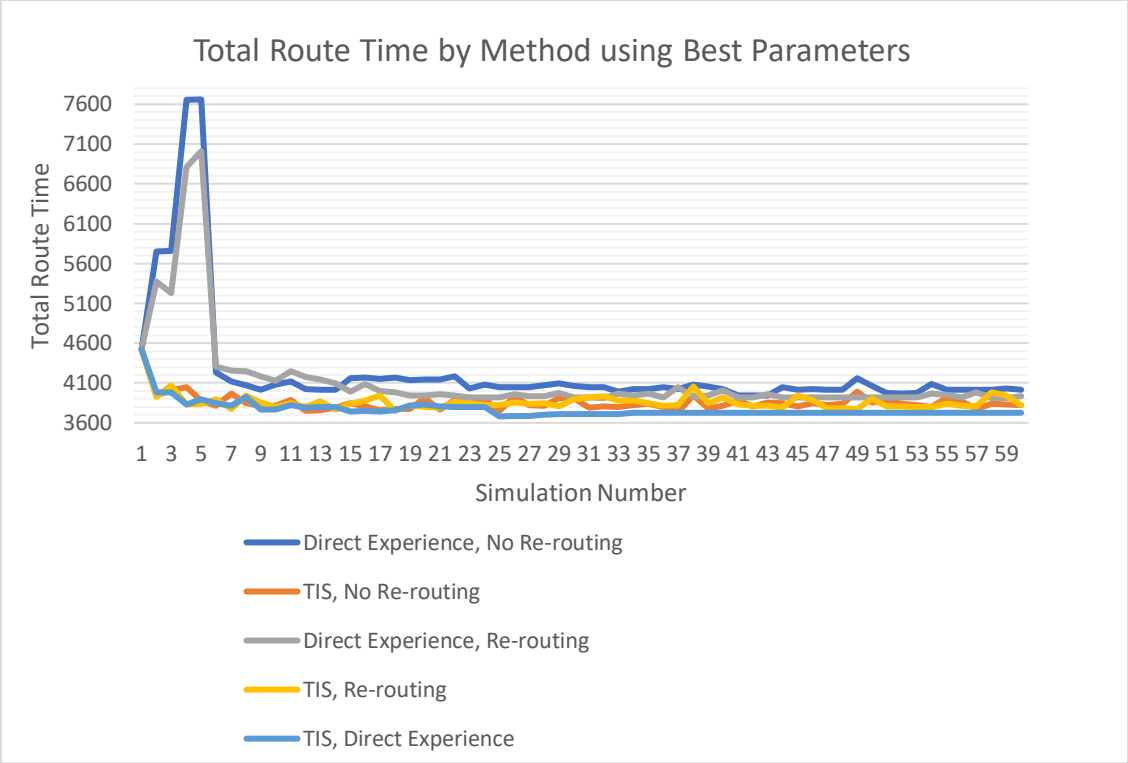

Figure 13. Total Route Time by Method using Best Parameters

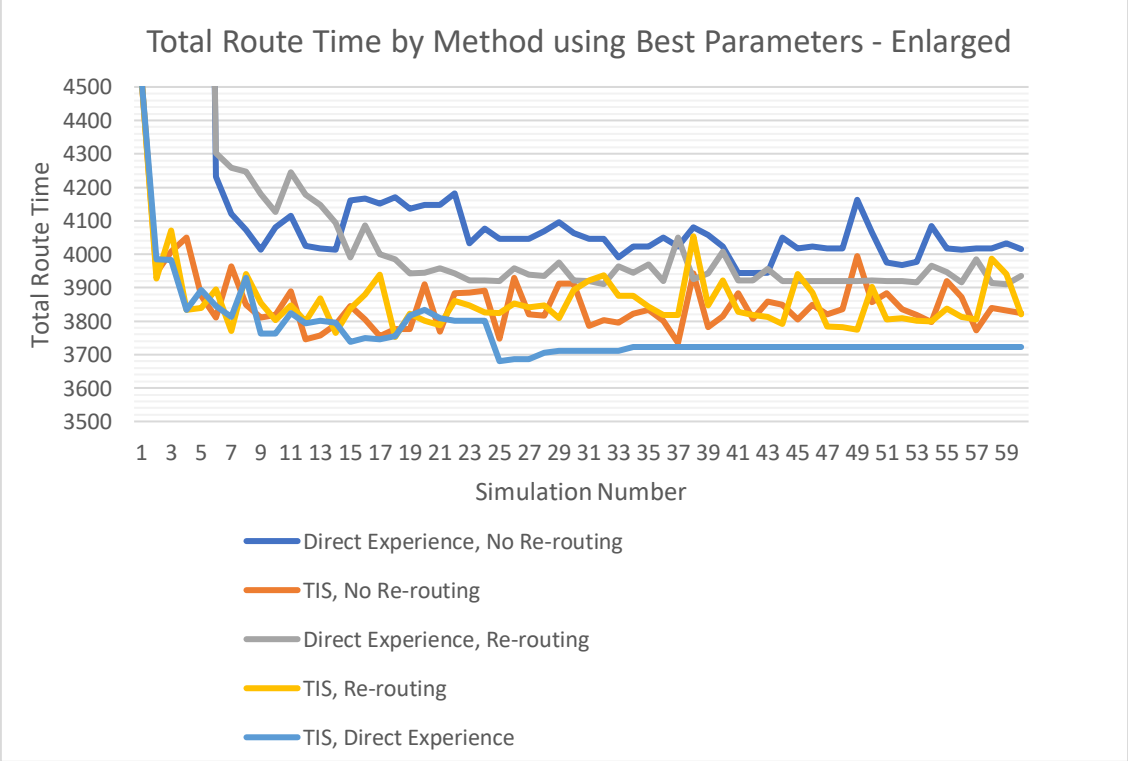

Figure 14. Total Route Time by Method using Best Parameters - Enlarged

6.2 Research Objectives

We discuss our results as they pertain to our research objectives.

6.2.1 Objective 1

Our first research objective is the following: **Determine the fastest route for the driver with the least exploration.**

We note that all routing methods produced an initial simulation with a total route time of 4524.8 seconds. This is due to each method using the same modified A* algorithm, *ModA**, to build each agent's list of potential routes. As the algorithm relies upon map data alone, it always provides the same fastest route, and as the agents have no initial travel time data beyond that provided by the map, they always select the same route.

Once the agents have completed their first simulation the methods diverge. Reviewing *Figure 13*, shows that the both the TIS and TIS with direct experience methods were able to quickly reduce their total route times from the initial simulation, while the methods that rely on direct experience alone see a large increase in route times before dropping in the 6^{th} simulation.

This increase in route times is due to the agents' need to explore alternative routes to develop an initial set of travel time data to apply to the potential route list. Once the agents have completed their initial exploration, they use the travel time data collected to develop routes that improve upon the total route time found in the initial simulation.

The cost of exploration – that the agents must try potentially worse routes to determine which is better with congestion – shows us a drawback to relying on direct experience alone. If the agent must rely on their own experiences to determine which route

is most likely to be the fastest on congested roads, then they will always see poorer routing results while acquiring this experience. As the TIS methods use the collective travel time data of all agents, each agent is able to avoid this period of exploration and see routing improvements immediately.

6.2.2 Objective 2

The second research objective is: **The routing method must adapt to changes in congestion over successive routing actions.**

Each of the routing methods showed adaptation to changes in congestion. When using direct experience alone, with fixed SAW weights, the agents were able to produce a lower median price of anarchy when using a lower weight value. As well, *Figure 4* and *Figure 7* show that, with or without re-routing, a weight value of 1.0 resulted in the highest median price of anarchy.

The poorer performance when using a high SAW weight is due to the agent's reliance upon long-term travel time data, which would be used exclusively when the weight is set to 1.0. As the long-term travel times used by the agents are averages from all previous simulations, the agent is less able to adapt to recent changes in congestion which comprise a relatively small component of this value. When combined with the agent's limited ability to gather travel time data, this results in slower route times.

Connecting the agents to a TIS produced greater adaptability to changes in congestion. As each agent had access to the travel time data produced by all other agents, they were able to select the route that was predicted to give the fastest route time, regardless of whether they had used it previously.

Under these conditions, the agents were found to produce the lowest median price of anarchy where the SAW weight was not fixed. During simulation, many agents were observed to change their weight values between routing episodes, often by as much as they were allowed to by the step limit. However, some agents rarely changed their SAW weights. This shows us that it is better for the agents to be able to modify their weight value to adapt to changes in the congestion problem.

Finally, when the agents were able to combine both a TIS and direct experience, they were best able to adapt to changes in congestion. This is reflected in this method having the lowest median price of anarchy. The use of direct experience prevents the agents from chasing the fastest route without regard for past results, allowing the congestion on each road segment to stabilize and become more predictable.

6.2.3 Objective 3

The final objective for this research is: **The routing method must produce routes that are fair for each driver but also minimizes the total travel time for all drivers.**

The multi-agent routing methods produced routes that were fair to the drivers, as each agent had the goal of selecting the fastest route to reach their destination. However, while each method proved successful at reducing the median price of anarchy and the price of anarchy on the 60th simulation, the combined TIS with direct experience method showed the lowest values.

Figure 14 shows that the TIS with direct experience method provided a total route time on the 60th simulation that was 100 seconds lower than when using a TIS alone, and 212 seconds lower than direct experience with re-routing. As well, user equilibrium was

maintained from the 34th simulation, demonstrating that all agents had found the fastest route possible, given the existing congestion.

6.3 Research Questions

We discuss our results in answer to our three research questions.

6.3.1 Research Question 1

Our first research question: **Will a multi-agent system using a combination of central database and direct agent experience achieve user equilibrium with fewer routing episodes than either a centralized real-time traffic information system or direct agent experience?**

User equilibrium was found to occur where the agents used direct experience alone, with or without re-routing, and where the agents used a combination of a TIS with direct experience. User equilibrium was not observed where only the TIS was used.

6.3.1.1 User Equilibrium with Direct Experience Alone

Figure 5 and *Figure 8* show that where direct experience alone was used, the agents reached equilibrium more often when they were unable to re-route. This was due to the effect of re-routing on the congestion problem. Where many agents re-route, the congestion on any given road segment may change from routing episode to routing episode. As the congestion changes, the agents adapt their paths to produce better route times, resulting in fewer instances of user equilibrium.

6.3.1.2 User Equilibrium with the TIS Alone

The number of agents changing routes in consecutive simulations ranged between 10 to 20 percent, regardless of whether re-routing was.

This is due to the agents adjusting their paths to select the fastest route given the existing congestion pattern. As the agents are not including their direct experiences routing – relying only on the collective experience of the TIS – they select the fastest route recommended by their application of the SAW formula (6). While many agents change their routes, the congestion pattern changes, thus the agents never reach a point of user equilibrium. It should be noted, however, that most agents will use the same route in consecutive routing episodes.

6.3.1.3 User Equilibrium with the TIS and Direct Experience

User equilibrium was found to occur most often when using a combination of TIS and direct experience. *Figure 12* shows that user equilibrium occurred for all parameter sets and many more times than when direct experience alone was used.

This was due to the agents use of past routing experiences to guide their selection of future routes. As the agents only selected routes that had been shown to be fast in the past, they were less likely to change routes between routing episodes. As such, once a fast route was found, the agents stayed with it, resulting in more instances of user equilibrium.

The earliest point of equilibrium was found to occur by the 10^{th} simulation for most sets of parameters, although this was not held for more than two consecutive simulations. The agents in *Figure 14* reached a point of consistent equilibrium by the 26^{th} simulation,

however, the equilibrium point does change at the 28th and 34th simulations, where some agents adjust their routes.

As using a TIS with direct experience produces user equilibrium points that last across many simulations, and does so earlier than direct experience alone, we can say that this method does achieve user equilibrium with fewer routing episodes than either of our other methods.

6.3.2 Research Question 2

Our next research question: **Will re-routing while on route result in lower total route times than when no re-route is used?**

Re-routing while on route lowered total route times for agents using direct experience alone but had little effect when using a TIS.

When agents had no access to a TIS, the ability to re-route allowed the agents to correct for a poor routing decision given the congestion encountered on its route. This is reflected in a lower median price of anarchy when re-routing is used.

Using a TIS removed the advantage given by re-routing, as the agents are relying upon travel time data provided by all agents and are able to make better routing choices.

The difference between the two methods shows that there is no advantage to using re-routing where a TIS is available. However, in instances where the TIS becomes unavailable – for example, in the case of a communications failure – the ability to re-route would be beneficial to the agents.

6.3.3 Research Question 3

Our final research question: **Will the weighting factor reach an equilibrium point at which the agent will no longer make adjustments between routing episodes?**

When simulation parameters were set to allow the agents to adjust their SAW weights, it was found that many agents did not reach an equilibrium point and continued to make changes to their weighting factor. This was observed as the agents were making changes to their routes and at user equilibrium.

We discuss the reasons for this below.

6.3.3.1 Localization of the SAW Weight

The ideal SAW weight is localized to the road segment to which it is applied. Over many simulations, congestion on a road segment may change, depending on the routing decisions of the agents using it. As changes in congestion on one road segment may be different than on another segment, the weighting factor that produces the most accurate estimate of the time to travel any given segment may be different than for another segment.

As the SAW weight used by an agent represents an aggregation of the weight for each road segment on a route, the ideal weight for an agent depends on the route used. When building a route, there is often little difference between fastest paths for the agent – the geography of the map is such that these routes may vary by only a few road segments. Thus, as the agent is learning which weighting factor is best, they will find a value that may differ significantly from the weighting factor used by an agent that is routing in a different part of the city. As such, while there were agents that were observed having the

same weighting factor, often many agents used weights that were unique to their origin and destination.

6.3.3.2 Changes in Weighting Factor at User Equilibrium

Many agents were found to change their weighting factor when at user equilibrium. Often the changes were the largest allowed by the SAW weight step limit, such that an agent might select a weight of 0 in one simulation and then a weight of 1 in the next.

When using a TIS with direct experience the agents will only select a route that is different than that used in the previous routing episode if the route is new to the agent or is significantly faster than route they used previously. As displayed in *Figure 14*, this eventually allows the agents to reach a user equilibrium. It also has the effect of stabilizing the congestion problem, as the agents always use the same road segments in the same order.

The stability of the congestion problem affects the travel time data provided to the agents. As the agents use a combination of short-term and long-term travel time averages to calculate their weighting factor, the longer congestion remains consistent, the more similar the short-term and long-term averages become. This results in there being little difference in estimated route times, regardless of the SAW weight used, and the agent makes changes to the weight value based on smaller differences between the two averages.

As such, we see that stabilization of the congestion problem is more important than selecting the most appropriate SAW weight when an agent is attempting to select the fastest route to their destination.

Chapter 7: Conclusions and Future Work

7.1 Discussion and Conclusions

In this research we have studied the use of a centralized real-time traffic information system (TIS) as a method to improve vehicle routing in a congested city.

We found that the combination of a TIS and direct agent experience allowed our agents to achieve a total route time that is significantly lower than when using either a TIS or direct experience alone. The combined system produced a lower median price of anarchy than either of the other methods, with or without re-routing, while also achieving a lower price of anarchy on the 60th simulation.

The research makes the following contributions:

1) Avoidance of poor route choices while improving a route. Agents using either TIS alone or in combination with direct experience are able to avoid exploring routes that perform poorly on congested roads by using travel time data collected from all agents. In comparison, agents that select routes using only their own previous routing experiences are required to explore many routes that perform poorly to collect enough travel time data to begin selecting routes that are faster on congested roads.

2) Using TIS allows adaptation to changes in congestion. The TIS provides each agent access to up-to-date travel time data for all road segments. As such, an agent using TIS can adapt its route to account for changes in congestion that may occur over time. An agent relying solely upon its own routing experiences does not know that congestion may

have changed on road segments it has not travelled upon recently, causing the agent to miss opportunities to reduce its route time.

3) Applying travel time data from vehicles allows for a simpler routing method. While the use of a centralized system is not unique to this method, the TIS offers advantages over previous research. As the system focuses on providing travel time data for each road segment, agents can simply apply the data to their routes without concern for the additional effects of congestion – it is already a part of the data they are using. This is an improvement over the Route Information Sharing (RIS) method studied by Yamashita et al. (2005), which required a server to estimate the effects of congestion based on routes provided by participating agents, without access to congestion data from non-participating agents.

As well, agents using a TIS can retrieve all travel time data required in one request to the central system, while RIS agents often require multiple calls to the server to develop a faster route. When developing a route, limiting communication with the central system is desirable, as it reduces the wait time for an impatient driver.

4) Rapid convergence upon a user equilibrium. The achievement of user equilibrium is an indicator of the fairness of the routing system as well as the amount of change in congestion that is occurring between routing episodes. When change in congestion is small each agent is able to better determine the fastest route to their destination and the total route time of all agents is reduced.

Levy et al. (2017) found that convergence upon a user equilibrium required approximately 2000 simulated routing episodes when using direct experience alone. While

this shows that user equilibrium is achievable without directly controlling agent routing decisions, it is not practical for the average driver. Our method converged upon a user equilibrium state quickly – often as early as the 26th routing episode – making the use of a TIS with direct experience more practical than relying upon direct experience alone.

7.2 Limitations of this Research

While routing with a TIS and direct experience offers a number of advantages to the driver, we note some of the limitations of this research:

1) Experimentation was limited to fixed origins and destinations. Each agent used the same origin and destination point for all simulations. This allows for a comparison of the different routing parameters, but is a limited representation of real-world driving. While a driver may indeed travel between the same origin and destination at the same time each day – such as travelling from home to work – it is to be expected that there would be variation in driver's routine, particularly on weekends. This may limit the effectiveness of our method to predict congestion and we would not expect the same amount of route time improvement in an actual city that we saw in our simulations.

Additionally, as the agent's SAW weight is a reflection of the congestion encountered on its previous routes, the addition of novel destinations may cause the agent to modify their weight such that it is no longer best suited for its previous origin and destination. However, it is possible that the agents may account for this issue by maintaining different weight values for each origin and destination combinations.

2) Higher traffic density may limit the effectiveness of a TIS. Our experiments were limited to 100 vehicles in 25 city blocks. While this allowed us to determine how the agents would route with options that included a variety of congested and uncongested roads, many cities are highly congested on most roads (Mandayam and Prabhakar, 2014; Bazzan, 2009). The effectiveness of a TIS may be limited in this case, as an agent may not find a good route to their destination, regardless of the travel time data available to them.

As well, using a TIS may have a limited ability to modify congestion. In our experiments we showed that the agents could achieve user equilibrium and, thus, the congestion adjusted to all agent decisions and became more predictable. When high levels of congestion are encountered all road segments may be at capacity and any routing choices made by the agents cannot make an improvement.

7.3 Future Work

We have demonstrated that combining a TIS with direct experience allows agents to build routes that are faster than when using either method alone, however, there is further research that may improve upon these results.

7.3.1 Additional Information Sharing

Our research demonstrated that the largest improvements in route times were found when agents used a TIS in combination with direct experience. However, when an agent travels to a novel destination, it lacks this previous experience. If each agent transmits its

updated SAW weight and route travelled to the TIS upon completing its route, other agents may be able to use this data to more quickly improve their routes.

While this may be of value, including route information in the data sent to agents raises concerns over privacy and the ability to track the movements of other users. Yamashita et al. (2005) note that methods developed for anonymous auctions may be used to anonymize the route data, but this would not guarantee the user's identity could not be determined using origin and destination points. Additional research into ways to anonymize the route data, such as the use of a geofencing approach to remove specific origin and destination locations, is needed to allow for the safe use of this data.

7.3.2 Intentional Travel Time Data Modification

As the agents make decisions based partly upon the travel time data that they receive from the TIS, there is an opportunity to affect their decisions by making modifications to the travel times provided to them. If we consider that sending an arbitrarily large travel time to an agent would effectively cause them to avoid a route using the affected road segment, we see a number of potentially useful possibilities.

In the case of road closure due to an accident, temporarily increasing the travel time along the road would cause agents to route around the affected area. This technique may also be useful in reducing the number of vehicles that attempt to use a road that is under construction. As well, in the case of evacuations, vehicles could be encouraged to use some roads but not others by modifying the travel times provided.

While modifying travel times could be useful in managing the flow of vehicles, it has been noted by Wang et al. (2016) that simply closing a road segment will affect

congestion on surrounding road segments. Further research must be done to understand how to best make modifications to travel time data to account for this effect.

References

Auer, P., Cesa-Bianchi, N., Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47, 235-256. https://doi.org/10.1023/A:1013689704352

Bazzan, A.L.C. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *AAMAS*, 18(3), 342-375. https://doi.org/10.1007/s10458-008-9062-9

Bazzan, A.L.C., & Chira, C. (2015). A hybrid evolutionary and multiagent reinforcement learning approach to accelerate the computation of traffic assignment. *International Conference on Autonomous Agents and Multiagent Systems*, Richland, SC, 2015, 1723-1724.

de Oliviera, D., & Bazzan, A.L.C. 2009. Multiagent learning on traffic lights control: Effects of using shared information. In A.L.C. Bazzan & F. Klugl (Eds.), *Multi-Agent Systems for Traffic and Transportation Engineering* (pp. 307-321). Hershey, PA: IGI Global. https://doi.org/ 10.4018/978-1-60566-266-8

Desjardins, C., Laumonier, J., Chaib-draa, B. (2009). Learning agents for collaborative driving. In A.L.C. Bazzan & F. Klugl (Eds.), *Multi-Agent Systems for Traffic and Transportation Engineering* (pp. 240-260). Hershey, PA: IGI Global. https://doi.org/ 10.4018/978-1-60566-266-8

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 269-271.

DLR- Institute of Transportation Systems (2017, May). *SUMO*. http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931 read-41000/

Erev, I., Ert, E., Roth, A.E. (2010). A choice prediction competition for market entry games: An introduction. *Games*, 1, 117-136.

Google. (2017, May). *Google Traffic*. https://www.google.ca/maps/

Hardin, G. (1968). The strategy of the commons. *Science*, 162, 1243-1248.

Hart, P.E., Nillson, N.J., Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.

Hong, G-J., & Cheng, S-T. (2016). Using intelligent vehicle infrastructure integration for reducing congestion in smart city. *Wireless Personal Communication*, 91(2), 861-883. https://doi.org/10.1007/s1 1277-016-3501-8

Kutz, M. (2011). Traffic congestion. In *Handbook of Transportation Engineering, Vol 2: Applications and Technologies* (2nd ed., ch. 3.2). New York: McGraw-Hill.

Levy, N., Klein, I., Ben-Elia, E. (2017). Emergence of cooperation and a fair system optimum in road networks: A game-theoretic and agent-based modelling approach. *Research in Transportation Economics*, 1-11. https://doi.org/10.1016/j.retrec.2017.09.010

Levy, N., Ben-Elia, E. (2016). Emergence of system optimum: A fair and altruistic agent-based route-choice model. *5th International Workshop on Agent-based Mobility, Traffic and Trans. Models, Methods And Applications*, 928-933. https://doi.org/10.1016/j.procs.2016.0.187

Mandayam, C.V., & Prabhakar, B. (2014). Traffic congestion: Models, costs and optimal transport. *SIGMETRICS'14*, Austin, Texas, USA. https://doi.org/10.1145/2591971.2592014

Roughgarden, T. (2001). Stackelberg scheduling strategies. *STOC'01*, Crete, Greece, 2001, 104-113.

Shoham, Y., & Leyton-Brown, K. (2009). Congestion games. In *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (Rev. 1.1, pp. 174-184). Cambridge: Cambridge Univ Press.

Shoham, Y., & Leyton-Brown, K. (2009). Self-interested agents. In *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (Rev. 1.1, pp. 47-48). Cambridge: Cambridge Univ Press.

Shoham, Y., & Leyton-Brown, K. (2009). Selfish routing and the price of anarchy. In *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (Rev. 1.1, pp. 180-181). Cambridge: Cambridge Univ Press.

Shoham, Y., & Leyton-Brown, K. (2009). Strategies in normal-form games. In *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (Rev. 1.1, pp. 59-60). Cambridge: Cambridge Univ Press.

Shoham, Y., & Leyton-Brown, K. (2009). Teams of selfish agents: An introduction to coalitional game theory. In *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* (Rev. 1.1, pp. 383-408). Cambridge: Cambridge Univ Press.

Sutton, R.S., & Barto, A.G. (2014). Multi-arm bandits. In *Reinforcement Learning: An Introduction* (2nd Ed., pp. 27-46). Cambridge: Cambridge Univ Press.

Sutton, R.S., & Barto, A.G. (2014). Upper-confidence-bound action selection. In *Reinforcement Learning: An Introduction* (2nd Ed., pp. 37-38). Cambridge: Cambridge Univ Press.

Thathachar, M.A.L., & Sastry, P.S. (1985). A new approach to the design of reinforcement schemes for learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15, 168-175.

Tumer, K., & Proper, S. (2013). Coordinating actions in congestion games: Impact of top-down and bottom-up utilities. *AAMAS*, 27(3), 419-443. https://doi.org/10.1007/s10458-012-9211-z

Wang, S., Djahel, S., Zhang, Z., McManis, J. (2016). Next road rerouting: A multiagent system for mitigating unexpected urban traffic congestion. *IEEE Transactions of International Transportation Systems*, 17(10), 2888-2899. https://doi.org/10.1109/TITS.2016.2531425

Wardrop, J. (1952). Some theoretical aspects of road traffic research communication networks. *Proceedings of the Institute of Civil Engineering, Part 2*, 325-378.

Waze. (2017, May). *Waze*. https://www.waze.com/

Yamashita, T., Izumi, K., Kurumatani, K., Nakashima, H. (2005). Smooth traffic flow with a cooperative car navigation system. *AAMAS 05*, The Netherlands, 478-485. https://doi.org/10.1145/1082473.1082546