MULTIMODAL TRAFFIC MODELING IN RIYADH, SAUDI ARABIA:

A COEVOLUTIONARY, BEST CASE ANALYSIS

by

Abdullah N. Binthunaiyan

APPROVED BY SUPERVISORY COMMITTEE:

____
Dr. Denis J. Dean, Chair

____
Dr. Brian J. L. Berry

____
Dr. Yongwan Chun

____
Dr. Anthony R. Cummings

MULTIMODAL TRAFFIC MODELING IN RIYADH, SAUDI ARABIA:
A COEVOLUTIONARY, BEST CASE ANALYSIS

by

Abdullah N. Binthunaiyan, BS, MS

DISSERTATION

Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN

GEOSPATIAL INFORMATION SCIENCES

THE UNIVERSITY OF TEXAS AT DALLAS

May 2018

ACKNOWLEDGMENTS

Many have guided, supported, and inspired me to do this research. My advisor and committee chair, Dr. Denis J. Dean, worked very hard with me on this research and provided brilliant suggestions and solutions for every obstacle I have faced. Dr. Dean is the dean for the School of Economic, Political and Policy Sciences and he teaches and works on very important research and scientific projects. Despite all his commitments, he was available whenever I asked for help. He worked with me a lot to improve the methodology, techniques, and even the text. I also acknowledge and thank my committee members, Dr. Brian Berry and Dr. Yongwan Chun, and Dr. Anthony Cummings who provided me with useful feedback on modeling issues and on the study as a whole.

I also would like to thank all the other GISC faculty members: Dr. Michael Tiefelsdorf, Dr. Daniel Griffith Dr. Fang Qiu, Dr. Bryan Chastain. In addition, many thanks to my colleagues and classmates who gave me support and feedback. I find it important to mention and acknowledge the wonderful people whom I don't personally know, but they provided me with tremendous support and help; especially Dr. Kai Nagel and Marcel Rieser from the MATSim blog and Q&A community. In addition, I would like to thank my friend, Hassan Ahmad, for his advice on Java coding.

I would not have been able to pursue my PhD studies without the support of the NIC top management, especially the former NIC's General Manager and now the MOI's Assistant to the Minister for IT; thus, I would like to present my deepest appreciation to HHP Dr. Bandr AlSaud as well as the leadership of my country who support and sponsor education at all levels. Data used in this study was provided by the ArRiyadh Development Authority (ADA); many thanks to Eng. Adel Alfassam and Eng. Abdulrahman Alqahtani from ADA for their support.

Last but not least, I would like to thank my mother for her love, support, and prayers. I also would like to thank my wife and daughters for being understanding and patient.

May 2018

MULTIMODAL TRAFFIC MODELING IN RIYADH, SAUDI ARABIA:

A COEVOLUTIONARY, BEST CASE ANALYSIS

Abdullah N. Binthunaiyan, PhD
The University of Texas at Dallas, 2018

Supervising Professor: Denis J. Dean

Navigational systems lack an option that allow users to choose a route from source to destination regardless of travel modes as long as the time is minimized. This implies that navigational systems might not search all feasible routes for minimizing the objective function in this type of routing problem. In this study, we developed two shortest path models; one is time-dependent "car" mode shortest path model and the other is time-dependent "mixed" mode shortest path model. Both models respect capacities of the modes, whether it is the street network or the train vehicles, when recommending the shortest path. Both models respect the queueing phenomenon by which users who make first arrival to a node or a link get to depart before those who make a latter arrival.

We tested both models using a real transportation network. We used a multiagent transportation simulation software called MATSim to simulate traffic on this network and test our shortest path models. Towards this endeavor, we showed a methodology of how to convert and model multimodal transportation datasets from their original shapefile format to a MATSim network format. Then we tested both models to find out more insight about the computational complexity and resulting trips' durations of the mixed-mode route guidance application compared to the car mode.

We found that it is possible to construct a route-finding system that allows for a mixed-modes travel option (with capacity and scheduling constraints) and takes into account the temporal aspect of travel network behaviors, while limiting the computational requirements to a level at which the system could be implemented in the real-world urban environment of a portable navigational device or a cell phone. We also found that such system can result in significant time savings for users who use it. In addition, we presented the critical role of network structure and capacities, or traffic loads, in calculating the shortest path. The more traffic jams in a city, the stronger the case of using a mixed-mode route guidance application becomes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Modeling transportation networks presents an attractive topic for researchers from many fields. Transportation networks are characterized by and composed of many elements. Among their components are the geographic arrangement of the network and the interactions between its agents (infrastructure, users, and external agents such as the weather or even navigational systems providers). These components and interactions determine the method and speed at which people navigate through the transportation network. Scientists from different disciplines such as Geographic Information Science (GISc), networks and graph theory, queuing theory, operations research, and game theory have all contributed to the literature on these topics.

Minimizing the cost of travel has long been an area of interest among researchers. One recurring theme within this area is the shortest-path problem. The invention of portable navigational devices has allowed users of transportation networks to address the shortest-path problem in real time. The wide adoption of smart phones and the ubiquitous nature of the Internet have caused a paradigm shift in the way these relatively new navigational systems are used. New mobile applications for navigation have been developed and are widely and freely available on smart phones. However, although the transportation network in most urbanized areas is composed of multi-modal networks (e.g., streets, rails, buses), to the best of the author’s knowledge, most of the widely used portable navigational devices and navigation applications do not allow the user to optimize his/her route by using multiple modes of travel.

Optimization problems are frequently addressed by employing a search mechanism that finds the best solution amongst a set of feasible ones. In the shortest-path problem, the solution is usually the path that minimizes an objective function that calculates distance, time, or some other negative impact of travel. In a multi-modal situation, the optimal solution to this problem could be a path that consists of a private mode of travel for part of the trip and a public one in another part.

The proposed research aims to create a model that minimizes travel time while allowing for multimodal travel to reach a destination. The practicality of this model in terms of its computational complexity will be tested. Important restrictions and complexities that could affect such a model will also be addressed.

1.2 Motivation

Recent surveys and reports indicate changes in employment and population patterns (Rapino and Fields 2013). These patterns suggest that urban populations will increase and average commuting times will also increase in most urban areas. With increased commuting times, there is an increasing need to enhance the capabilities of navigational models and algorithms. One obvious enhancement is to design models to find optimal modes of travel, that is, to search the whole multimodal transportation network.

For this multimodal search model to be usable, it must keep track of the capacity and scheduling constraints of public transportation systems. In addition, it should prioritize routing based on the time of inquiry. That is, users who are routed at time *t* will be in the service queue of either mode before users who are routed at a later time; for example, *t*+1. For instance, it would be pointless to route a user through a rail station where the incoming vehicle is already filled with passengers. Users need to be informed about the time cost of going through a rail station, which includes time spent waiting for an accessible rail vehicle to serve him/her. Similarly, a routing algorithm user would not prefer to be routed through roads that are jammed with traffic if the rail system will take him/her to the destination in a shorter time.

New within-cities travel behaviors further extend the justification for considering multimodal networks in navigational apps. There is an increased demand for shared ridership and public transit. For instance, Uber, a new economy taxi company, recently announced that it provides rides to one million people per day (Uber 2014). We speculate that many of these passengers would use mixed modes of travel (Uber and rail rides) if they could arrive at their destinations in a shorter time or at a lower monetary cost. In fact, a recent study by the American Public Transportation Association found that the more people use Uber or similar services, "the more likely they are to use public transit" (American Public Transportation Association 2016, 3). Based on such trends, I speculate that more private car users will use public transit to fulfill part of their commuting trips, especially in crowded cities.

1.3 Study Objectives

This research seeks to contribute to the domain of transportation routing and navigation models by innovating a shortest-path algorithm that is both multimodal and capacity aware. The algorithm will also address the flow characteristic of multimodal transportation systems by including the queueing effect at the time of arrival. In order to make this model suitable for implementation on hand-held computing devices, the proposed model combines methods from knowledge domains such as queueing theory, GISc, operations research, and game theory.

The proposed model will also be designed to address the time-dependent nature of shortest-path calculations within the context of multi-modal transportation systems. Thus, the new model will demonstrate capacity-, queue-, mode-, and schedule-aware qualities.[1] As noted by Goodchild (2000) "few efforts have been made to create databases that com-bine modes, by representing both road and rail networks and their interconnections, for example, but these would be essential for multimodal routing." Therefore, we will create a data structure to model the transportation network's interconnection in a way that can be efficiently queried by our proposed shortest path model.

The time-saving benefits of expanding the solution space to include more than one mode of transportation system will be examined as well as the characteristics of the resulting shortest paths. To analyze these time savings benefits, our model will be designed in a way that addresses a best-case scenario. In this scenario, the traveler, could pick up a car at both ends of a rail trip towards his/her destination. This case is usually rare, but it actually exists for those travelers who

[1] These are the boundaries within which the proposed model works. While capacity for the street network refers to the actual flow of vehicles, a street could allow at the street's speed limit; in the case of the rail system, capacity refers to the limit load of passengers a rail vehicle could accommodate during a rail trip. Being queue-aware means that the algorithm can recognize that the number of a rail system's users waiting for a rail to come and pick them up from a rail station will be served in the temporal order they arrive to the rail station. The model is mode-aware in the sense that it can classify the overall transportation network based on the mode of service. The schedule-aware model refers to the rail system schedule of operation.

commute to work and do not own cars, for example. Analyzing the proposed model given such best-case assumption is important because it gives an upper limit to possible time savings of using such model over a car-mode only route guidance application.

Another objective of this research is to examine how the produced results are affected by the scheduling constraints of the public transportation system. Scheduling of public transportation is an important network element. Because cities share scheduling data, this element has begun to appear in public transit navigational systems (Google, Inc. 2016). This element has special importance in multimodal shortest-path algorithms, because it facilitates realistic searching for the time-dependent shortest route.

A comparison between (1) a time-dependent, capacity-aware, and car-mode only shortest-path model and (2) the proposed time-dependent, capacity-, multimode-, and schedule-aware, shortest-path model will address both how computationally complex the proposed model is. Quantifying the difference between the shortest-path algorithms represented in (1) and (2) will help in determining whether the algorithm specified with the constraints in (2) will suit the computational capacity of hand-held devices like smart phones. The suitability of such an algorithm to be implemented using other computational resources, such as cloud computing, will be addressed as well.

In addition, this study aims to document a method of preparing and combining multimodal transportation data sets – usually not found in the same place for most cities – to be used in traffic simulation experiments. This is particularly important for ArcGIS users who are not familiar with other open-source spatial data-editing tools like Java Editor for Open Street Maps, which is used as the main spatial data platform by traffic simulation programs like AnyLogic or MATSim. It is equivalently important for those users who want to study areas where street data is not accurate in the open street maps website but can be found at high quality in the ArcGIS-based format.The next section discusses the research questions that are formulated based on the study’s motivations and objectives.

1.4 Research Questions

The spatial problem under investigation is to study the feasibility of optimizing the shortest path between a source and a destination in the presence of three transportation modes that could be used together or separately, depending on the schedule and capacity of all modes. These three modes of travel are namely the private cars on the road network and the public transit that operates on rail networks, as well as walking pathways that connects both previous modes. The addition of capacity and scheduling constraints for the rail network might cause a significant increase in the model's runtime. Testing for such an environment that simulates traffic on a real transportation network will allow to further test the benefits of adding such complexity to the widely used routing systems. In order to assess the proposed model, further insights need to be gained regarding the following research questions:

1- **Does the proposed best-case model result in closer-to-optimal routes compared to a best case, standard, time-dependent, car-mode, capacity-aware, shortest-path, Dijkstra-based model?**

2- **From a run-time perspective, how does the proposed algorithm differ from the standard (Dijkstra-based) capacity -, and car-mode, time-dependent shortest-path model?**

The answers to the above questions will be used to address the more general, fundamental question of this research, which is thus:

Is it possible to construct a route-finding system that allows for a multimodal travel option (with capacity and scheduling constraints) that takes into account the temporal aspect of travel network behaviors, while limiting the computational requirements to a level at which the system could be implemented in the real-world urban environment of a portable navigational device or a cell phone?

This question will be analyzed and addressed later in the results section. The next section discusses different perspectives of the of the routing problem as found in our literature review.

CHAPTER 2

LITERATURE REVIEW

Different tools and analysis procedures have been utilized in the study of transportation. For example, graph theory is a methodology for analyzing networks mostly within a mathematical framework (Bondy and Murty 1976) and is widely used in the study of transportation geography. In addition, adjacency and connectivity matrices are powerful tools for modeling connectivity in transportation geography (McNally 2007). Similarly, different tools and theories that were essentially developed in different fields have been employed to analyze routing problems in traffic networks. The next subsections present brief discussions on the relevant topics from those fields for this research.

2.1 Transportation Geography

Transportation is an economic activity in which most people engage. The main purpose of transportation is to overcome the space that separates us. In this context, transportation is a field of application rather than a science. In the geographic context, transportation geography is a sub-field of geography that studies the movement of people, freight, and information (Rodrigue, Comtois and Slack 2006). There are four main components to transportation: mode, infrastructure, networks, and flows. These components of transportation are usually analyzed individually or together to provide insights into the transportation phenomena of interest.

Most studies of the shortest-path problem in transportation networks are focused on the flow and mode components of transportation. Those two components have been addressed more directly in fields such as game theory (Kaufman, Jason and Smith 1998), queueing theory (Xu, et al. 2014), and optimization algorithms (Bertsekas 1998).

2.1.1 Game Theory

Game theory is a collection of analytical tools designed to formulate and facilitate the understanding of many situations in which decision makers interact (Osborne and Rubinstein 1994). One of the earliest principles in game theory were Wardrop’s first and second principles of traffic assignment (Correa and Stier-Moses 2010). Later versions of the principles became prominent in game theory literature (as the first and second principles of equilibrium. Wardrop's first principle, known also as "user equilibrium" stresses that routes that are selected by travelers are more efficient than any other empty routes, that they did not select, between the origin and destination points. (Correa and Stier-Moses 2010). It states that "The journey times on all routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route." Therefore, "no driver can unilaterally reduce his/her travel costs by shifting to another route." This traffic flow pattern is referred to as selfish Wardrop equilibrium or user equilibrium. Wardrop's second principle of equilibrium is sometimes referred to as system optimal. It states that "at equilibrium the total journey time is minimized." (Wardrop 1952).

Other researchers have investigated the relationships between game theory and transportation (Hollander and Prashker 2006). Fisk (1984) makes use of Nash and Stackelberg equilibria to draw a connection between game theory and various problems in transportation-systems modelling. He demonstrated that game theory may be used to model and solve transportation problems. Generalizing the notion of Wardrop equilibria, Schulz and Stier-Moses (2003) proposed a solution for network games with uncertain travel times. The players involved are the transportation users who seek to minimize their travel times by choosing a path with the best worst-case travel time. Ivanov et al. (2013) suggested a game between two navigational systems and proved that a Nash Equilibrium explicitly exists in such a game. They did not, however, present the practical details of designing how this game is translated into an algorithm that could be used to find optimal paths. Bhaskar et al. (2009) showed the possibility of multiple solutions, i.e. multiple equally good solutions, in routing games of individual transportation system users.

The applicability of game-theory tools in transportation network studies is still debatable. Most of the analytical methods that prove that a solution exists rely on weak or unrealistic assumptions. However, the amount of research that addresses transportation routing problems from a game-theory perspective is quite large. The notion of equilibrium, which has been proven analytically to exist within the different settings of game theory, is the same idea traffic managers try to achieve to minimize the congestion that occurs in transportation systems.

However, how to encode game theoretic analytical solutions into a multimodal navigational system is not well understood.

In summary, game theory provides an analytical approach that can be used in studying different traffic situations. However, these models cannot capture many traffic situations and patterns, as they rely on very constrained assumptions. Most game theory routing results are based on sets of assumptions about the users, and more importantly about the network that are sometimes far from reality. While the objective of using game theory models in transportation is to achieve equilibrium in traffic assignments and thus increase overall mobility, the same notion of equilibrium could be achieved by viewing the problem in a different setting[2] that might be more efficient to encode in a navigational system.

2.1.2 Queuing Theory Applications in Transportation

The mathematical theory that investigates how people wait in traffic congestion is referred to as queuing theory (Kleinrock 1975). A quick look at the transportation patterns within an urban area reveals that cars and people form queues as part of their commuting activities. Queues are formed in streets when the rate of arrival for cars is greater than the capacity that these roads can handle. Queues of people are formed in the case of rail stations when people's demand for rail service exceeds the capacity and/or schedule specifications of the rail system at a given station. The proposed model could be envisioned as a set of interdependent queues.

Many queueing models depict different traffic situations. For example, delays resulting from queues at highway entrance ramps differ from the delays resulting from traffic light stops at intersections. Cascetta (2009) described deterministic and stochastic queueing models for different parts of street networks. Application of these queueing models to navigation systems that deal with real, and usually large, transportation systems is computationally complex. These models must integrate information about variables such as traffic flow (i.e. number of cars per unit distance per unit time), drivers' behavior, link (route) choice, type of vehicles in the traffic,

[2] The natural computing field of study known as co-evolutionary algorithms is used to model selfish drivers in the traffic simulation program used in this research.

type of the trip[3], and the type of street segment (highway, ramp, etc.); such integration is both conceptually and computationally complex.

The key here is to understand that the appropriate mathematical models that govern the rail system's operation and the traffic assignment through this mode within the capacities constraints of its vehicles are queue-based ones (Xu, et al. 2014). Furthermore, the queuing concept that regulates the routing of users in the rail is a finite customer population, with a single server. The server here is the rail vehicle that transports users who are in it or waiting at the next stop station to be served or transported. Let's assume that each rail vehicle has a capacity of 500 users[4] and that each stop station has a limited occupancy level of 500 users to wait for the next vehicle. Then, our system could be viewed as demonstrating a finite population in which users are in train vehicles or in queues at the stop stations or somehow arriving at the station.

Like in the road-network systems, there are many models of queues that are formed in rail networks as passengers move from the stairs to enter rail stations to the stairs they use to walk out of the destination's rail station (Chen, et al. 2012). On a normal day, the movement of a passenger within the rail system occurs in different stages at different speeds: walking into a station, waiting for a train, in train moving at the train's speed, and walking out. This stage of travel has been addressed as data model element called "switch point" by Liu but time taken to traverse this network element has not yet been fully modeled (Liu 2010). However, we believe that the most significant stage that needs to be addressed in a multimodal shortest-path model is the time delay until the passenger gets into the rail vehicle after being routed to the rail mode. Combining private and transit routing systems based on a good approximation of such time would result in more accurate time-dependent shortest-path algorithms.

[3] Trip type refers to the goal of making the trip, such as going to work or visiting a friend. Many transportation studies assert that trip type affects the user's driving behavior and route choice. This becomes an important element in traffic assignment.
[4] I chose the number 500 because, in my study area, the rails are designed to have a capacity of 500 persons.

2.1.3 Traffic Flow Theory

Traffic engineering employs many mathematical techniques to model the rate at which traffic flows. Traffic flow models are very popular among transportation specialists as a tool to study transportation phenomena. While these models are not used to explain scheduled operation of rail system, they are widely used to illustrate and explain how traffic bottlenecks form and what delay they cause over the links on which they develop. The part in which the model proposed in this study relates the flow capacities of the street network with the time dependency[5] nature of traffic can be better understood through traffic flow theory models. For example, the time dependency in the proposed model means that, at different times t_0, t_1, t_2, t_3,…, t_k , the shortest path between the same origin and destination points might differ. The reason for this potential difference is that the traffic load on the network changes the time cost of travel.

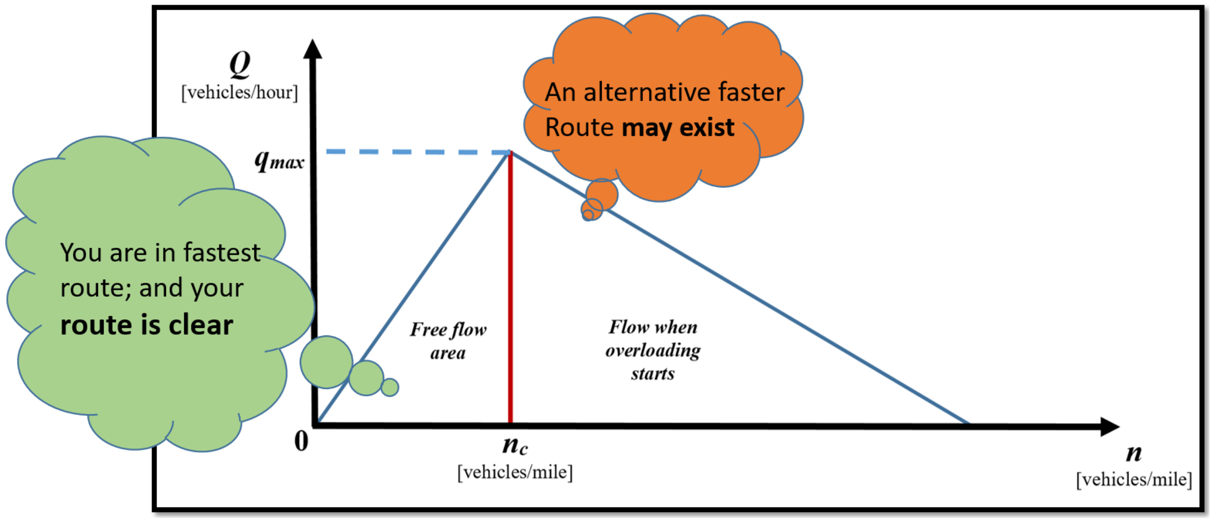
Traffic flow is defined as the number of vehicles that pass through a specific point during a specific time window. The adopted time window in most studies is one hour. Adopting such time window, traffic flow is measured in vehicles per hour. Traffic flux, traffic intensity, traffic volume, and traffic throughput are other terms used in traffic literature to mean the same thing (Maerivoet and Moor 2005). Traffic flow rate on a street link is related to the number of vehicles that occupy a unit length of that link. This number is referred to as the density and is usually measured in cars per mile or kilometer. Figure 2.1 shows a well-known diagram called the fundamental traffic flow diagram.

When the density is at a specific level, called the critical density, all vehicles at the link are assumed to move at the free-flow speed of that link. At this level, the critical density n_c, the flow rate of the link is at its maximum. When the critical density is exceeded, the actual traffic flow rate is forced to decrease. The reason for this decrease is that users on each network link can no longer maintain a safe time of separation (referred to as safe time of impact[6]) between consecutive cars. If the density of a link keeps increasing, there will come a time at which the

[5] This includes scheduling of rail system.

[6] Safe time of impact is the time equivalent of the safe distance at a given traffic flow condition that should be maintained between two consecutive cars if the car in front comes to a sudden stop.

flow will become zero which means that the speed over this link is also zero (Maerivoet and Moor 2005).



Q
[vehicles/hour]
q_{max}
An alternative faster Route **may exist**
You are in fastest route; and your **route is clear**
Free flow area
Flow when overloading starts
0
n_c
[vehicles/mile]
n
[vehicles/mile]

Figure 2.1: Fundamental traffic flow diagram

The importance of such conceptualization of traffic flow is that it could be used to derive a relationship between the speed of a vehicle and the density on a given street link that is overloaded (Badii 2014). Therefore, traffic engineers are usually concerned about calculating the maximum flow q_{max} when they design a new bridge or street with a specific number of lanes and predefined speed limits. The maximum flow parameter is also needed in most traffic simulation models. Sometimes this parameter is referred to as flow capacity.

Shortest path models that are based on the fundamental traffic flow diagram can provide good prediction of optimal routes especially in highways. However, they require an extensive amount of time and effort to be implemented and tested when many queries of these models, just like the route-guidance applications, are considered. That is, these models are not suitable for large-scale scenarios (Thunig and Nagel 2017). Moreover, the core assumption they are based on is violated at network elements like street intersections, roundabouts, or T-junctions. In addition, dynamic traffic assignment is not easily derived, and thus implemented, by utilizing the concepts of the fundamental traffic flow diagram. Therefore, this study employs agent-based simulation techniques that rely more on queuing models to test the proposed multimodal shortest path model.

2.2 Navigational Applications: Review of the Current State

The ability to predict flow changes that take place in the network has influenced many theoretic and industrial advancements in incorporating traffic conditions such as effects of spill back jams and queues into the routing problem. Different techniques and strategies have been employed to estimate the possible speed on a given street when the number of vehicles exceeds its critical density. Google Maps, for example, predicts traffic conditions based on automatic crowd sourcing of the collective speed of cars moving on streets. This crowd sourcing happens by granting the Google Maps app access to the users' location on their smart phones (Google Official Blog 2009).

A large amount of research has been done addressing topics such as intelligent transportation systems, route recommendation applications, and optimal paths. Recently, a common goal of such studies has been to make navigation applications smarter by increasing their ability to predict future traffic conditions and make near-real-time route recommendations based on these predictions. Today, location-enabled mobile phones are also used to enhance time cost estimations of traveling through many streets in the world (Google Official Blog 2009).

Users of route-guidance applications, such as Google Maps, enjoy shortest-path routing with traffic condition estimations as well as re-route options. However, these applications provide recommendations for the shortest path based on the user's preselected preference of the mode of transportation. In addition, routing options in the major routing applications such as Google Maps and Here do not include an option which the user could choose to use two modes of travel between origin and destination points.

In addition to the route-guidance applications that are designed to serve street network users, route-guidance applications are available that serve users of the rail systems. Such applications are usually referred to as transit navigational systems. These systems are in some cases an extension of the traditional ones used to guide car drivers, or they could be stand-alone applications designed to only serve the users of the rail systems, like the Moovit or Transit apps.

To the best of the author's knowledge, some of the most widely used transit navigational systems (namely Transit, Here, Moovit, and Google Maps Transit) do not recommend routes that incorporate the time effects of the possible queues at rail stations. Approximating the time delay at rail stations in a way similar to that which has been done for the highway ramps or streets intersections would make multimodal shortest-path algorithms more accurate. This is especially possible with the emergence of electronic ticketing kiosks that scan paper tickets and electronic tickets (on mobile apps) before allowing a person to enter the rail station (Trainline 2016). One result of this technology is that the number of individuals waiting for the next train to come is actually known and could be communicated with a queue-aware routing system if it is made available as a mobile application.

Google has recently started Google Maps Transit (Google Inc. 2015), which enables the users of Google Maps to navigate their route based on the public transportation system mode of travel. Data requirements for such options come from collaboration with cities around the world which provide Google their public transportation schedule. Google Maps then enables users to observe public transportation navigation over its mapping platform. However, the user has no option of combining both public and private modes to get the best shortest-path time-dependent travel plan.

Another navigation application called Waze, recently acquired by Google, uses social crowd sourcing to determine traffic conditions and then re-route its users to the best-available routes. This application provides a user-friendly interface for its users and a ranking system to get immediate traffic conditions. However, it does not route or re-route its users to any public transportation mode. Other mobile applications (MxData and Transit) integrate the scheduling of public transportation with new-generation taxi companies like Uber. This feature helps commuters request a taxi after they leave the rail station towards another area that is not close by.

Within cities traveling behavior have changed in a way that "smart phones have become central to travelers' decision-making" (Iacobucci, Hovenkotter and Anbinder 2017, 75). The same authors mentioned in the same paper that "Even Google Maps does not integrate all available modes into a particular set of directions." (Anbinder at. el. 2017, 75). In fact, most popular navigational applications do not allow the typical user who uses his/her private car to choose public transport for a portion of the trip from point A to point B. We argue that such a feature could be developed to benefit private users who could be recommended to weigh their options of continuing their journey in a congested traffic or park in the next subway station and use the rail.

2.3 Computational Capabilities of Smart phones

Large part of this study focusses on how well, from a computational perspective, can a smart phone handle the computational needs of a multimodal route-guidance application. To address such a question, we need to know what is a standard computational power of the smart phones at the time of doing this study. Most, if not all, hand held devices like smart phones or personal navigational devices use a type of processors known in the integrated circuits (IC) and semiconductors industry as low-power or mobile processors. The term System on Chip (SOC) is also used to describe mobile processors because they include different types of processing unites (Central Processing Units and Graphical Processing Units) that perform many computing functions.

There are many mobile processors manufacturers, yet the market of mobile processors used by smart phone is largely dominated by three vendors (The Linley Group 2015). This might be a result of the nature of competition in the smart phones market. In our review of processing capabilities of mobile processors, we found that smart phones like Google Pixel 2, LG V30, Samsung Galaxy S8, or Microsoft Lumia 950 are all powered by octa-core processors from Qualcomm (Qualcomm 2017) (Microsoft 2017). Apple on the other hand, designed its own processor, called A11 Bionic, for its iPhone X that was released in 2017. In the next paragraphs, we will present two examples of how Qualcomm or A11 Bionic processor support the functionality of two of the famous smart phones, Apple iPhone X and Google Pixel 2.

Apple's iPhone X is equipped with a processor called A11 Bionic chip. This phone has a 3 GB of random access memory (RAM) and a storage capacity of up to 256 GB. The company's website states that A11 Bionic processor can perform 600 billion computing operations[7] per second (Apple 2017). We could not get an estimate about how many operations are triggered by a routing query, but Apple stated in its iPhone X webpage that taking a picture, which includes a geo-tagging operation, while adjusting for vibration could consume billions of computing operations per second (Apple 2017). The graphics processing unit (GPU) of the A11 Bionic processor is 30% better than the A10 fusion that Apple used for its iPhone 7 (Apple 2017).

On the other hand, Google Pixel 2 is equipped with a Snapdragon 835 processor made by Qualcomm. The phone has a 4 GB of RAM and up to 128 GB of storage capacity. This processor on this phone has a performance speed of up to 2.34 GHz. This processor is 35% smaller and 25% more power-efficient than the previous model, Snapdragon 821. The snapdragon 835 processor is 25% faster than the 821 model in terms of graphics rendering. Although improvements in graphic rendering capabilities is targeting the high demand for gaming and video streaming applications, it also helps the users of online mapping as well.

Processing power is utilized by a smart phone to meet its core functions like calling, texting, or password check. This processing power is also needed to run the applications added by the users like games and video streaming as well as route-guidance applications. Both smart phones discussed above are capable of streaming ultra-high definition (UHD) videos and can detect facial movements as well many other computationally-complicated tasks. All these tasks can be done with mobile processors that are 10 nano-meter in size in both phones.

As discussed in previous section, route-guidance applications provide very advanced routing functionalities. Both iPhone X or Pixel 2 support location-related functionalities by being compatible with famous global navigational systems, enabling e-compass applications, and providing sensor-based micro location services. These technical specifications support both indoor and outdoor route-guidance applications as well as many other location-based applications. Google's Pixel 2 is enabled with a service in which the user can ask the device about the traffic condition on the route between two places (Google 2017).

[7] Apple X webpage does not state the A11 Bionic processor processing speed in GHz.

In most of these applications, especially the route-guidance, the user usually expects to get a recommended route drawn on his phone's screen within reasonable time. This time is important to our study because it determines what is an acceptable run-time for a route-guidance query. We asked some of the major navigational systems discussed in the previous section about such metric. Out of the companies we asked, we received only one response. The response from the navigational application and services provider was as follows:

> "In most cases the time from sending a route calculation request to receiving a response will be < 1 second. There are many factors that influence the time however:
> •Is the route calculation taking place locally (i.e. on the iPhone) or on a remote routing server
> •In the case where the route calculation is taking place on a remote routing server, the network performance plays a key role
> •The mode of transport (car, pedestrian, public transit etc.)
> •In the case of car, is traffic being taken into account
> •The type of route requested (fastest v's shortest)
> •The complexity of the route (e.g. mostly rural/highway, or dense urban environment)"

The comment above from one of the industry-leading navigational applications provider illustrates that the run-time of a route-guidance query could be impacted by the network topology, the traffic loads on the network, the mode of travel, the time-dependency nature of the requested route, and whether the calculations are done on the smart phone itself or on a remote server where a network connection will be needed. As mention earlier, we have not seen any route-guidance application that enables the use of mixed modes of travel in the searching for the fastest route between two locations. This study is an effort to investigate if the computational complexity of such functionality is a major barrier for it to be added to a smart phone-compatible navigational application. The question is whether the computational capabilities that can be encompassed within a hand-held devise such as a smart phone could allow for such functionality.

In 1965, Gordon Moore[8] observed that the number of transistors on a microchip doubles every 18 months (intel 2017). Whether Moore's observation continues to be true in the coming years or slow in its pace remains a big question. However, we have noticed that many of the smart phones have not changed, significantly, in dimensions or weight for the past few years

[8] He is intel cofounder and his observation is also known as Moore's Law

although the computational power and the size of the microprocessors powering them have improved rapidly. Thus, we expect the dimensions to remain about the same in the near future, mainly because wide screens are needed to enhance the user-experience of many popular applications. The improvements in processing power, which are expected to continue, will either allow reduction of prices or allow for supporting more functionalities by these phones. One of the functionalities that could be made available due to improvements in processing power might be multimodal route-guidance.

CHAPTER 3

METHODOLOGY

In order to conduct a valid test of the computational complexity of the proposed model, a real street network will be used, as suggested by Noon and Zhan (1998). Most studies that test shortest-path algorithms use randomly generated networks. However, real street networks might have different characteristics that affect the computational complexity of the algorithms. For this reason, the network used in this study for testing purposes is a real multimodal transportation network. Transportation datasets for the city of Riyadh, Saudi Arabia, are used for the purpose of testing the research questions.

In addition, simulation techniques are used to compare how users of the proposed capacity-, mode-, schedule-, and time-dependent shortest-path model will be routed through the multimodal transportation network against users who use a single-mode, time-dependent shortest-path model. The problem addressed by this study involves decision making at the individual level, since every user is assumed to be using a route-guidance single-mode or multimodal application. Both scenarios involve interaction between transportation network users and continuous competition over network resources. Simulation-based techniques help in studying our models without oversimplifying the assumption related to the environment in which they are designed to operate. However, simulation techniques require very advanced computational power.

The simulation technique used in this study could be implemented using many agent-based modeling programs. An open-source, multi-agent traffic simulation software known as MATSim[9] was selected for use in this study. MATSim is considered as the closest way of employing the Wardrop first and second equilibrium principles. In addition, it is flexible and dynamic, since it is open source. Furthermore, it allows the user to specify the shortest-path algorithm to be used by the agents as a base model for their movement on the network, whereas

[9] MATSim stands for Multi Agent Transportation Simulation (MATSim 2017). This software employs co-evolutionary algorithms in traffic assignment that approximate the behavior of selfish drivers.

other reviewed packages restrict the modeler to use only one built-in shortest-path algorithm. In addition, MATSim has a data input design structure that allows elastic input, which in turn makes it capable of handling different traffic scenarios (Gao 2009).

Within MATSim, unimodal shortest path algorithm will best serve as the basis for developing the new multimodal shortest-path model. This consideration needs to be addressed first before setting the parameters and the constraints of the shortest-path algorithms to be discussed and compared in this research. Magzhan and Jani (2013) tested the algorithms by Dijkstra, Floyd-Warshall, and Bellman-Ford and found that they perform relatively the same in terms of run time and that they only produce one solution. Because it is an exact (and arguably the most often used) shortest-path algorithm, this research employed the Dijkstra algorithm as a base model for both single-mode and multimodal shortest-path scenarios. The adjustments necessary to make it a time-dependent as well as a capacity-, mode-, and schedule-aware model will be explained in the reminder of this chapter. The next section discusses study area and data used.

3.1 Study Area

Analysis of the research question could be performed with any multimodal (rail and road) urban transportation network. We decided to conduct this study using Riyadh city transportation datasets. These datasets have been provided by the ArRiyadh Development Authority (ADA). The capacity constraint for the rail vehicles is given by the ADA: 500 riders per rail vehicle, with an additional standing capacity of 50. The availability of other attributes, like the number of lanes for each street segment and speed limits, enhances the reality of the network. Figures 3.1 and 3.2 present the street network and rail network datasets, respectively.

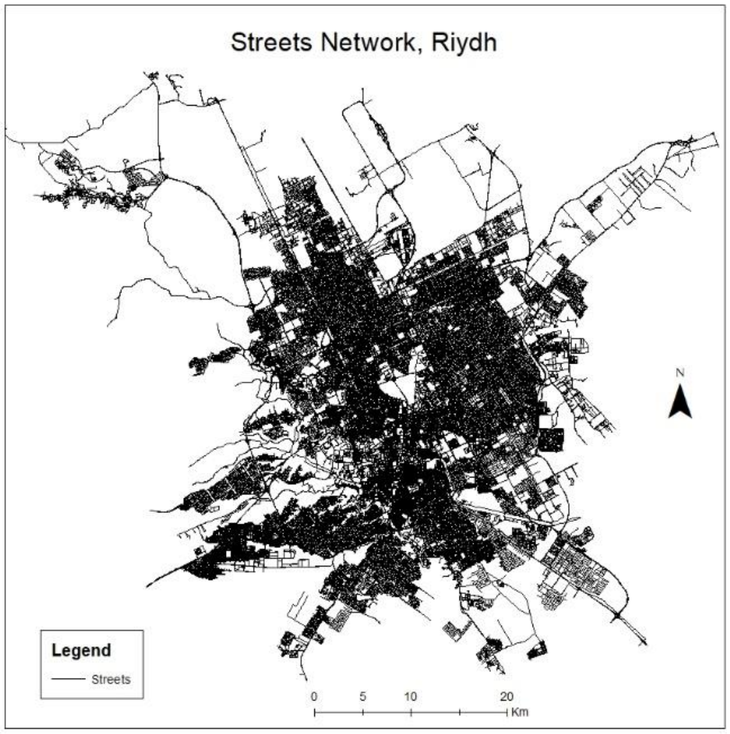


Figure 3.1: All major and residential streets of Riyadh city

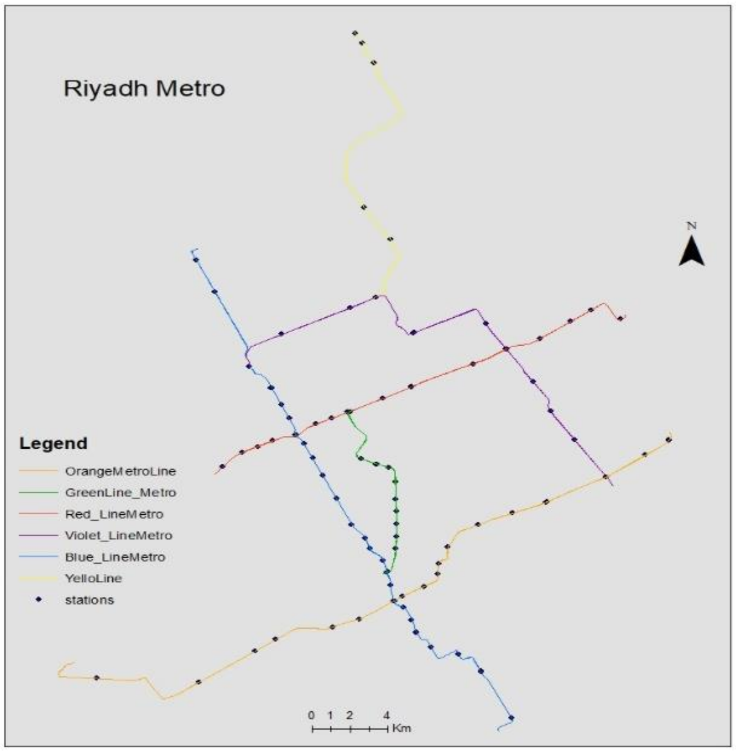


Figure 3.2: Layout of Riyadh City’s Rail Network System

For testing purposes, we decided to use part of the city’s transportation network to conduct our study. The selected portion contains streets and rail lines in business and residential areas. This choice excludes areas like the international airport and the south and southwestern parts of the city as well as some peripheral areas that are still being developed. In addition, all streets that are classified as minor were excluded. This was done to speed up the data preparation and to follow the common practice of starting a project with manageable scope and requirements, then scaling up as we learn more. Figure 3.3 shows the Riyadh street network after minor residential road removal, and Figure 3.4 shows the transportation network bounded by the study area’s boundary line.

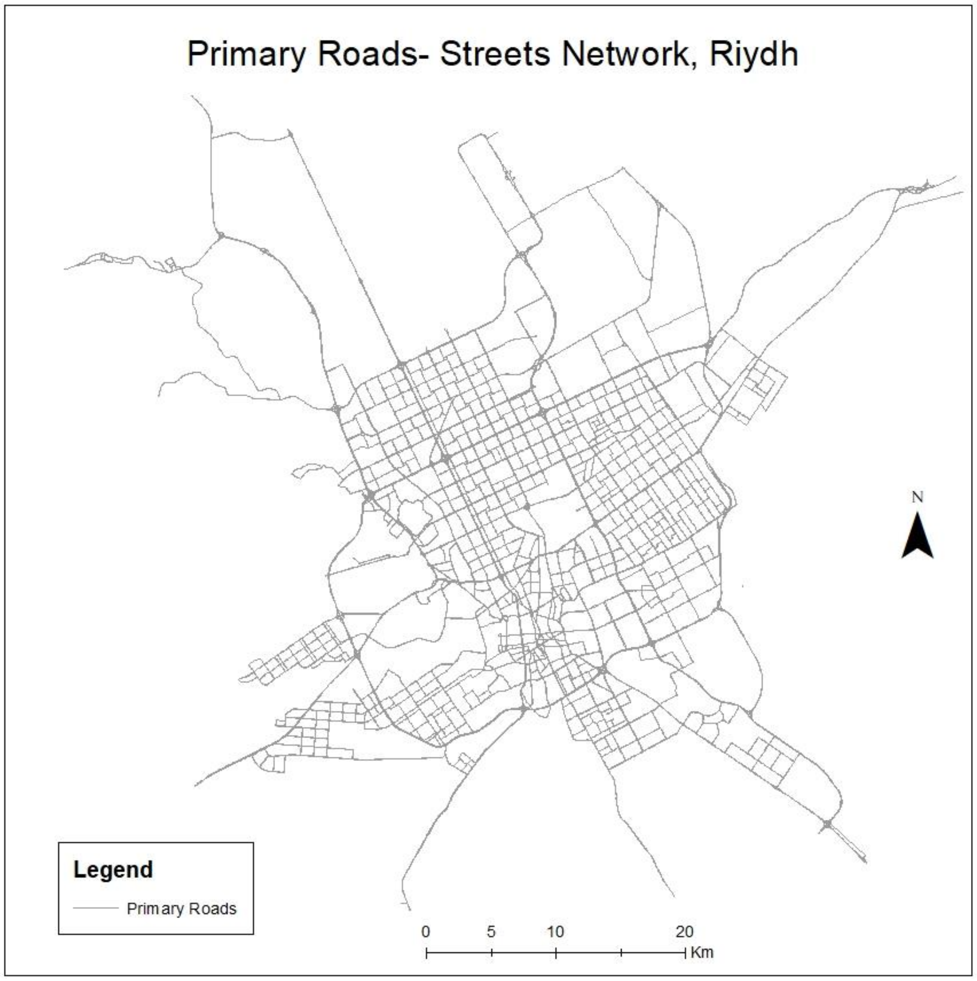


Figure 3.3: Primary streets in Riyadh city

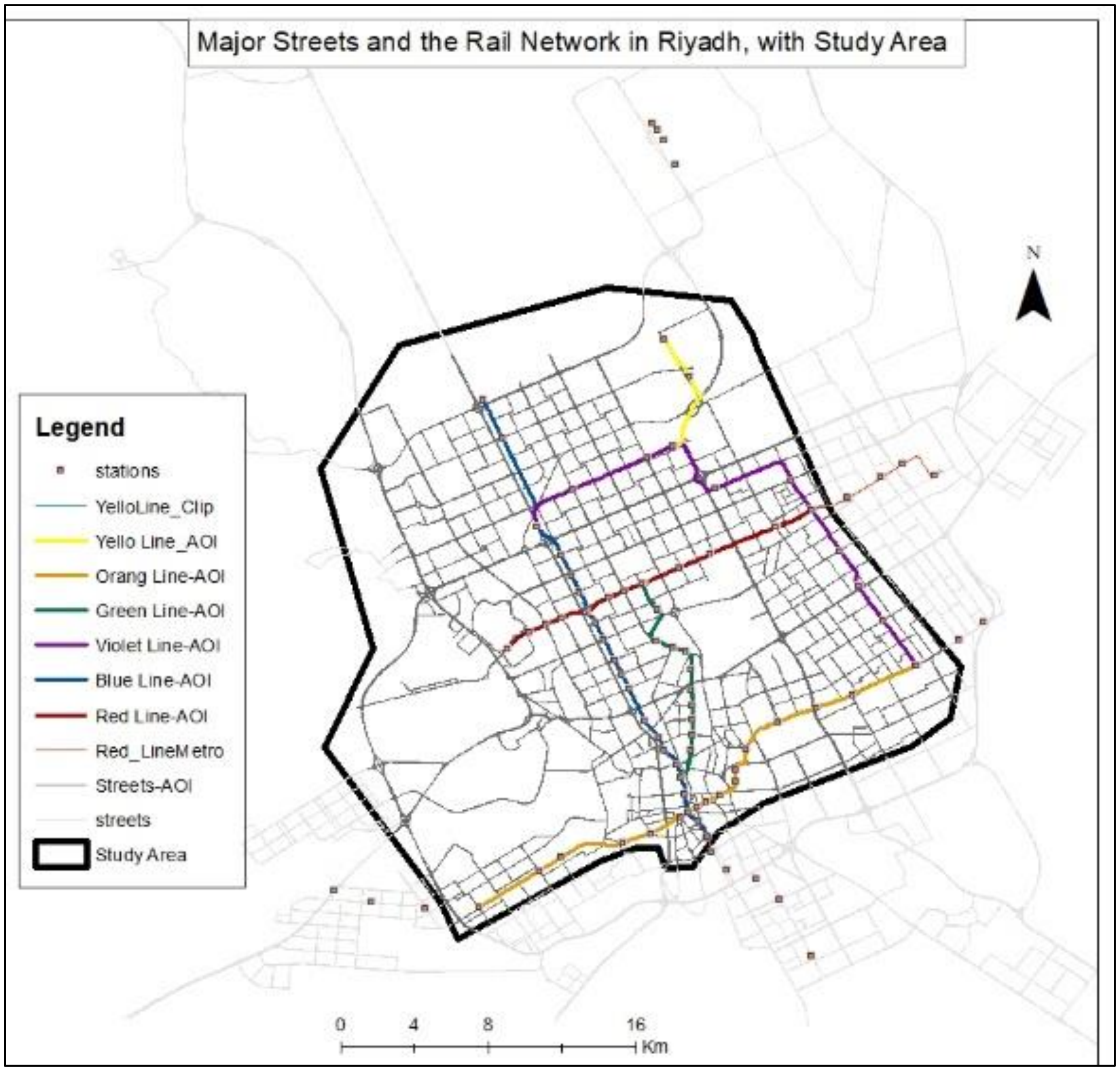


Figure 3.4: Major streets of Riyadh city as well as the rail system, with the study area represented by the bold black line

After specifying the study area, we needed to select points of origins and destinations that were used to simulate trips guided by the shortest-path models described earlier. Dean et al. (2015) proposed a method for selecting the locations of pairs of origins and destinations to test optimal route finding algorithms across a landscape that contains high-cost (to travel through) features. In their experiment, the researchers selected pairs of origins and targets that were as far apart as possible within their study area. This selection criterion ensured that the surface between the source and destination contained all of the costs that their proposed algorithm would minimize.

In this research, we decided to select 30 pairs of origin and destination points to test the models discussed earlier. While the origin points were randomly selected within downtown business districts, destination points were semi-randomly selected in a way that ensured that they fall close to both the major road travel corridors of the city and the train stations. In addition, the destination points were selected from the residential areas at different ranges of distance from the city center. This selection design ensured that the shortest path models were tested over different ranges of distances between origins and destinations and also different spatial proximities to the rail network. It also ensures that we can realistically test the computational complexities of both finding the fastest route and recommending an alternative route after the traffic level changes.

Finally, we decided to exclude the rail system's yellow line, because it is used mainly to transport users from the far northeast side to the airport, an area that is not needed for commuting between the origin and destination locations. Figure 3.5 shows the points of origin and destination locations within our study area.

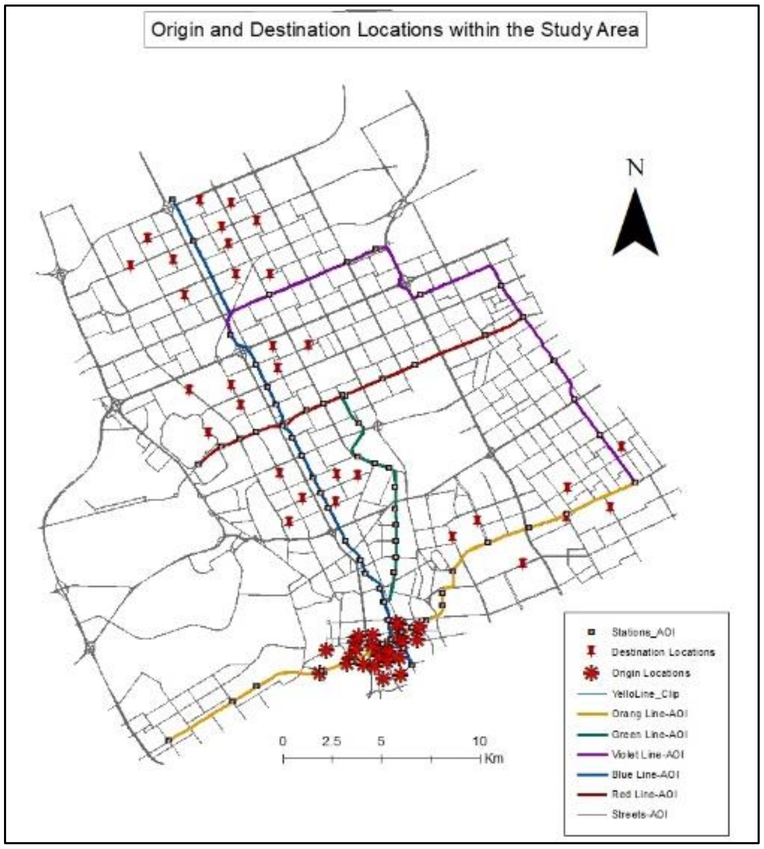


Figure 3.5: Origin and destination locations within the study area

After clipping transportation datasets to cover only the study area and specifying locations for origins and destinations, we further created a separate layer dedicated for walking mode, in which users walk to and from train mode and car mode. Initially, this data set was not provided by the ArRiyadh Development Authority (ADA), because it did not exist. Therefore, we needed to create this data because it represents a logical linkage between the car and train modes of travel. The next subsections provide a deeper look into the attributes of the transportation datasets of our study area including the one we created for walking mode.

3.1.1 Street data set

The first element of the multimodal transportation network that we used to conduct this research is the street data set. We assumed that users would use primary streets and would tend to avoid small residential streets on their trips. By adopting this assumption, we eliminated the street links that are classified as minor residential. This choice significantly reduced the size of the network from over 217,000 links to about 36,000 for the whole city.

The road network that we obtained from ADA contained many attributes. The most important attributes for our purpose are the ones defining the speed limit over the street link, the number of lanes of that link, and whether the link is going one way or if it could be used for travel in both directions. We deleted most of the unnecessary attributes from the street shape file. A critical attribute that was missing from the ADA data but needed for the simulation was the mode of travel over any link in the multimodal network. The street data we obtained did not include this data, so, we added an attribute field and named it “mode” and set its value to “car” over all street links.

3.1.2 Train mode (links and stations) data sets

The train network is composed of 6 lines and 85 stations. We eliminated the yellow line and some other train links and stations that are out of the study area. The attributes that we used from this dataset were the line IDs for links that describe the subtype of the rail (known as its color that specifies its spatial, or service, coverage). We added to that dataset an attribute field and named it “mode” and set its value to “train” over all train links. We did the same for the stations feature class, and we only used their X Y location in our modeling.

3.1.3 Walkways data set

As we mentioned earlier, walking links that connect the car and train travel networks needed to be created. Two sets of walking pathways were created, and their topological correctness in terms of how they are connected to both modes (streets and train lines) was validated. These linkages were created using the editor tool within ArcGIS 10.5. We benefited from the ESRI multimodal network model (ESRI 2017) to define how the car mode and rail mode were topologically linked by the walking mode. Thus, we created pathways for the walking mode that were composed of two types of links: one link connects streets to nearby train stations, and the other link connects nearby train stations together. Then both walk mode feature classes were combined into one feature class named "walkways" using the Append tool in ArcGIS. Like in the streets links case above, we added a "mode" attribute field and set its value to "walk" over all walking links. Figure 3.6 shows an example of walking mode links.

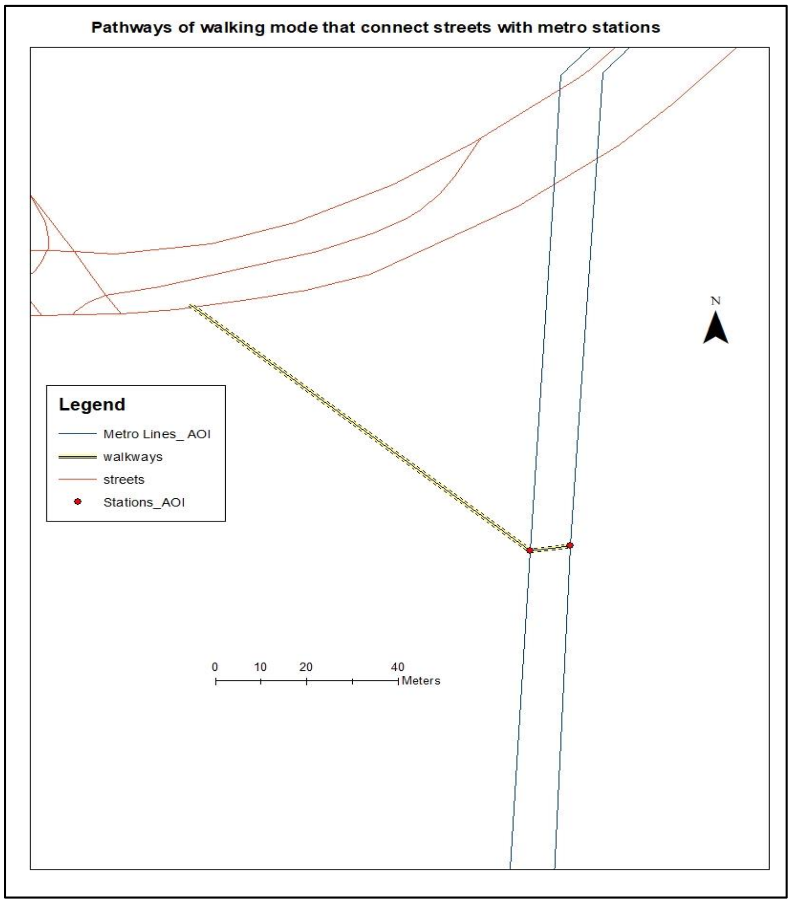

Pathways of walking mode that connect streets with metro stations
N
Legend
Metro Lines_ AOI
walkways
streets
Stations_AOI
0
10
20
40
Meters

Figure 3.6: Walk mode links

3.2 General overview of the Multi-Agent Transport Simulation Software (MATSim)

The optimization problem discussed here involves users or agents competing for the resources they need to travel along their perceived shortest path. Competition for resources along any travel route is governed by queuing modeling, and the choice of traveling along any link is governed by game theoretic models. Algorithms that work as a basis for such settings are known in the computer science literature as coevolutionary algorithms (Coello et. al., 2007). Essentially, coevolutionary algorithms are used to model problems, “for which no function for evaluating potential solutions is present or known. Instead, algorithms rely on the aggregation of outcomes from interactions among evolving entities in order to make selection decisions” (Popovici, et al. 2012). Moreover, coevolutionary simulation environments are suitable for modeling situations in which the best solution for an agent is dependent, in part, on the best solutions for the other agents.

The coevolutionary environment is suitable for developing the proposed model. Fortunately, a software called Multiagent Transportation Simulation (MATSim) uses a coevolutionary algorithm to simulate traffic and search for users’ equilibrium solution over changing traffic loads during a specific time frame (Andreas et.al., 2016). The basic idea of the coevolutionary algorithm as it is employed by MATSim is that agents co-evolve to compete for network resources based on each agent’s predefined utility function in a way that achieves best results for every agent or user. Figure 3.7 how the coevolutionary framework is implemented by MATSim in which agents iteratively and collectively learn to select routes that maximizes their utility.

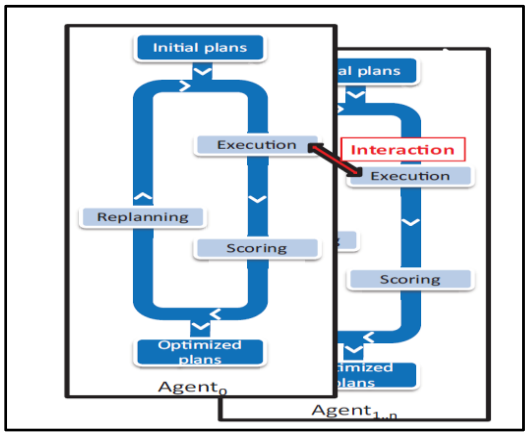


Figure 3.7: MATSim's Coevolutionary Framework (Andreas et.al., 2016)

In section 2.1.3 of this document we mentioned that queue-based models are practical in terms of capturing traffic phenomenon at network components like roundabouts, T-junctions, or traffic lights. MATSim uses a queue-based mobility simulation engine called QSIM that models time dependencies and other first-in first-out (FIFO) characteristics of traffic networks (Axhausen, Horni and Nagel 2016).

It should be noted that MATSim employs spatial queue models for modeling vehicles that enter a link (Thuniga and Nagel 2017). In spatial queue models, as opposed to point queue models, vehicles occupy space. If a link is filled with cars, the following car will not be allowed to enter this link. In high traffic situations, all the following cars that are not allowed to enter a link due to (1) capacity constraints and/or (2) cars' dimensions, will spill back into upstream links.

MATSim is not limited to testing shortest path models for single trips from an origin to a destination location. In fact, MATSim is mostly used to simulate and optimize daily "plans" of trips done by agents. These trips include, for example, going from home to work, to grocery stores, then back to home, and later going to a theater. These trips can all be listed for each agent with their typical durations, start times, and end times. In these situations, MATSim searches for optimized trips' plans for all agents, given their constraints and preferences. A state of equilibrium is reached when "agents cannot further improve their plans unilaterally" (Andreas et. al. 2016).

3.3 Implementing Multi-Agent Traffic Simulation for Testing Shortest Path Models

To implement the capacity-, mode-, and schedule-aware and time-dependent shortest-path model as well as the capacity-aware, car-mode, time-dependent, shortest-path model using MATSim, several data preparation steps need to be taken. The transportation network datasets discussed earlier in sections 3.1.1, 3.1.2, and 3.1.3 are not structured in a way that allow multimodal navigation. Thus, we started by combining these datasets into one network dataset in which modes are assigned appropriately.

The following subsections discuss how we prepared a simulation-ready multimodal transportation network dataset using the previously discussed datasets.

3.3.1 Creating the Multimodal and Car Mode Transportation Networks from Street, Train, and Walkways data sets

The transportation datasets that we have prepared using ArcGIS into the MATSim XML network data structure. There is no tool in ArcGIS that performs this kind of conversion. In the next section, we detail how we overcame this issue.

3.3.2 Using Java Open Street Maps Editor to convert shapefiles into OSM format

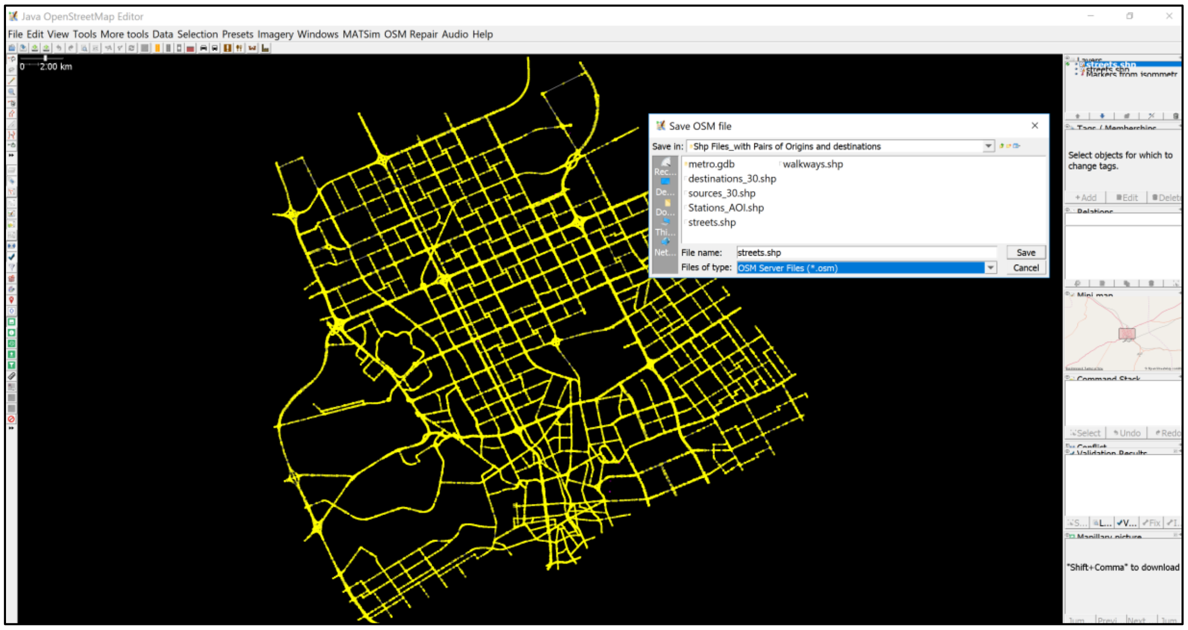
Most traffic simulations performed with MATSim benefit from open-access and high-quality transportation network data freely available at openstreetsmap.org. Unfortunately, open street map data for our study area are of poor quality compared to the data we obtained in shapefile format from ADA. This might be the case for many cities around the world. While there are ways to read open street map networks and convert them into MATSim XML network format, there is no reliable way to do this for ArcGIS network datasets or ESRI shapefiles[10]. The lack of an available conversion tool that converts shapefile format to MATSim XML network data format is a limitation of both ArcGIS and MATSim.

A network data set is defined within ArcGIS by two basic elements: edges and junctions (ESRI 2010). MATSim uses this same basic assumption. However, in MATSim, all attributes of the network elements that are used for traffic flow and routing are contained in one XML file. In the case of ArcGIS, the attributes that are used to solve a routing problem (like cost, restrictions, and speed) are contained in separate line and point source files. Moreover, this information cannot be viewed or manipulated by inspecting the combined network dataset that is created from the multiple source files, even within ArcGIS, unless we inspect these source files individually. To solve this issue, we needed to convert the files that are used to build the network dataset in ArcGIS to open street map (.osm) format, then convert the resulting open street map file into a MATSim-ready XML format.

We found that the most efficient way to accomplish this was to use Java Open Street Map Editor (JOSM) to open the shapefile (.shp) that constitutes a part of our network, whether it is a

[10] Interestingly, we found that MATSim network can be exported to shapefile format (.shp); however, MATSim network editor cannot read shapefile format. We also found that there is no conversion tool that converts shapefiles to open street maps; however, Java Editor for Open Street Map can be used to import any shapefile and then saves it as open street map format (.osm).

line feature class (link) or a point feature class(node), then save that file in an open street map format (.osm). Figure 3.8 shows a screenshot of the study area map being opened and saved in open street map format using JOSM.


Java OpenStreetMap Editor
File Edit View Tools More tools Data Selection Presets Imagery Windows MATSim OSM Repair Audio Help
0 2.00 km
Save OSM file
metro.gdb
walkways.shp
destinations_30.shp
sources_30.shp
Stations_AOI.shp
streets.shp
Save
Cancel
Select objects for which to change tags.
"Shift+Comma" to download

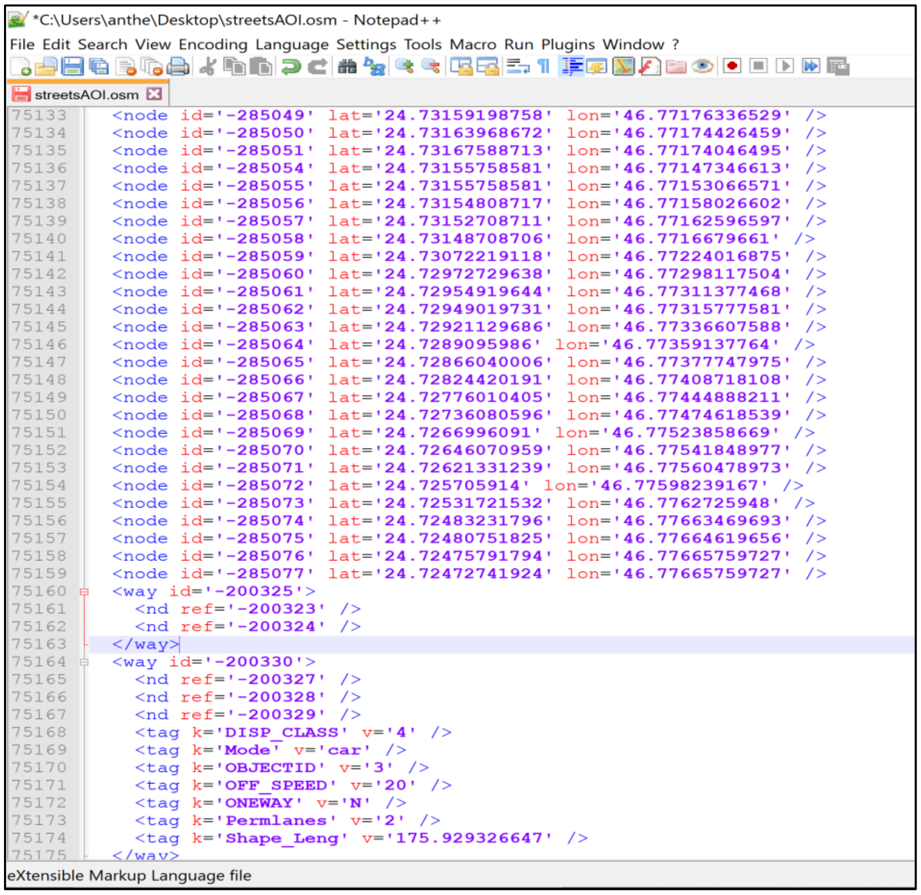
Save in: Shp Files_with Pairs of Origins and destinations

File name: streets.shp

Files of type: OSM Server Files (*.osm)

Figure 3.8: A screen shot for JSOM which we used to open our transportation network's shapefiles and save them as .osm

When we save the line feature class that represents the street network in .osm format, the result is a single file that contains a set of nodes that exist at each intersection and a set of links that represents the streets themselves. The information in the new .osm file is very similar to the information that we would get from converting the street shape file into a topologically defined ESRI coverage file. Figure 3.9 shows the XML format of the street feature class of the study area after saving it as an open street file format.



```
    <node id='-285049' lat='24.73159198758' lon='46.77176336529' />
    <node id='-285050' lat='24.73163968672' lon='46.77174426459' />
    <node id='-285051' lat='24.73167588713' lon='46.77174046495' />
    <node id='-285054' lat='24.73155758581' lon='46.77147346613' />
    <node id='-285055' lat='24.73155758581' lon='46.77153066571' />
    <node id='-285056' lat='24.73154808717' lon='46.77158026602' />
    <node id='-285057' lat='24.73152708711' lon='46.77162596597' />
    <node id='-285058' lat='24.73148708706' lon='46.7716679661' />
    <node id='-285059' lat='24.73072219118' lon='46.77224016875' />
    <node id='-285060' lat='24.72972729638' lon='46.77298117504' />
    <node id='-285061' lat='24.72954919644' lon='46.77311377468' />
    <node id='-285062' lat='24.72949019731' lon='46.77315777581' />
    <node id='-285063' lat='24.72921129686' lon='46.77336607588' />
    <node id='-285064' lat='24.7289095986' lon='46.77359137764' />
    <node id='-285065' lat='24.72866040006' lon='46.77377747975' />
    <node id='-285066' lat='24.72824420191' lon='46.77408718108' />
    <node id='-285067' lat='24.72776010405' lon='46.77444888211' />
    <node id='-285068' lat='24.72736080596' lon='46.77474618539' />
    <node id='-285069' lat='24.7266996091' lon='46.77523858669' />
    <node id='-285070' lat='24.72646070959' lon='46.77541848977' />
    <node id='-285071' lat='24.72621331239' lon='46.77560478973' />
    <node id='-285072' lat='24.725705914' lon='46.77598239167' />
    <node id='-285073' lat='24.72531721532' lon='46.7762725948' />
    <node id='-285074' lat='24.72483231796' lon='46.77663469693' />
    <node id='-285075' lat='24.72480751825' lon='46.77664619656' />
    <node id='-285076' lat='24.72475791794' lon='46.77665759727' />
    <node id='-285077' lat='24.72472741924' lon='46.77665759727' />
    <way id='-200325'>
      <nd ref='-200323' />
      <nd ref='-200324' />
    </way>
    <way id='-200330'>
      <nd ref='-200327' />
      <nd ref='-200328' />
      <nd ref='-200329' />
      <tag k='DISP_CLASS' v='4' />
      <tag k='Mode' v='car' />
      <tag k='OBJECTID' v='3' />
      <tag k='OFF_SPEED' v='20' />
      <tag k='ONEWAY' v='N' />
      <tag k='Permlanes' v='2' />
      <tag k='Shape_Leng' v='175.929326647' />
    </way>
```

*C:\Users\anthe\Desktop\streetsAOI.osm - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
streetsAOI.osm
eXtensible Markup Language file

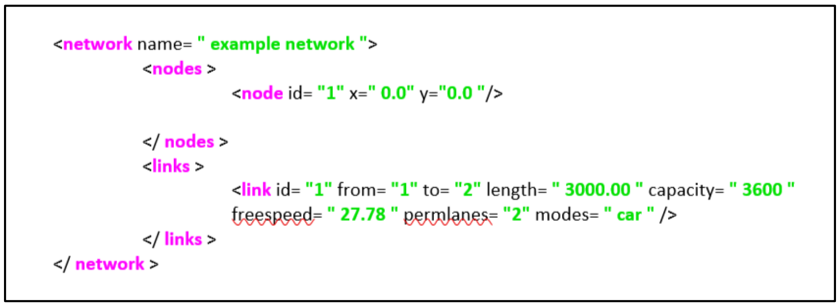
Figure 3.9: XML structure of the streetsAOI.osm file

3.4 MATSim Requirements for simulating and testing the multimodal time-dependent shortest-path algorithm

Although the MATSim XML network data file represents the cornerstone of transportation simulation in MATSim, other data inputs are still needed. Before listing the data inputs that are needed for MATSim to run the simulation, it is important to note that data inputs may differ depending on the simulation scenario. MATSim can use as many as three data inputs: a configuration file (called a config file), a network file, and a population file. In the next subsections, we discuss how we built these files as well as other input data that are necessary for our study.

3.4.1 MATSim Network

The MATSim network XML file has structure shown below in figure 3.10:

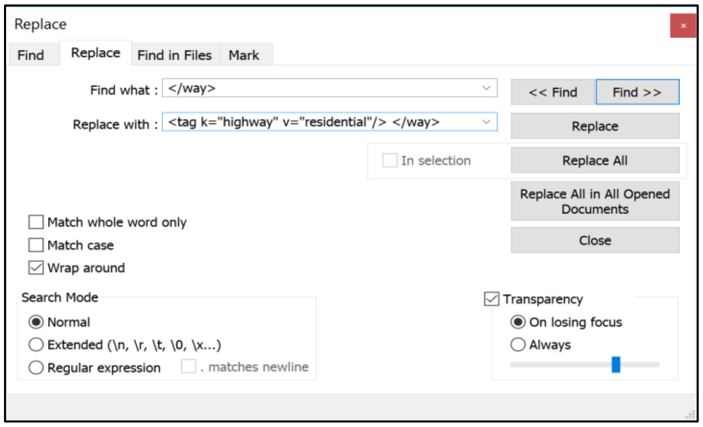


```
<network name= " example network ">
        <nodes >
                <node id= "1" x=" 0.0" y="0.0 "/>

        </ nodes >
        <links >
                <link id= "1" from= "1" to= "2" length= " 3000.00 " capacity= " 3600 "
                freespeed= " 27.78 " permlanes= "2" modes= " car " />
        </ links >
</ network >
```

Figure 3.10: An example for MATSim network XML's data structure

This format is slightly different from the open street map data structure shown in Figure 3.8. In order for us to transform our multimodal transportation network into a format that MATSim can use to simulate the scenarios we have discussed earlier we needed to do the following:

1- We needed to edit the previously created open street map files that represent our multimodal network (streets with origins and destinations locations, walkways, stations, and train lines) by adding a tag to the label <way> of type[11] <highway> and give it a value of “residential” for all links in every file. This was done by using the find-and-replace editing tool in Notepad++[12]. Figure 3.11 shows how this step is done in Notepad++.



Replace

Find what : </way>

Replace with : <tag k="highway" v="residential"/> </way>

In selection

Match whole word only

Match case

[✓ Wrap around]

Search Mode: [✓ Normal] Extended (\n, \r, \t, \0, \x...) Regular expression . matches newline

[✓ Transparency] [✓ On losing focus] Always

Figure 3.11: Editing the street links of the open street map file.

[11] In Open Street Map, it is called attribute pairs key tag, and the letter *k* is used to define the name of the attribute.
[12] Find: <way> and replace with: <tag k="highway" v="residential"/> </way>. This is to define a one-way link. MATSim handles only one-way links (Axhausen, Horni and Nagel 2016). In the case of two-ways link, MATSim internally creates duplicate lines and assigns a direction to each one

2- After making this change, we used a script to make a binary format change for each .osm file into MATSim XML format.

3- For the single-mode scenario, we only needed to convert the streetsAOI.osm file.

4- We repeated this process (1 and 2) with all files for multimode network datasets.

5- Then we opened all the XML files and edited all the numbers that represent node IDs and link IDs in a way that made these IDs unique. Then, we copied all nodes under the label <node> and all the ways under the label <way> in one XML file. Then we saved it in the same file folder that contains the other MATSim input files.

6- To confirm the network was created correctly, from the command prompt, we ran the following command:

“java -Xmx512m -cp “FolderPath”\matsim-0.8.1\networkEditor-0.8.1\networkEditor-0.8.1.jar; “FolderPath”\matsim-0.8.1\matsim-0.8.1.jar org.matsim.contrib.networkEditor.run.RunNetworkEditor”

The above command will result in opening the MATSim network editor window, from which we could open the newly created MATSim XML network and verify its structure. Figure 3.12 shows a screenshot of MATSim’s Network Editor.

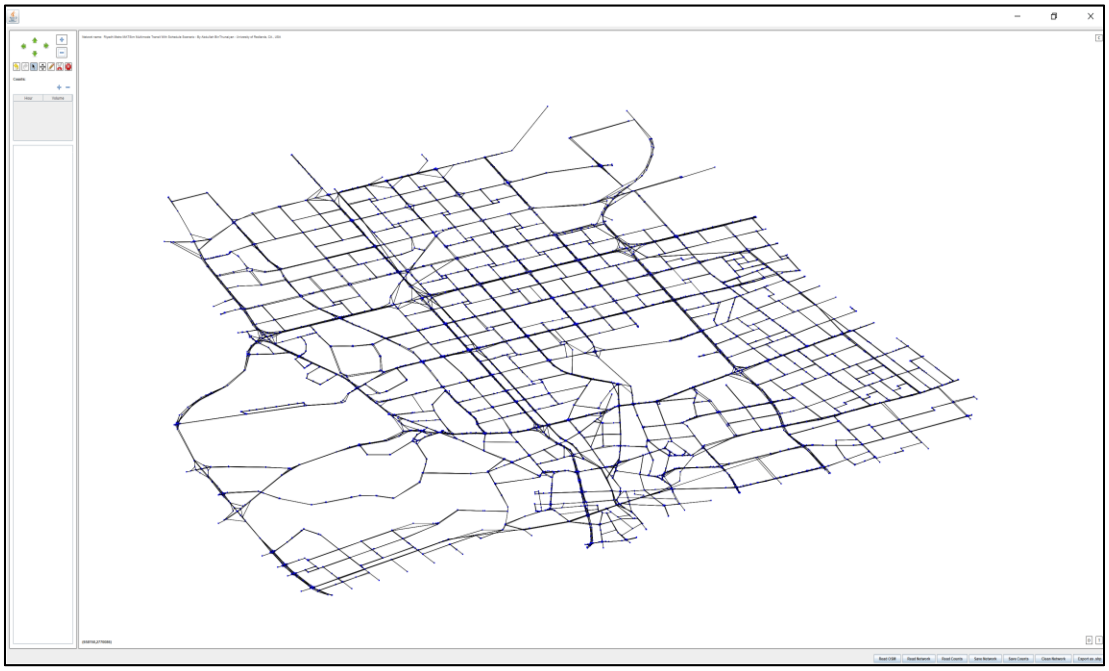


Figure 3.12: Screenshot of MATSim’s Network Editor that we used to open the XML network data file.

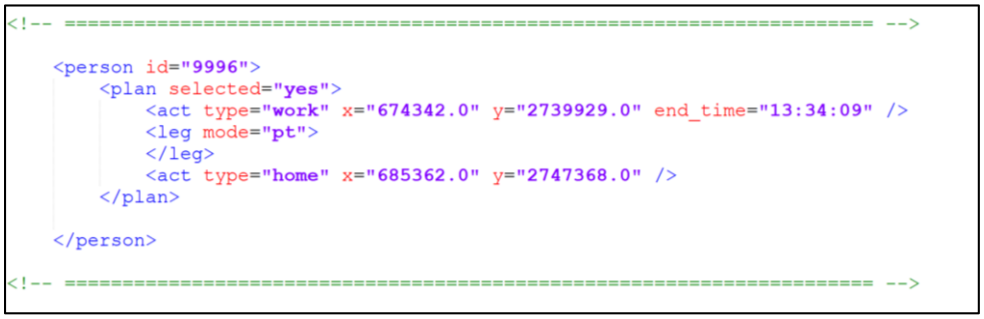
An excerpt of the MATSim XML network data file that we have just build is shown in Appendix B. An important assumption that we made about how we answer our research questions is that agents could move freely between different modes of travel as long as that this achieves a fast arrival to their destinations. With this assumption, agents can leave their own cars at a train station and get on a train vehicle, and then get off the train to walk towards the street and get an immediate “car” ride towards the destination. Agents in MATSim’s terminology are either represented by cars moving in the streets or pedestrians walking on links that allow walking mode or passengers traveling on a train vehicle. Although this assumption rarely exists in reality, it helps in focusing our study to answer the research questions.

After preparing the MATSim network XML, the next section presents how we created the population file that defines how agents move from the origin locations to destination locations.

3.4.2 Population File

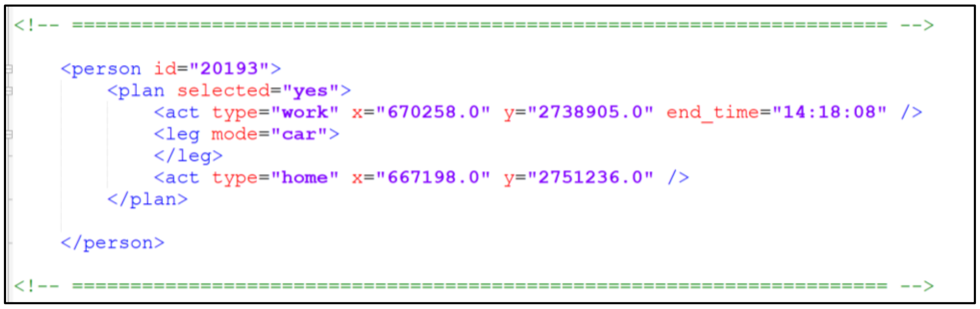
The population file contains agents whose trips will be simulated on the network that we have discussed in the previous section. Each agent in the population file is labeled as <person>. For every person, there are several attributes[13] that describe the travel plan of that person, like the location of origin, type of origin facility, and allowable modes of travel. Figures 3.11 and 3.12 show excerpts of our population file. Two versions of the population file were created: one in which the mode was set to “pt” to be used to test the proposed model, and the other in which mode was set to “car” to test the base-car-only model. The transportation plan for every user or agent, defined by the population file, may include whole daily activities that may add up to as much as 24 hours in duration and consist of two or more stops. In both of our scenarios, all agents’ plans contained only two stops or act types: work and home. As shown in Figures 3.13 and 3.14, we specified different end times for work. In all scenarios, all agents were assigned a “work end time” of between 13:30 to 14:30. However, agents may select different departure times as they learn about the network conditions[14].

[13] In some literature, these attributes are referred to as nested tags.
[14] In terms of its capacities and traffic loads at their times of departure.



```
<!-- ====================================================================== -->

    <person id="9996">
        <plan selected="yes">
            <act type="work" x="674342.0" y="2739929.0" end_time="13:34:09" />
            <leg mode="pt">
            </leg>
            <act type="home" x="685362.0" y="2747368.0" />
        </plan>

    </person>

<!-- ====================================================================== -->
```

Figure 3.13: One person's plan in population file that is used for multimode scenarios



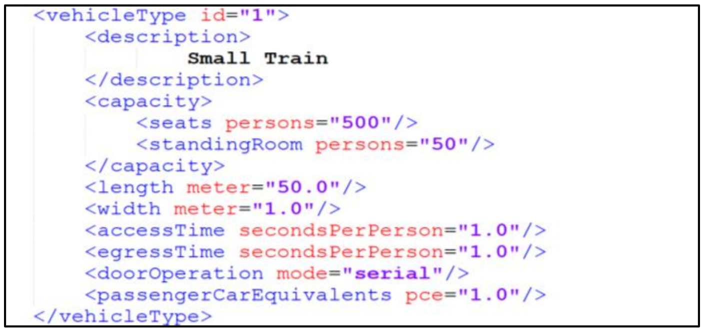
```
<!-- ====================================================================== -->

    <person id="20193">
        <plan selected="yes">
            <act type="work" x="670258.0" y="2738905.0" end_time="14:18:08" />
            <leg mode="car">
            </leg>
            <act type="home" x="667198.0" y="2751236.0" />
        </plan>

    </person>

<!-- ====================================================================== -->
```

Figure 3.14: One person's plan in population file that is used for car-mode scenarios

To build this population XML file, we used a script that contains 30 loops so that we generate agents for each pair of origin-destination location.[15]

3.4.3 Transit Vehicles

This input data file is important in the case of public transportation traffic simulations. In this file, we defined the type of vehicles used in the study area's rail system, their capacities, and the time needed to load or unload travelers. All train vehicles in our transit system are assumed to be of the same type and specifications. Figure 3.15 shows the structure of this input data file.



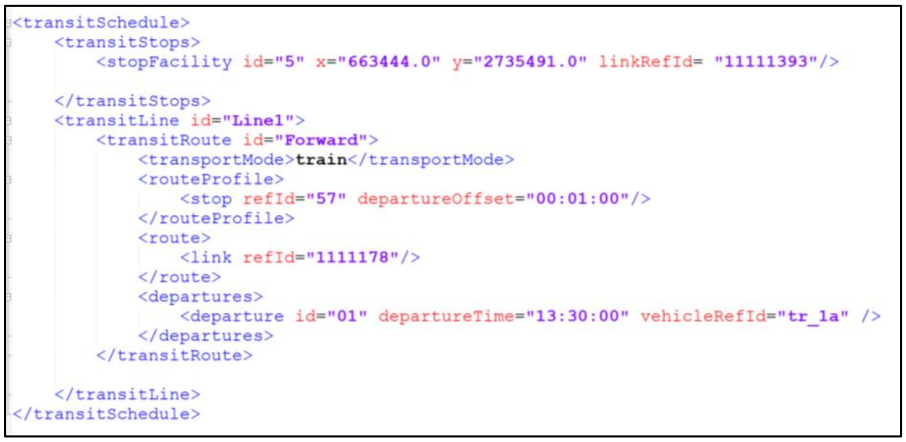
```
<vehicleType id="1">
    <description>
            Small Train
    </description>
    <capacity>
        <seats persons="500"/>
        <standingRoom persons="50"/>
    </capacity>
    <length meter="50.0"/>
    <width meter="1.0"/>
    <accessTime secondsPerPerson="1.0"/>
    <egressTime secondsPerPerson="1.0"/>
    <doorOperation mode="serial"/>
    <passengerCarEquivalents pce="1.0"/>
</vehicleType>
```

Figure 3.15: Transit_Vehicles Input Data File

[15] Other more methods for generating population include generating population from census data. (Andreas, Axhausen and Nagel 2016)

3.4.4 Transit Schedule

This input file is very important to simulate public transport. It is used to provide information about four main attributes: transit line, transit stops, transit route, and departure times. For example, in the defining a route, we provide information about the sequence of stops and the links followed, arrival and departure times, as well as time offsets that are expected to delay those arrival and departure times. Figure 3.16 shows a typical data structure for a transit schedule XML file.



```xml
<transitSchedule>
    <transitStops>
        <stopFacility id="5" x="663444.0" y="2735491.0" linkRefId= "11111393"/>

    </transitStops>
    <transitLine id="Line1">
        <transitRoute id="Forward">
            <transportMode>train</transportMode>
            <routeProfile>
                <stop refId="57" departureOffset="00:01:00"/>
            </routeProfile>
            <route>
                <link refId="1111178"/>
            </route>
            <departures>
                <departure id="01" departureTime="13:30:00" vehicleRefId="tr_1a" />
            </departures>
        </transitRoute>

    </transitLine>
</transitSchedule>
```

Figure 3.16: Typical data structure for a transit schedule XML file.

The next section discusses a necessary input data file for any MATSim simulation, the configuration file.

3.4.5 Configuration File

The configuration file is used to define or establish a connection between the other input data files, also referred to as simulation modules, and MATSim. It is also used to allow the user to define simulation parameters that define and govern the behavior of the agents. In this file, these modules that govern the behavior are specified with two parameters: the first is the name of the module,[16] and the second is a value for the parameter or a path link to the files that contain data that represents the values of that parameter. There are many types of data inputs or modules that could be used for simulating traffic in MATSim. In this study, two **config.xml** files were created, one with a defined module for transit mode and another where the transit mode was disabled.

[16] Some modules are built in within MATSim, like the **controller**, and the user is only required to set the value of its parameters (like the number of iterations). Some modules are built by the user and fed to MATSim by setting their values as a path link that points to their location like the **network** module.

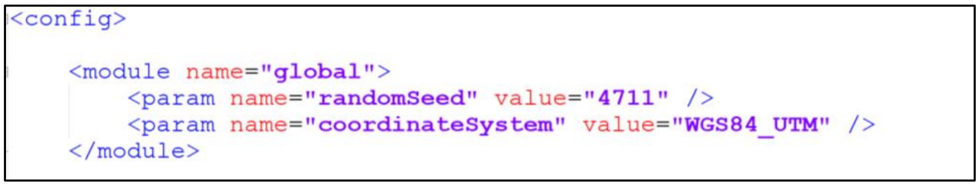
The following subsections provide a brief description of the **config.xml** modules[17] used in this study and the way they are specified for the traffic simulation scenarios described earlier in section 7 (experimental design).

1- Global Module:

This module is important because it tells MATSim where the network data is located on the globe. MATSim uses the Pythagoras theorem for calculating distances between any two points. Therefore, a Cartesian coordinate system is needed (Axhausen, Horni and Nagel 2016). Since the study area data was converted from its format of open street map, which is in the WGS-84 coordinate system, we created a script that performs coordinate system conversion from WGS-84 to UTM-Zone 38.

This script is based on Dr. Steven Dutch's "Convert Geographical Units" webpage (Dutch 2017).

Appendix C shows script for this module that we recommend using with MATSim for simulating traffic in Saudi Arabia. We used the file location of that script as a value for the parameter that defines the coordinate system. Figure 3.17 shows the global module settings of both scenarios analyzed in this study.



```
<config>

    <module name="global">
        <param name="randomSeed" value="4711" />
        <param name="coordinateSystem" value="WGS84_UTM" />
    </module>
```

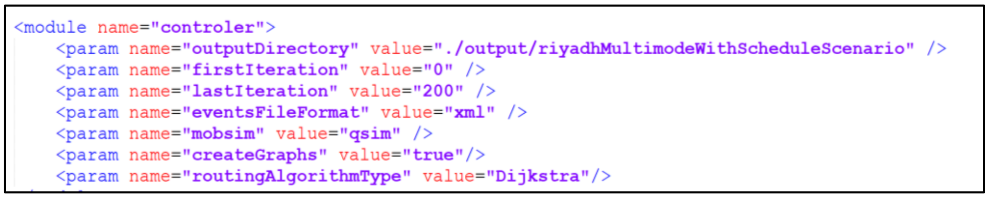
Figure 3.17: Settings of the Global module

2- The Controller Module

This module is used to set the output directory, which MATSim then uses to present the simulation's results. It is also used to specify the base shortest path algorithm used for routing agents in their trips. Moreover, it is used to specify the

[17] The results of the simulations are largely influenced by the modules we describe here. Thus, reading this section carefully (the config.xml input data file) helps improve understanding of the results of the simulations.

number of simulations' iterations and the mobility engine (mobsim)[18] that is used for simulating traffic flows. Figure 3.18 shows the global module settings of both scenarios analyzed in this study.



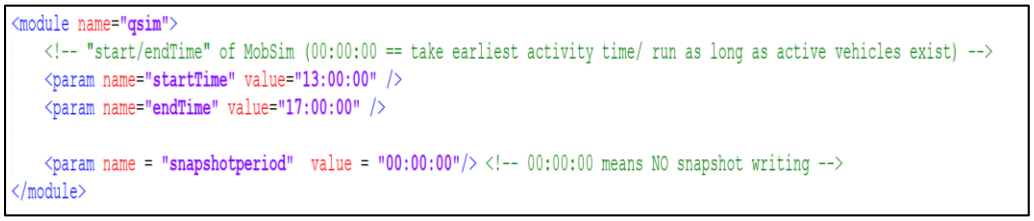
```
<module name="controler">
    <param name="outputDirectory" value="./output/riyadhMultimodeWithScheduleScenario" />
    <param name="firstIteration" value="0" />
    <param name="lastIteration" value="200" />
    <param name="eventsFileFormat" value="xml" />
    <param name="mobsim" value="qsim" />
    <param name="createGraphs" value="true"/>
    <param name="routingAlgorithmType" value="Dijkstra"/>
```

Figure 3.18: Settings of the Controller module

3- QSIM Module

This module is the default mobility simulation engine built-in MATSim. In this module, the user specifies the times of the agents' daily plan that is simulated. Figure 3.18 shows this module's settings in both of the scenarios implemented in this study. Although agents in the population file start departing from the origin towards destination locations between time 13:00 and 15:30, the simulation end time is set to be 17:00 to give the last departing agent enough time to arrive at its destination. We decided to set the simulation to end at 17:00 because it is unrealistic for users in our study area to spend over one and half hour from an origin to a destination location. Even if the network is overloaded to cause such situation, our research focuses on different aspects that can be achieved more efficiently with reasonably enough simulation duration time.

Figure 3.19 shows the QSIM module settings of both scenarios analyzed in this study.



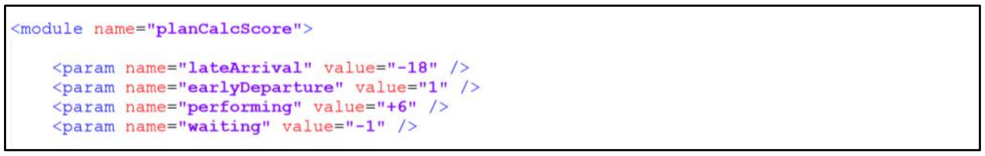
```
<module name="qsim">
    <!-- "start/endTime" of MobSim (00:00:00 == take earliest activity time/ run as long as active vehicles exist) -->
    <param name="startTime" value="13:00:00" />
    <param name="endTime" value="17:00:00" />

    <param name = "snapshotperiod"  value = "00:00:00"/> <!-- 00:00:00 means NO snapshot writing -->
</module>
```

Figure 3.19: Settings of the QSIM module

[18] Other simulation engines that are not built into MATSim can be used as well.

4- Plan's Score Calculator Module

This module specifies the advantages the agents should achieve, like time or money savings, and select their routes based on these objectives. Within MATSim configuration settings, this module is named "PlanCalcScore". In this module, we specified attributes' scores in a way that encourages agents to select fastest-time routes. Figure 3.20 shows an excerpt of this module's specifications.



```
<module name="planCalcScore">

    <param name="lateArrival" value="-18" />
    <param name="earlyDeparture" value="1" />
    <param name="performing" value="+6" />
    <param name="waiting" value="-1" />
```

Figure 3.20: Settings of the PlanCalcScore module

The parameters specified in this module are used to calculate the marginal utility of going from work to home, or an activity i, using the shortest path models that are implemented in this study is calculated using the following function:

$$U_i = U_{lat,i} + U_{ear,i} + U_{per,i} + U_{wait,i} \qquad (3.1)$$

Every agent tries to maximize its marginal utility at every iteration of the simulation. In brief, the parameters that we have specified above are:

- Late Arrival: arriving late to a destination.
- Early Departure: departing early to a destination.
- Performing: making the activity, or the trip[19].
- Waiting: time spent before an activity or a public transport becomes available.

This module is important to the learning process that agents undergo throughout the simulation's iterations. In every iteration, MATSim produces up to 6 different routes and scores their utility[20]. With these plans being produced, the agents then perform an iterative learning process that technically continues as many as the

[19] Recommended to be set at +6 in the case of not defining a mode choice logit model (Axhausen, Horni and Nagel 2016). We did not specify such model and the mode selection in all of the multimodal scenarios in this study are strictly based on the fastest time to destination.

[20] The plan with worst score is deleted from the agent's memory for the next iteration. The reader is recommended to read (Nagel, Axhausen, et al. 2016) for more discussion on this plan generation and removal approach of MATSim.

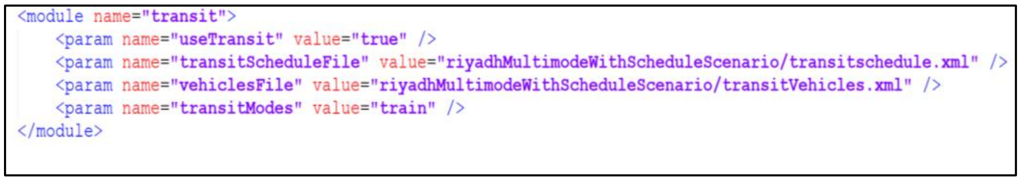
specified number of iterations. This process was described by Flotterod (2016) in the following steps:

- Iterate:
  1- Based on the plan's score and the previously observed network conditions, Agents select a travel plan.
  2- Agents execute this plan.
  3- Agents observe the resulting network conditions, and learn for the next iteration

This iterative learning process is important in understanding how agents select their routes. As we will see in the results section, executed plans are not necessarily the ones with the highest utility score. But in a well-designed simulation, the average utility score of the executed plans should be very close to the one of the best plan after few iterations. (Flotterod 2016) described this notion of learning through iterations as "moving MATSim closer to its solution point." Our study benefited from such setting in the sense that we are testing the computational complexities in the presence of a functionality of rerouting users when network conditions change.

5- Transit Module

This module is specific to the config file that is used when public transit routing is an option.[21] While the transit mode value can be set as one or more modes directly within this module, the vehicle specifications and transit schedule are usually assigned a value of a path to the folder that contains their XML file names. Figure 3.21 shows an excerpt of this module description.



```
<module name="transit">
    <param name="useTransit" value="true" />
    <param name="transitScheduleFile" value="riyadhMultimodeWithScheduleScenario/transitschedule.xml" />
    <param name="vehiclesFile" value="riyadhMultimodeWithScheduleScenario/transitVehicles.xml" />
    <param name="transitModes" value="train" />
</module>
```

Figure 1.21: Settings of the PlanCalcScore module

[21] This module is disabled in the case of simulating the single mode.

In addition to these configuration modules, other modules that establish a connection between MATSim and the other input files. Appendix D contains the full settings of the configuration files that are used to test both the time-dependent, capacity-aware, car-mode shortest path and the proposed capacity-, mode-, schedule-, and time-dependent shortest-path models described earlier. So far, we have presented how to prepare ArcGIS-based transportation network data to perform a multiagent traffic simulation. Moreover, we described how to build and configure the necessary data input files that are required by the simulation software used in this study.[22]

[22] Studies that have different objectives will most likely require different settings.

CHAPTER 4

RESULTS AND DISCUSSIONS

To answer the research questions, we decided to test the model utilizing three different loads on the network, or in another word, three different numbers of agents. For each different number of agents' scenario, we performed two simulations. The first simulation tested the base time-dependent, capacity-aware, car-mode, shortest-path model. The second simulation tested the proposed capacity-, mode-, schedule-, and time-dependent shortest-path model. For each of these tests, we created a population input data file to describe the trip preferences for each agent.

The numbers of population (agents) we tested in the three scenarios are 15,000 agents in the first scenario, 45,000 agents in the second scenario, and 75,000 agents in the third scenario. Other than the number of agents and their configuration setting which specifies whether or not they could use mixed modes of travel. All input data and software configurations were kept the same between scenarios. We ran 200 iterations for each simulation. All simulations were completed on a computer running Windows 7 with 16 GB of random access memory. The following subsections summarize the results for each case.

To answer our research question, we decided to simulate traffic during the rush hour when schools' closing times overlap with break times for major businesses and the closing time for government offices. In Riyadh, these activities occur from 1:00 PM to 3:30 PM. During this period, we tested both discussed models three times. In each scenario, we initiated 500, 1500, and 2500 trips from each origin point to its paired destination point. With these different loads of traffic, we were able to establish more informed answers to the research questions. The next section provides a brief overview of the simulation software that we used in this study and how the simulations of our scenarios were implemented.

4.1 Results of Scenario 1: 15,000 Agents

In this scenario, we ran the simulation using 15,000 agents. we ran two versions of this scenario. In the first, agents are allowed to only use the car mode of travel. In the second, we the simulation was configured to allow agents to use mixed-modes of travel. The next subsections present the results of both simulations of this scenario.

4.1.1 Scenario 1-A: Time-Dependent, Capacity-Aware, Car-Mode, Shortest-Path Model

In this simulation, all agents can use only the car mode. In addition, the utility function that each agent sought to maximize stresses the rapid arrival to a destination and discourages waiting in queues of traffic. For every simulation's iteration, some agents might encounter delays in some links or nodes that are on their path from origin to destination points. The strategy defined by the utility function of the shortest path models used in this study encourages agents to try to avoid such links and nodes in the next simulation's iteration.

The utility score is discussed here to show that routes are selected based on the preferences defined in our models. Then we show other metrics, namely run time, average trip duration, average trip distance, and traffic assignment and mode usage.

4.1.1.1 Average Utility Score[23]

The average utility score for all executed shortest paths in this scenario converged at around 43.7. In each iteration, MATSim produces an output plan file. In this file, several alternative routes are provided and prioritized in terms of their appropriateness for each agent. Whether the route is selected or not depends largely on the utility score.[24] The score presented here is of an absolute unit and it is calculated based on the function defined in the configuration file discussed earlier. Figure 4.1 shows the average utility score per iteration for this scenario.

[23] Also referred to as plan score. It has no unit but the larger the score the better it is.

[24] The higher the score, the better the route. Usually the highest score is referred to as the "best" score. After agents execute a plan with the "best" score and observe the network conditions, they try a different plan in the next iteration in a try to improve the trip duration.

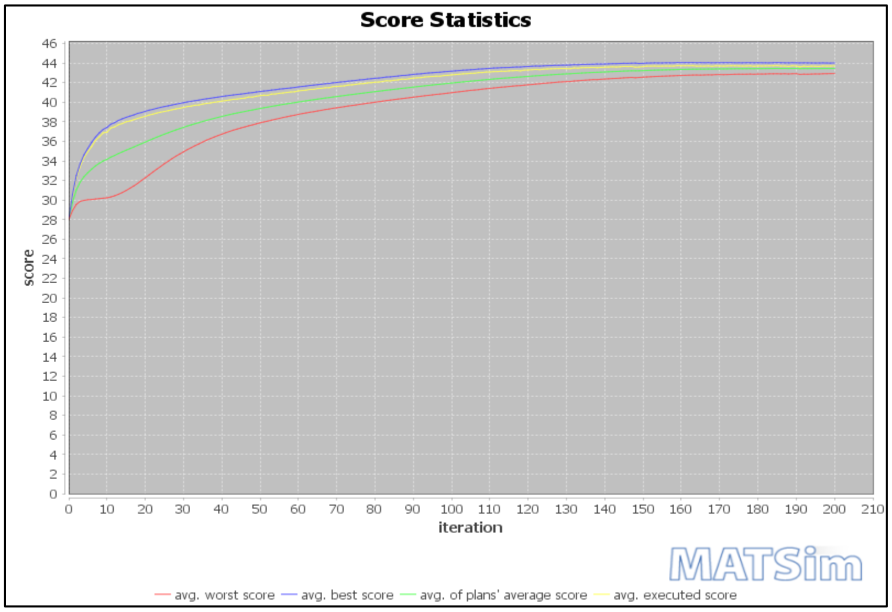


Figure 4.1: Average utility score per iteration for scenario 1-A.

We notice that the score improves as more iterations of the simulation are executed. This is due to the learning process during which each agent uses the effect on links' free-flowing speeds introduced by other agents in the previous iteration to adjust the route, mode, or departure time in the current iteration. In the end, the process reached a user-equilibrium state by which users seem unable to further improve their utility score beyond 43.72 by making allowed changes to their trips.[25] The convergence in this scenario shows that user equilibrium can be searched for if we have 15,000 users of agents within the computational power detailed above and run time, which will be discussed later. In this scenario, convergence started approximately after the 120th iteration, for which the score is 43.3.

4.1.1.2 Run time of the simulation

The simulation was executed for 200 iterations. The total run time for all iterations in this simulation is 41 minutes and 26.031 seconds. The average run time for all iterations was found to be 11 seconds. The summation of the average run time of all iterations is smaller than the overall run time, because the simulation as a whole performs other tasks that include writing aggregated results of all iterations to the output folder.

[25] As explained in Section 4.4.5, all agents have been assigned the same utility function specification. They only differ in their origin-destination trip specifications, their departure times, or both.

4.1.1.3 Average Trip Duration

The metric of average trip duration is very important to our study. If we can conclude that the average trip duration in the case of capacity-, mode-, schedule-, and time-dependent shortest-path model is similar to or less than the average trip duration in the case of time-dependent, capacity-aware, car-mode, shortest-path model, then we can present a strong case to promote the usage of the earlier model for navigational applications.

In this scenario of the time-dependent, capacity-aware, car-mode, shortest-path model for 15,000 agents, the average trip duration after the 200th iteration is found to be 23 minutes and 59 seconds. We only reported this metric for iteration 200 because, unlike the utility score and distances, this metric does not get reported for all iterations in one graph or text file by MATSim, a limitation of MATSim that will be discussed later.

4.1.1.4 Average Trip Distances

Although we simulated traffic over a diverse range of distances, it is still important to report this metric and compare it across different models at different traffic loads to see if any insight could be gained about these models across different scenarios. For this specific scenario, the average distance for the 15,000 trips after the 200th iteration is 17549.4 meters. Figure 4.2 shows the average distances for all 200 iterations.

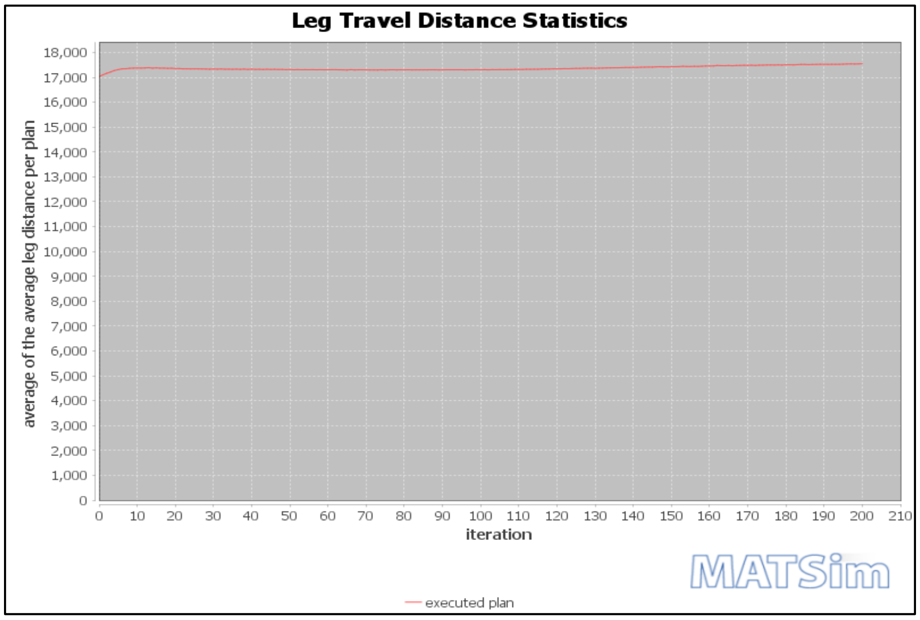


Figure 4.2: Average trips' distances for all the 200 iterations for all 15,000 trips

4.1.1.5 Traffic Assignment and Mode Usage

MATSim produces very useful information:

i- Number of users who started their trips for a prespecified time-slots (default is every 5-minutes).

ii- The mode of travel those users are employing.

iii- Number of users who arrived at their destinations.

iv- Number of users who are stuck, and the mode they are using.

The importance of these information in our study is to gain a deeper insight on whether the use of mixed-mode route-guidance could improve travel experienced by its users compared to if the users are guided by only a car-mode route guidance model.

Table 4.1 shows the numbers of agents who have finished departure, arrival, or are still in route during each time interval simulated. We notice that all routes are free-flowing[26] for agents at their start time of travel or during their trips. In addition, figure 4.3 shows a histogram for trip departures and arrival as en-route trips for this scenario. We noticed that at time 14:45 en-route trips line coincides with arrival trips which indicates that simulation has ended because all agents have arrived successfully to their destinations.

[26] Because the number of “stuck car” is 0.

Table 4.1:Numbers of agents who have finished departure, arrival, or are still in route for scenario 1-A

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car
13:00:00	9557	0	0	9557	9557	0	0	9557
13:05:00	889	138	0	10308	889	138	0	10308
13:10:00	1059	719	0	10648	1059	719	0	10648
13:15:00	1054	1654	0	10048	1054	1654	0	10048
13:20:00	833	2181	0	8700	833	2181	0	8700
13:25:00	541	2463	0	6778	541	2463	0	6778
13:30:00	424	2443	0	4759	424	2443	0	4759
13:35:00	281	2036	0	3004	281	2036	0	3004
13:40:00	161	1448	0	1717	161	1448	0	1717
13:45:00	80	841	0	956	80	841	0	956
13:50:00	46	521	0	481	46	521	0	481
13:55:00	36	290	0	227	36	290	0	227
14:00:00	17	132	0	112	17	132	0	112
14:05:00	6	52	0	66	6	52	0	66
14:10:00	7	31	0	42	7	31	0	42
14:15:00	2	28	0	16	2	28	0	16
14:20:00	3	8	0	11	3	8	0	11
14:25:00	3	4	0	10	3	4	0	10
14:30:00	1	5	0	6	1	5	0	6
14:35:00	0	1	0	5	0	1	0	5
14:40:00	0	4	0	1	0	4	0	1
14:45:00	0	1	0	0	0	1	0	0
Total	15000	15000	0		15000	15000	0	

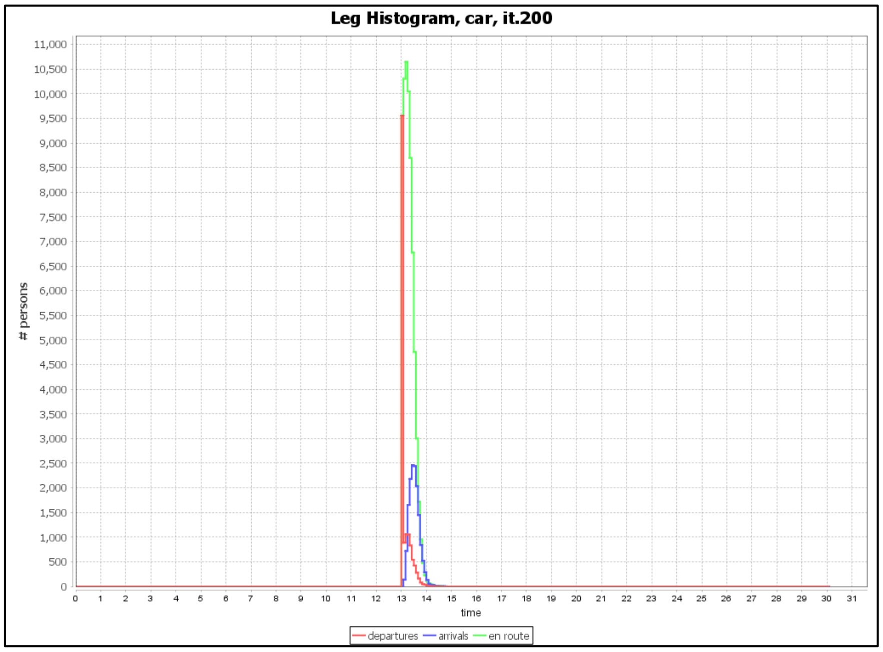


Figure 4.3: Histogram for trip departures and arrival as well as en-route trips for scenario 1-A

4.1.2 Scenario 1-B: Capacity-, Mode-, Schedule-, and Time-Dependent Shortest-Path Model (15,000 Agents)

In this scenario, we used the same number of agents as the previous one (15,000), but agents are allowed to use mixed modes of travel.

4.1.2.1 Run time of the simulation

The overall simulation's run time was found to be 43 minutes and 22.64 seconds. The average run time for all iterations was 12 seconds.

4.1.2.2 Average Utility Score

The average utility score for the executed plan at the 200th iteration is 43.85 compared to 43.72 in scenario 2-A. Figure 4.4 shows the average utility score per iteration for this scenario.

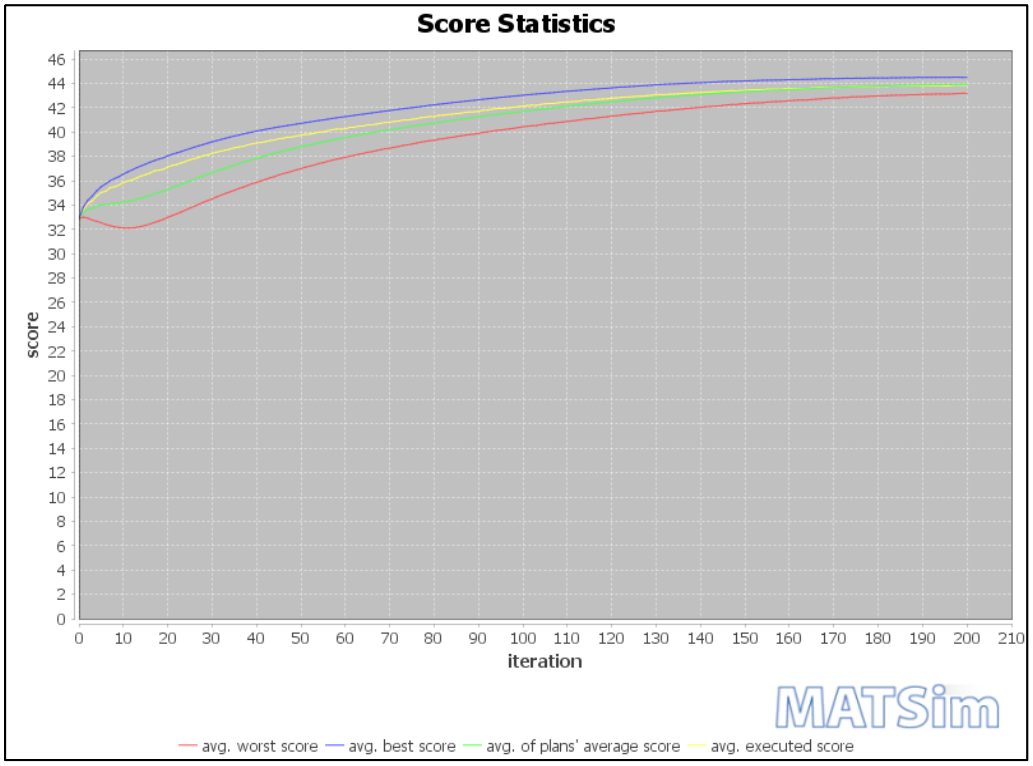


Figure 4.4: Average utility score per iteration (scenario 1-B)

4.1.2.3 Average Trip Duration

In this scenario, the average trip duration in the 200th iteration was found to be 21 minutes and 21 seconds. This is about 1 minute and 40 seconds less than the previous scenario, in which only car mode is allowed.

4.1.2.4 Average Trip Distances

The average trip distances for all 15,000 agents at the 200th iteration is 17516.43 meters, compared to 17549.4 meters in the previous simulation (scenario1-A). Figure 4.5 summarizes the average distances for all iterations of this scenario.

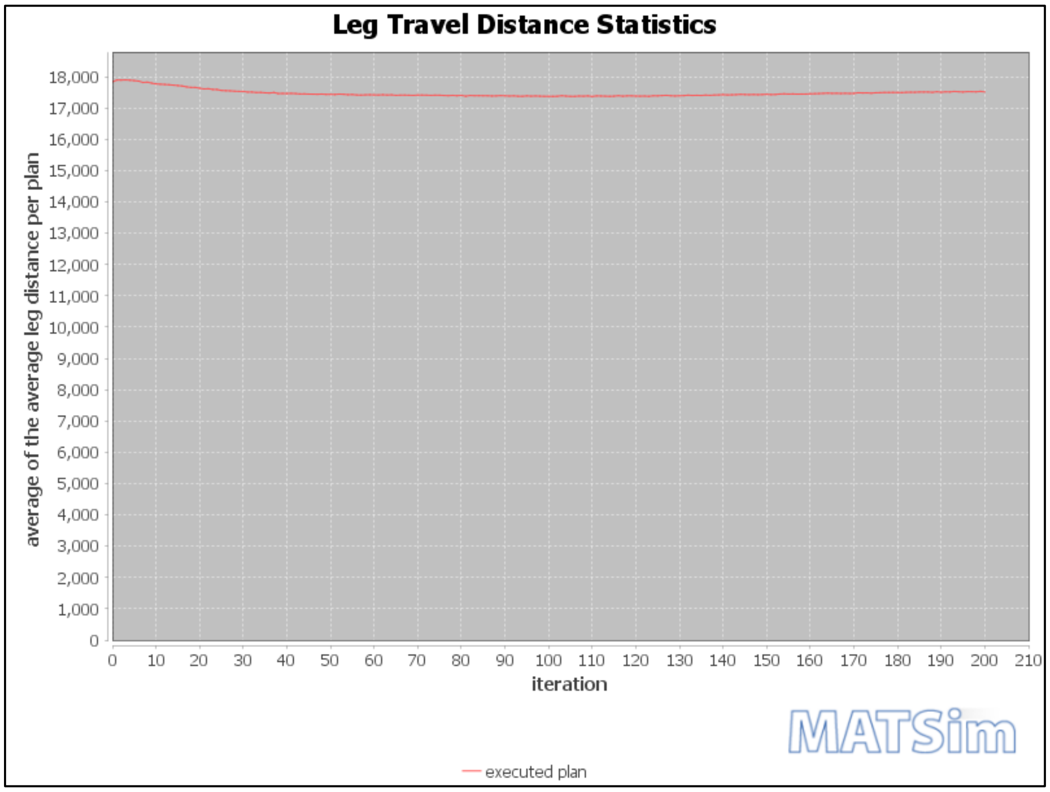


Figure 4.5: Average trips' distances for all 200 iterations for all the 15,000 trips in scenario 1-B

4.1.2.5 Traffic Assignment and Mode Usage

In the 200th iteration of this scenario, 12,307 agents (82.047%) were routed through car mode only, and the remaining agents (2693, 17.953%) were assigned mixed modes of travel toward their destinations. Most agents who selected mixed modes of travel did so at the start of the simulation (2,027 agents). This is because our model discourages waiting in queues. In all discussed scenarios, most traffic queues form at the beginning of the simulation, when all agents try to exit the workplace through a one-way street link, then going either north or east toward their destinations.

Figure 4.6 shows a graph of iteration 200 for the users who were assigned mixed-mode routes referred to in the graph as “pt,” and figure 4.7 shows a graph of the same iteration for users who were routed through car-mode only throughout their trips in this scenario.

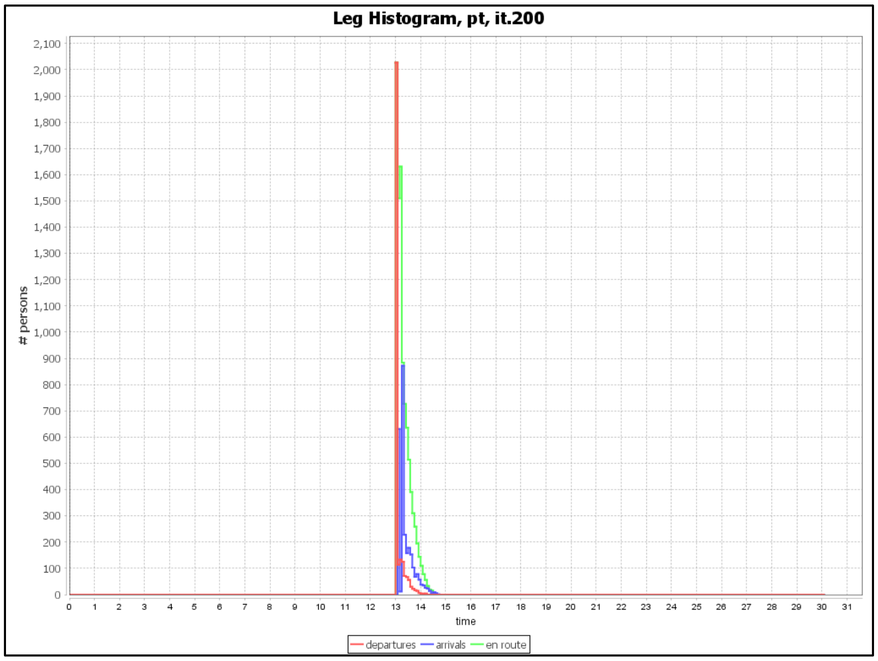


Figure 4.6: Iteration 200 of Scenario 1-B, users who were assigned mixed-mode routes

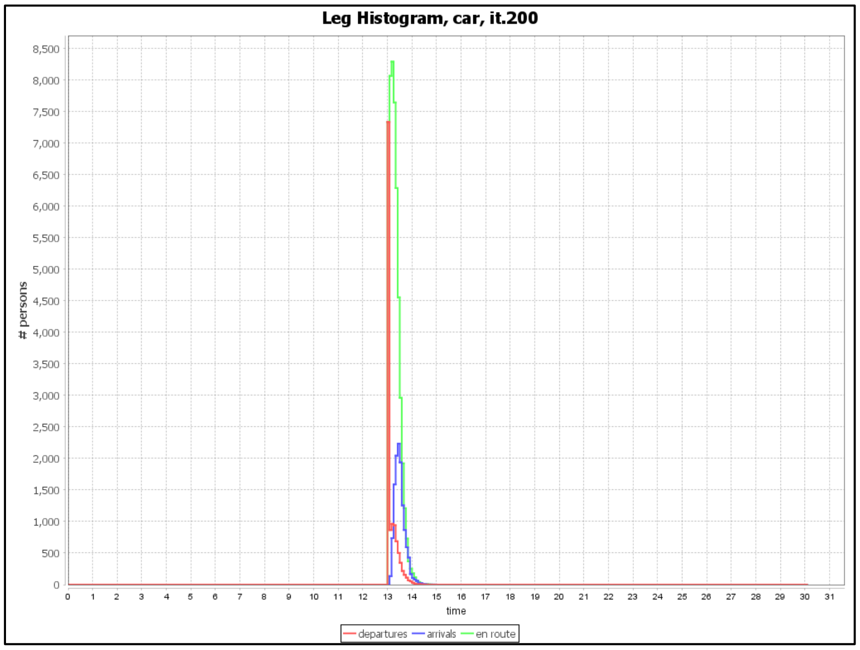


Figure 4.7: Iteration 200 of Scenario 1-B, number of users who were assigned car-mode only during the same run.

Table 4.2 shows the numbers of agents who have finished their departure or arrival or are still en-route as well as their assigned mode of travel (either car only or mixed) for each time interval throughout the simulation. We notice that all routes are free-flowing for agents who were assigned either car-only mode or mixed-mode travel. This indicates that 15,000 agents departing from 30 nearby locations is probably a small number for a rush-hour period. This was shown earlier in section 4.1.1.5, where we saw that all agents are assigned to car-only mode, because they have no other option, yet the number of agents who are stuck in the network was still zero. However, in this scenario, we saw some agents being recommended to use mixed modes of travel at this low level of traffic. This is consistent with the travel duration results where we found that, in this section, the travel duration is less in the case of mixed modes of travel compared to the car mode. Such a result presents a strong case for adopting and using mixed-mode models in navigational applications. .

Table 4.2: Numbers of agents who have finished departure, arrival, or are still in route based on their assigned mode of travel. (scenario 2-B)

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car	departures_pt	arrivals_pt	stuck_pt	en-route_pt
13:00:00	9364	0	0	9364	7337	0	0	7337	2027	0	0	2027
13:05:00	979	765	0	9578	865	134	0	8068	114	631	0	1510
13:10:00	1096	749	0	9925	963	737	0	8294	133	12	0	1631
13:15:00	1067	2461	0	8531	942	1589	0	7647	125	872	0	884
13:20:00	758	2272	0	7017	687	2044	0	6290	71	228	0	727
13:25:00	569	2394	0	5192	502	2236	0	4556	67	158	0	636
13:30:00	402	2117	0	3477	346	1939	0	2963	56	178	0	514
13:35:00	247	1410	0	2314	217	1257	0	1923	30	153	0	391
13:40:00	177	969	0	1522	155	866	0	1212	22	103	0	310
13:45:00	127	660	0	989	110	592	0	730	17	68	0	259
13:50:00	81	507	0	563	67	429	0	368	14	78	0	195
13:55:00	58	224	0	397	52	167	0	253	6	57	0	144
14:00:00	34	142	0	289	30	104	0	179	4	38	0	110
14:05:00	18	114	0	193	15	79	0	115	3	35	0	78
14:10:00	10	85	0	118	6	59	0	62	4	26	0	56
14:15:00	6	50	0	74	6	27	0	41	0	23	0	33
14:20:00	3	37	0	40	3	23	0	21	0	14	0	19
14:25:00	2	16	0	26	2	12	0	11	0	4	0	15
14:30:00	1	15	0	12	1	6	0	6	0	9	0	6
14:35:00	1	7	0	6	1	2	0	5	0	5	0	1
14:40:00	0	3	0	3	0	2	0	3	0	1	0	0
14:45:00	0	2	0	1	0	2	0	1	0	0	0	0
14:50:00	0	0	0	1	0	0	0	1	0	0	0	0
14:55:00	0	1	0	0	0	1	0	0	0	0	0	0
Total	15000	15000	0		12307	12307	0		2693	2693	0	

4.2 Results of Scenario 2: 45,000 Agents

In this scenario, we ran the simulation in a similar fashion as we did in scenario 1 but we used 45,000 agents here instead of 15,000. The results of this scenario (both car mode and mixed-modes) are discussed in the following subsections.

4.2.1 Scenario 2-A: Time-Dependent, Capacity-Aware, Car-Mode, Shortest-Path Model

4.2.1.1 Average Utility Score

The average utility score for all executed shortest paths in this scenario converged to around **36.9**. we notice that this score is lower than the previous scenario (43.72 in scenario 1-A). This indicates that more waiting times or/and later arrivals have been encountered by agents. In this scenario, convergence started approximately after the 70th iteration in which the score is 36.6. Figure 4.8 shows the average utility score per iteration for this scenario.

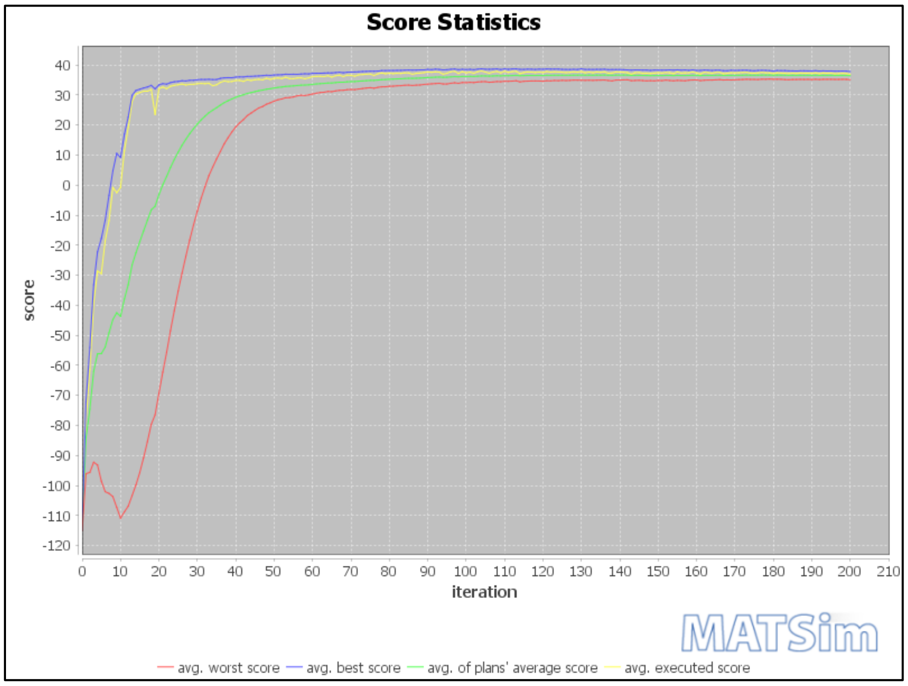


Figure 4.8: Average utility score per iteration, scenario 2-A

We noticed that at the first 20 iterations, average executed and best scores went from -90 to about +30. This is because of the way scoring is defined in the configuration file. Not performing the trip, late arrivals, or waiting in queues can all impact the utility score. When this average score jumps from a small number to a large number (-90 to +30), then there is an indication that some agents are accumulating most of the (dis)utilities that we defined in the utility function[27]. We notice that agents learnt and adapted to the network conditions after the 20th iteration. We did not see such a behavior in the previous scenario (section 4.1) when the traffic load was 15,000 agents. Because the network structure and capacities are the same as in the previous scenario, it is the size of traffic loads in this scenario that caused this phenomenon. We will explore other metrics and then discuss this behavior in the context of network structure and capacities.

4.2.1.2 Run-time of the simulation

The simulation was executed for the whole 200 iterations. The total run-time for this simulation is 127 minutes and 20.55 seconds. The average run-time for all iterations was found to be around 36 seconds.

4.2.1.3 Average Trip Duration

In this scenario, the average trip duration after the 200th iteration has been found to be 45 minutes and 43 seconds. It is larger than the average time in the previous scenario (scenario 1-A).

4.2.1.4 Average Trip Distances

For this specific scenario, the average distance for the 45,000 trips after the 200th iteration is: 17976.4 meters. Figure 4.9 shows the average trips' distances for all the 200 iterations.

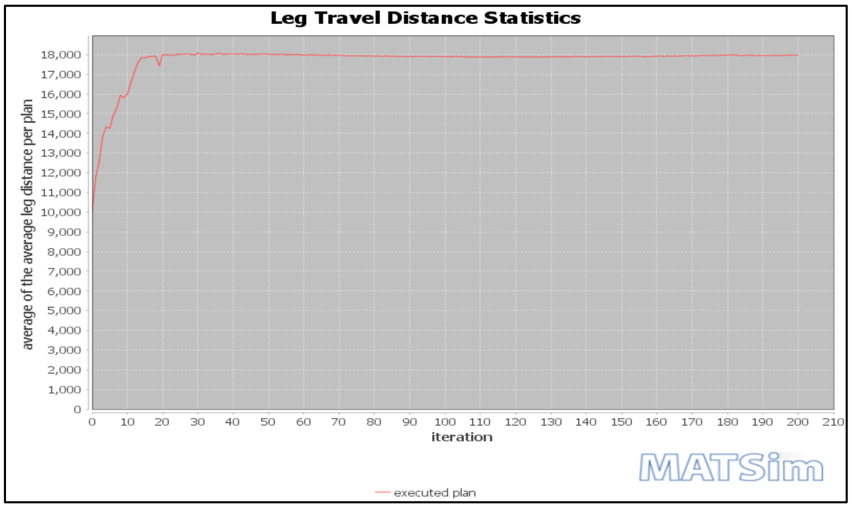


Figure 4.9: Average trips' distances for all the 200 iterations for all the 45,000 trips of scenario 2-A

[27] It also indicates that MATSim's iterative learning process is fast.

We notice in figure that average trip distances did not converge to about 18,000 meters until after the 20th iteration. From this graph, we can identify that the reason is that some agents did not make an arrival to their destination, or in MATSim's terminology did not perform the trip. Figure 4.10 shows a histogram for trip departures and arrival as well as en-route trips for the first iteration of this scenario. As specified in the simulation's configuration file, simulation is scheduled to end at time 17:00, and we see in this graph that the simulation ended while about 12,000 agents (green line) are still en-route.

The reason these agents were not able to make an arrival in a reasonable time is that these agents were trying to execute a travel plan of a high score. But the score in the first iterations does not capture traffic waiting times or late departures that are caused by many vehicles trying to move through the same links. Such un-awareness of traffic conditions caused these agents to get stuck in bad routes until the simulation ended; therefore, the distances of their trips were not counted in calculating the average distance in the first 20 iterations. However, the iterative-learning approach of MATSim causes this problem to disappear and all agents will be assigned better routes as we see later in figure 4.11.

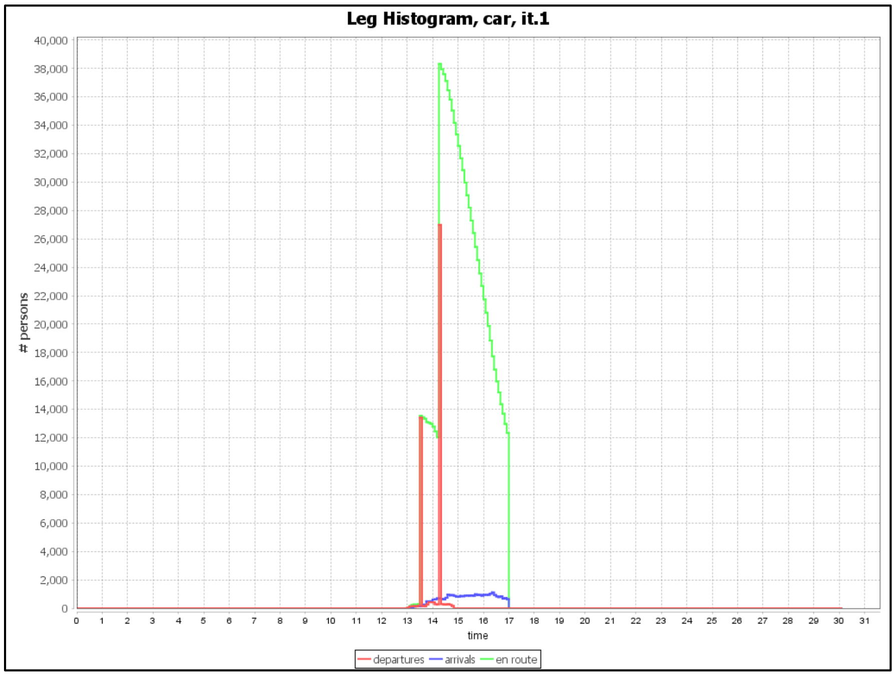


Figure 4.10: Iteration number 1 of scenario 2-A, number of agents who departed or arrived as well as those who are stile n-route.

4.2.1.5 Traffic Assignment and Mode Usage:

Table 4.3 shows the numbers of agents who have finished departure, arrival, or are still in route. We notice that all routes are free-flowing for agents at their start time of travel or during their trips. Although routes are reported here to be free-flowing, the fact that average trip time is larger than when the number of agents is 15,000 indicates that agents are forced to travel at lower speed here. Figure 4.11 shows a histogram for trip departures and arrival as en-route trips for this scenario. As shown in table 4.3 and figure 4.11, simulation ended at time 16:00 (before 17:00 that we've specified in the configuration file as "simulation shut-down time") because all agents have arrived successfully to their destinations.

Table 4.3: Numbers of agents who have finished departure, arrival, or are still in route for scenario 2-A, Iteration 200

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car
13:00:00	24742	0	0	24742	24742	0	0	24742
13:05:00	1395	136	0	26001	1395	136	0	26001
13:10:00	1399	640	0	26760	1399	640	0	26760
13:15:00	1347	1513	0	26594	1347	1513	0	26594
13:20:00	1321	2006	0	25909	1321	2006	0	25909
13:25:00	1311	2181	0	25039	1311	2181	0	25039
13:30:00	1300	2154	0	24185	1300	2154	0	24185
13:35:00	1260	2361	0	23084	1260	2361	0	23084
13:40:00	1234	2319	0	21999	1234	2319	0	21999
13:45:00	1270	2474	0	20795	1270	2474	0	20795
13:50:00	1207	2391	0	19611	1207	2391	0	19611
13:55:00	1045	2479	0	18177	1045	2479	0	18177
14:00:00	966	2498	0	16645	966	2498	0	16645
14:05:00	827	2615	0	14857	827	2615	0	14857
14:10:00	733	2593	0	12997	733	2593	0	12997
14:15:00	729	2223	0	11503	729	2223	0	11503
14:20:00	606	1774	0	10335	606	1774	0	10335
14:25:00	522	1672	0	9185	522	1672	0	9185
14:30:00	395	1533	0	8047	395	1533	0	8047
14:35:00	335	1641	0	6741	335	1641	0	6741
14:40:00	263	1466	0	5538	263	1466	0	5538
14:45:00	230	1143	0	4625	230	1143	0	4625
14:50:00	175	900	0	3900	175	900	0	3900
14:55:00	137	742	0	3295	137	742	0	3295
15:00:00	89	607	0	2777	89	607	0	2777
15:05:00	63	664	0	2176	63	664	0	2176
15:10:00	42	574	0	1644	42	574	0	1644
15:15:00	18	485	0	1177	18	485	0	1177
15:20:00	15	447	0	745	15	447	0	745
15:25:00	16	351	0	410	16	351	0	410
15:30:00	6	252	0	164	6	252	0	164
15:35:00	0	158	0	6	0	158	0	6
15:40:00	0	4	0	2	0	4	0	2
15:45:00	0	1	0	1	0	1	0	1
15:50:00	2	1	0	2	2	1	0	2
15:55:00	0	1	0	1	0	1	0	1
16:00:00	0	1	0	0	0	1	0	0
Total	**45000**	**45000**	**0**		**45000**	**45000**	**0**	

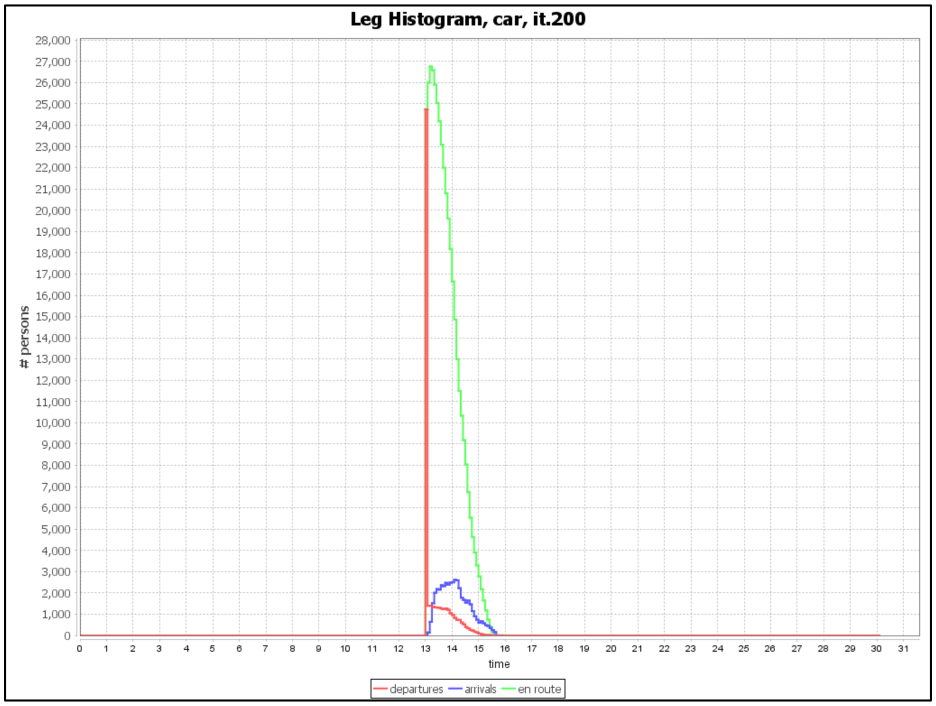


Figure 4.11: Iteration 200, Trip departures and arrival as well as en-route trips for scenario 2-A

4.2.2 Scenario 2-B: Capacity-, Mode-, Schedule-, and Time-Dependent Shortest-Path Model (45,000 Agents)

In this scenario, we used the same number of agents as the previous scenario (45,000), but allowed agents to use mixed modes of travel.

4.2.2.1 Run time of the simulation:

The overall simulation's run time was found to be 129 minutes and 16.53 seconds, and the average run time for all iterations is 37 seconds.

4.2.2.2 Average Utility Score:

The average utility score for the executed plan at the 200th iteration is 41.67 compared to 36.9 in the previous scenario (scenario 2-A). This indicates that, on average, agents made faster arrivals because less waiting times were encountered by them as a result of allowing them to use mixed-modes of travel. Figure 4.12 shows the average utility score per iteration for this scenario.

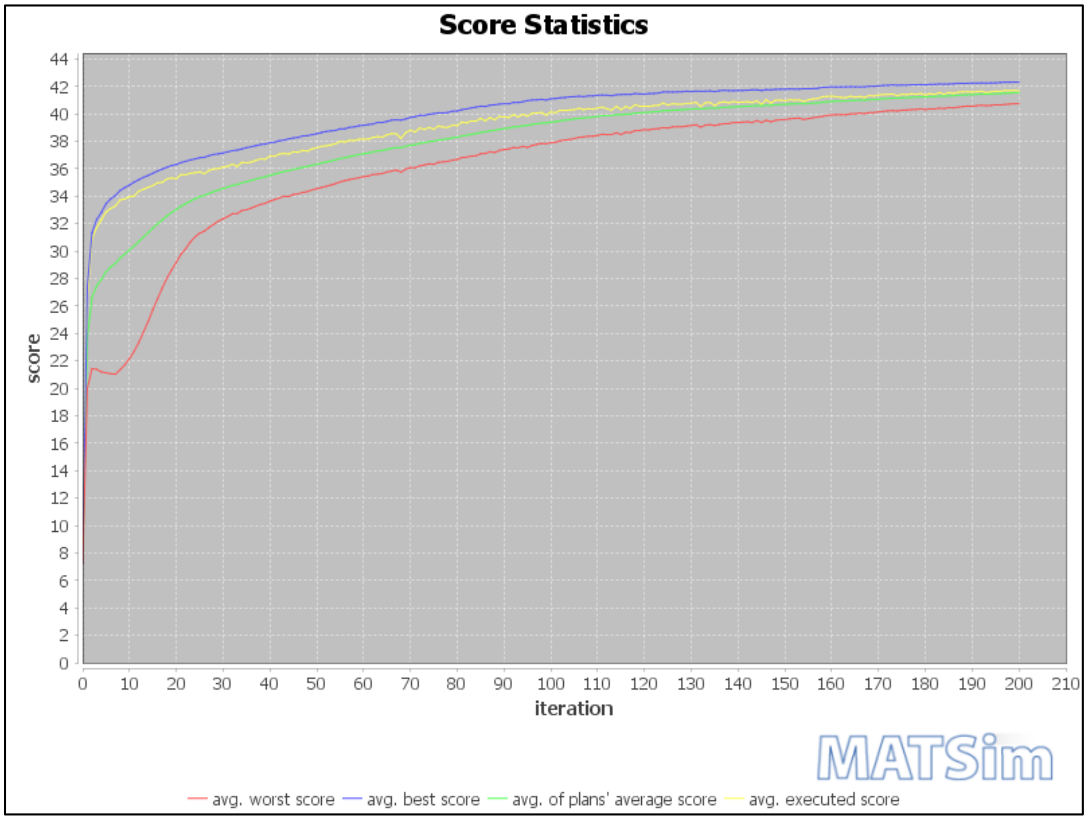


Figure 4.12: Average utility score per iteration (scenario 2-B)

We notice that the score is increasing in a more gradual manner than in scenario 2-A. This is because agents are taking more time, or iterations in MATSim's terminology, to learn and explore available routes since searching the public transit mode in this scenario is now enabled. We also notice that the average scores during first 20 iterations are higher compared to those of scenario 2-A because all agents in this scenario made arrival to their destinations throughout all iterations.

4.2.2.3 Average Trip Duration

In this scenario, the average trip duration in the 200th iteration was found to be 26 minutes and 38 seconds. This is about 19 minute and 5 seconds less than the previous scenario (scenario 2-A) where only car mode is allowed.

4.2.2.4 Average Trip Distances

The average trip distances for all the 15,000 agents at the 200th iteration is 17891.45 meters compared to 17976.4 meters in the previous simulation (scenario 2-A). Figure 4.13 summarizes average distances for all iterations.

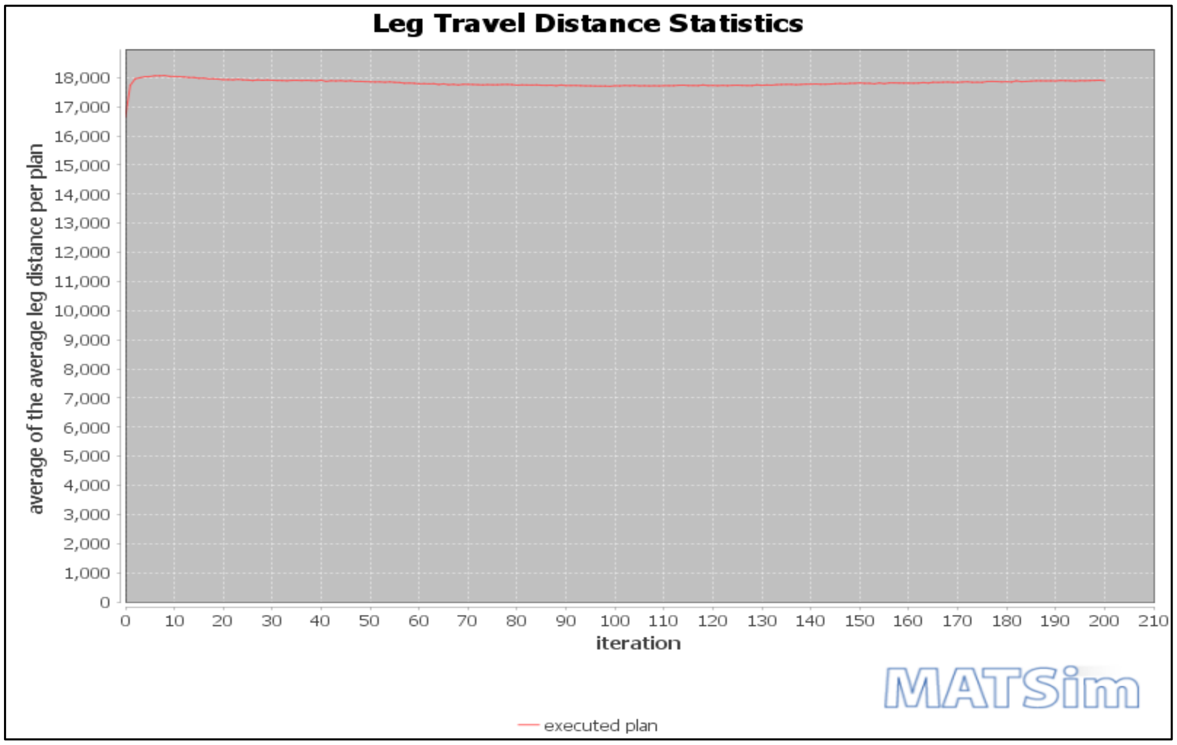


Figure 4.13: Average trips' distances for all the 200 iterations for scenario 2-B

4.2.2.5 Traffic Assignment and Mode Usage:

In this scenario, specifically in the 200th iteration, the car legs graph shows that 24,553 agents (54.56 %) were routed through car mode only, and the remaining agents (20,447 45.44%) were assigned mixed-modes of travel towards their destinations.

Figure 4.14 shows a histogram of iteration 200 for users who were assigned mixed mode routes, referred to in the graph as "pt", and figure 4.15 shows the same histogram for users who were routed through only car mode only throughout their trips in this scenario.

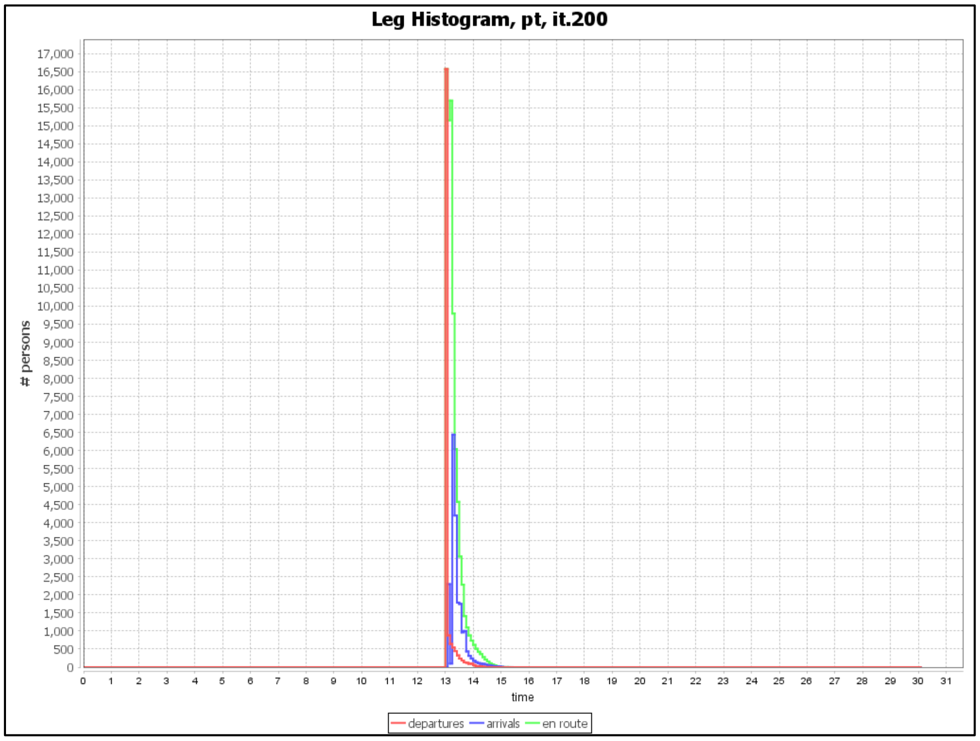


Figure 4.14: Iteration 200, Number of users who were assigned mixed mode routes (scenario 2-B)

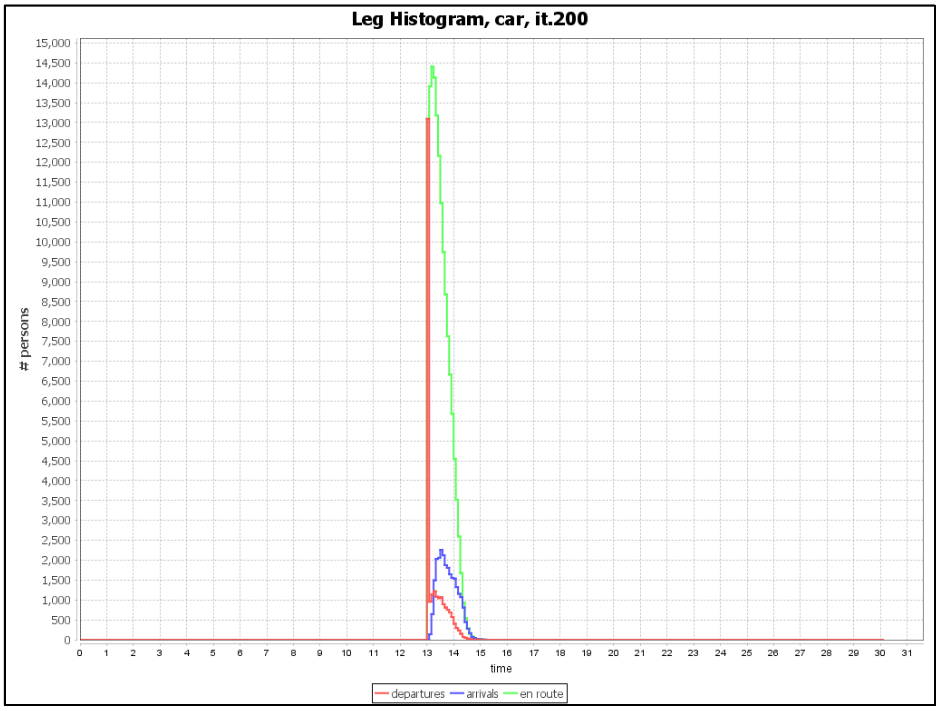


Figure 4.15: Iteration 200, number of users who were assigned only car mode (scenario 2-B).

Table 4.4 shows the numbers of agents who have finished departure, arrival, or are still in route, as well as their assigned mode of travel (either car only or mixed) for each time interval since the start of simulation until it finished. We notice that agents who were assigned either car only mode or mixed mode did not fill any link or a rail station up to its capacity. While none of the agents in both scenarios (2-A and 2-B) got stuck while en-route, the average duration to make an arrival was less in scenario 2-B when mixed-modes of travel was searched for the shortest-path. That is why the simulation in scenario 2-A lasted until 16:00 for the last agent to arrive to his destination while in scenario 2-B it lasted until 15:20. Again, this result present a strong case for implementing mixed-mode routing in the navigational applications.

Table 4.4: Numbers of agents who have finished departure, arrival, or are still in route based on their assigned mode of travel. (scenario 2-B)

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car	departures_pt	arrivals_pt	stuck_pt	en-route_pt
13:00:00	29672	0	0	29672	13093	0	0	13093	16579	0	0	16579
13:05:00	1836	2437	0	29071	955	137	0	13911	881	2300	0	15160
13:10:00	1778	741	0	30108	1134	642	0	14403	644	99	0	15705
13:15:00	1762	7938	0	23932	1218	1493	0	14128	544	6445	0	9804
13:20:00	1526	6234	0	19224	1081	2030	0	13179	445	4204	0	6045
13:25:00	1372	3848	0	16748	1045	2058	0	12166	327	1790	0	4582
13:30:00	1304	4010	0	14042	1066	2257	0	10975	238	1753	0	3067
13:35:00	1077	3088	0	12031	893	2123	0	9745	184	965	0	2286
13:40:00	942	2877	0	10096	810	1874	0	8681	132	1003	0	1415
13:45:00	874	2244	0	8726	752	1807	0	7626	122	437	0	1100
13:50:00	777	1957	0	7546	684	1644	0	6666	93	313	0	880
13:55:00	669	1797	0	6418	572	1555	0	5683	97	242	0	735
14:00:00	459	1712	0	5165	401	1536	0	4548	58	176	0	617
14:05:00	324	1460	0	4029	294	1323	0	3519	30	137	0	510
14:10:00	262	1267	0	3024	232	1154	0	2597	30	113	0	427
14:15:00	173	1160	0	2037	149	1071	0	1675	24	89	0	362
14:20:00	79	903	0	1213	68	810	0	933	11	93	0	280
14:25:00	52	515	0	750	49	444	0	538	3	71	0	212
14:30:00	31	331	0	450	28	274	0	292	3	57	0	158
14:35:00	16	216	0	250	15	164	0	143	1	52	0	107
14:40:00	8	105	0	153	7	70	0	80	1	35	0	73
14:45:00	2	64	0	91	2	43	0	39	0	21	0	52
14:50:00	4	45	0	50	4	21	0	22	0	24	0	28
14:55:00	1	28	0	23	1	10	0	13	0	18	0	10
15:00:00	0	13	0	10	0	7	0	6	0	6	0	4
15:05:00	0	6	0	4	0	4	0	2	0	2	0	2
15:10:00	0	2	0	2	0	2	0	0	0	0	0	2
15:15:00	0	1	0	1	0	0	0	0	0	1	0	1
15:20:00	0	1	0	0	0	0	0	0	0	1	0	0
Total	45000	45000	0		24553	24553	0		20447	20447	0	

4.3 Results of Scenario 3: 75,000 Agents

In this scenario, we ran the simulation in a similar fashion to the previous scenarios but with 75,000 agents. The results of this scenario (both car mode and mixed-modes) are discussed in the following subsections.

4.3.1 Scenario 3-A: Time-Dependent, Capacity-Aware, Car-Mode, Shortest-Path Model

4.3.1.1 Average Utility Score:

The average utility score for all executed shortest paths in this scenario converged to around 21.34. we notice that this score is lower than the previous scenarios (scenario 1-A and 2-A). Figure 4.16 shows the average utility score per iteration for this scenario. This indicates that more waiting times have been encountered by agents. In this scenario, convergence started approximately after the 181st iteration in which the score is 20.5. However, this graph is not as smooth as in the previous scenarios for car-only mode. This can be attributed to the large number of agents that cause more traffic jams across more links than in the previous cases; and thus, more re-planning and re-routing is required for agents in every iteration. We saw such trend at the first 20 iterations in scenario 2-A before the curve started to smooth out but we did not see this trend in scenario 1-A.

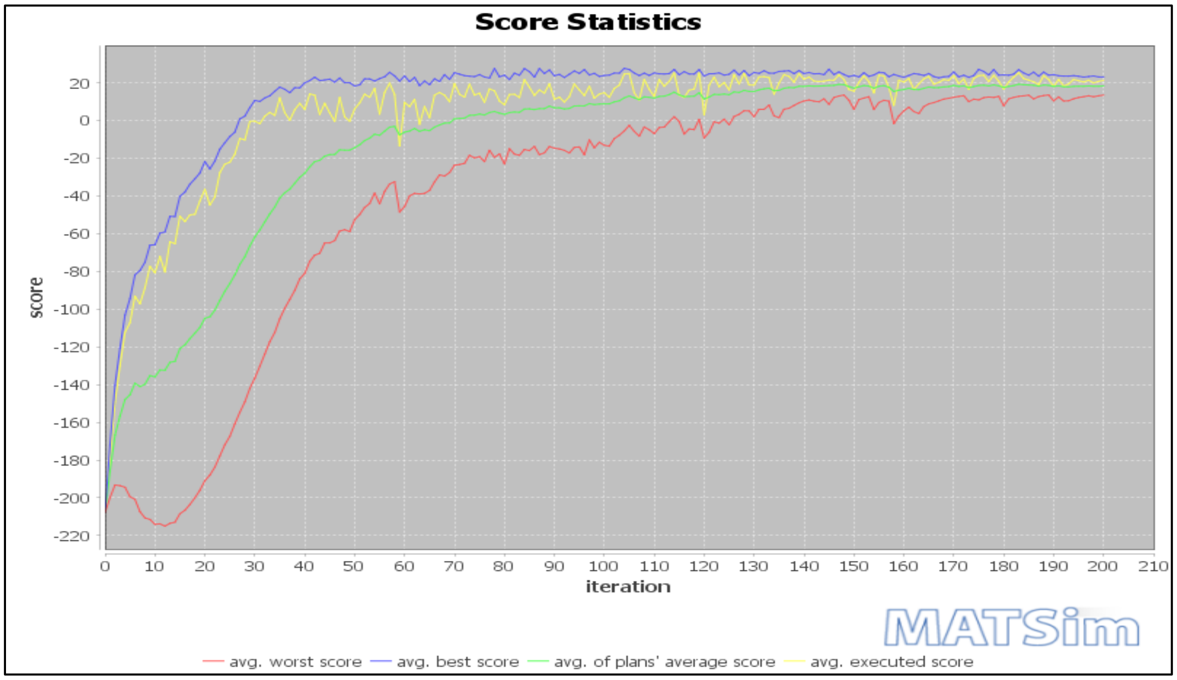


Figure 4.16: Average utility score per iteration (scenario 3-A)

We notice in this scenario that the score increased from about -200 around 0 after 30 iterations. We saw such behavior in scenario 2-A but it is more sever here because of lager number of agents. Figure 4.17 shows the traffic assignment of this scenario's first iteration. We notice that the simulation ended while about 40, 000 agents are still en-route. As agents learn through iterations, they will adjust departure times and selected better routes, and as a result, more agents will make an arrival causing the average utility score to improve.

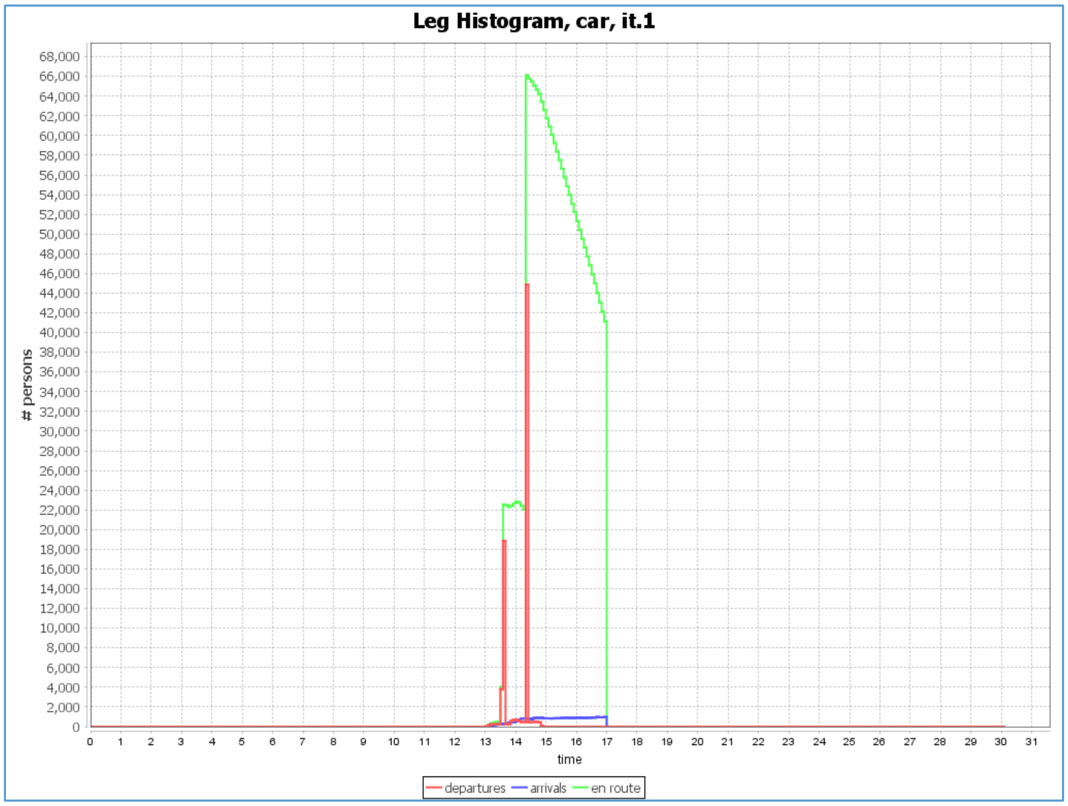


Figure 4.17: Iteration 1, number of users who were assigned only car mode (scenario 3-A).

4.3.1.2 Run-time of the simulation:

The simulation was executed for the whole 200 iterations. The total run-time for this simulation is: 190 minutes and 37.41 seconds. The average run-time for all iterations was found to be around 55 seconds.

4.3.1.3 Average Trip Duration

In this scenario, the average trip duration after the 200th iteration has been found to be: 75 minutes and 7 seconds. It is larger than the average time in the previous car-mode only scenarios (scenario 1-A, and scenario 2-A). moreover, the number of simulated trips is 73.363 trips (or agents). The remaining agents were not able to make an arrival to their final destination because they were stuck in route or waiting to leave their locations of origin until time 17:00, which is the simulation's end time.

4.3.1.4 Average Trip Distances

For this specific scenario, the average distance for the 75,000 trips after the 200th iteration is: 17594.33 meters. Figure 4.18 shows the average distances for all the 200 iterations. Although it is less than the previous car-mode only scenarios (scenarios 4.1.1 and 4.2.1), yet it is not an indicative measure of any improvement in this scenario because 8 agents here could not make a departure and another 1663 agents did not make an arrival. Thus, a total of 1671 agents were not counted in calculating the average distance here[28].

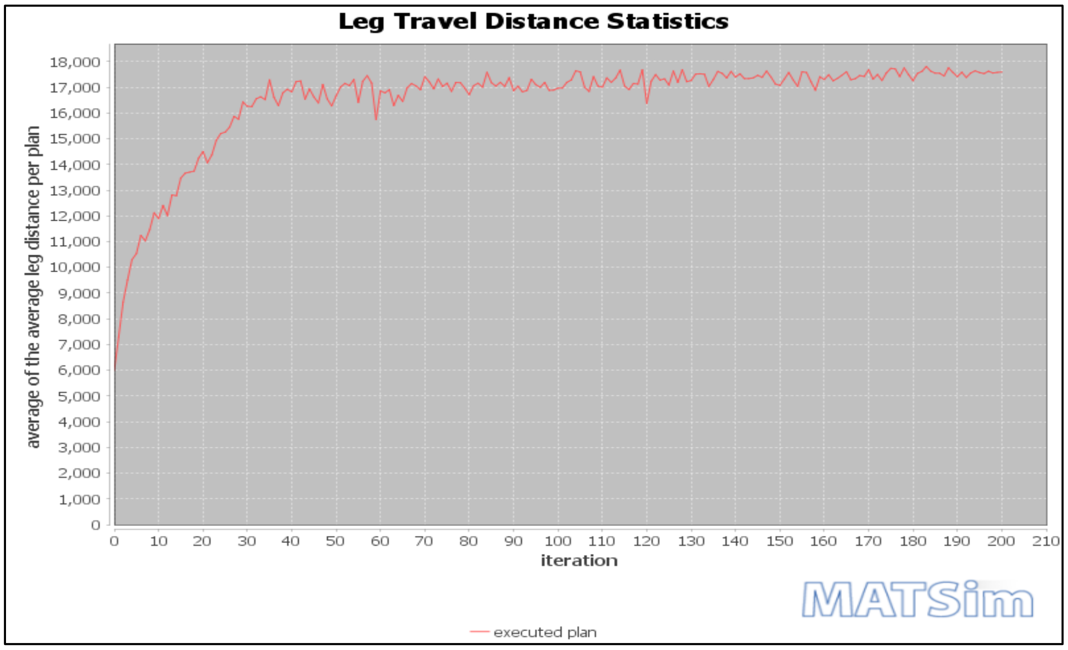


Figure 4.18: Average trips' distances for all the 200 iterations (scenario 3-A)

4.3.1.5 Traffic Assignment and Mode Usage:

Table 4.5 below shows the numbers of agents who have finished departure, arrival, or are still in route. We notice that all routes are free-flowing for agents at their start time of travel or during

[28] It should be noted though that the routes for these agents have been computed. But the time duration of the simulation ended before these routes have been executed. While the distance and trip-duration calculation are affected in this situation, the run-time is not.

their trips. Although routes are reported here to be free-flowing, the fact that average trip time is larger than when the number of agents is 15,000 indicates that agents are forced to travel at lower speed here. Figure 4.19 shows a histogram for trip departures and arrival as en-route trips for this scenario. Simulation ended at 17:00 although some agents are either "stuck" in the network or have not yet departed their origins due to queues at the "exit gate" of their locations of origin.

Table 4.5: Numbers of agents who have finished departure, arrival, or are still in route for (scenario 3-A)

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car
13:00:00	38997	0	0	38997	38997	0	0	38997
13:05:00	2287	134	0	41150	2287	134	0	41150
13:10:00	2236	641	0	42745	2236	641	0	42745
13:15:00	2077	1483	0	43339	2077	1483	0	43339
13:20:00	1973	1835	0	43477	1973	1835	0	43477
13:25:00	1917	2005	0	43389	1917	2005	0	43389
13:30:00	1807	2038	0	43158	1807	2038	0	43158
13:35:00	1667	2087	0	42738	1667	2087	0	42738
13:40:00	1501	2107	0	42132	1501	2107	0	42132
13:45:00	1517	2112	0	41537	1517	2112	0	41537
13:50:00	1474	2159	0	40852	1474	2159	0	40852
13:55:00	1371	2247	0	39976	1371	2247	0	39976
14:00:00	1327	2231	0	39072	1327	2231	0	39072
14:05:00	1221	2274	0	38019	1221	2274	0	38019
14:10:00	1167	2266	0	36920	1167	2266	0	36920
14:15:00	1094	2231	0	35783	1094	2231	0	35783
14:20:00	1041	2200	0	34624	1041	2200	0	34624
14:25:00	916	2351	0	33189	916	2351	0	33189
14:30:00	836	2306	0	31719	836	2306	0	31719
14:35:00	857	2317	0	30259	857	2317	0	30259
14:40:00	768	2236	0	28791	768	2236	0	28791
14:45:00	692	2285	0	27198	692	2285	0	27198
14:50:00	647	2289	0	25556	647	2289	0	25556
14:55:00	577	2215	0	23918	577	2215	0	23918
15:00:00	544	1895	0	22567	544	1895	0	22567
15:05:00	522	1843	0	21246	522	1843	0	21246
15:10:00	465	1801	0	19910	465	1801	0	19910
15:15:00	442	1836	0	18516	442	1836	0	18516
15:20:00	370	1767	0	17119	370	1767	0	17119
15:25:00	367	1829	0	15657	367	1829	0	15657
15:30:00	332	1835	0	14154	332	1835	0	14154
15:35:00	257	1860	0	12551	257	1860	0	12551
15:40:00	228	1783	0	10996	228	1783	0	10996
15:45:00	224	1685	0	9535	224	1685	0	9535
15:50:00	171	1583	0	8123	171	1583	0	8123
15:55:00	174	1178	0	7119	174	1178	0	7119
16:00:00	160	991	0	6288	160	991	0	6288
16:05:00	125	954	0	5459	125	954	0	5459
16:10:00	118	839	0	4738	118	839	0	4738
16:15:00	101	722	0	4117	101	722	0	4117
16:20:00	88	582	0	3623	88	582	0	3623
16:25:00	66	529	0	3160	66	529	0	3160
16:30:00	57	474	0	2743	57	474	0	2743
16:35:00	54	302	0	2495	54	302	0	2495
16:40:00	45	261	0	2279	45	261	0	2279
16:45:00	36	266	0	2049	36	266	0	2049
16:50:00	46	247	0	1848	46	247	0	1848
16:55:00	33	251	0	1630	33	251	0	1630
17:00:00	0	1	1629	0	0	1	1629	0
Total	**74992**	**73363**	**1629**		**74992**	**73363**	**1629**	

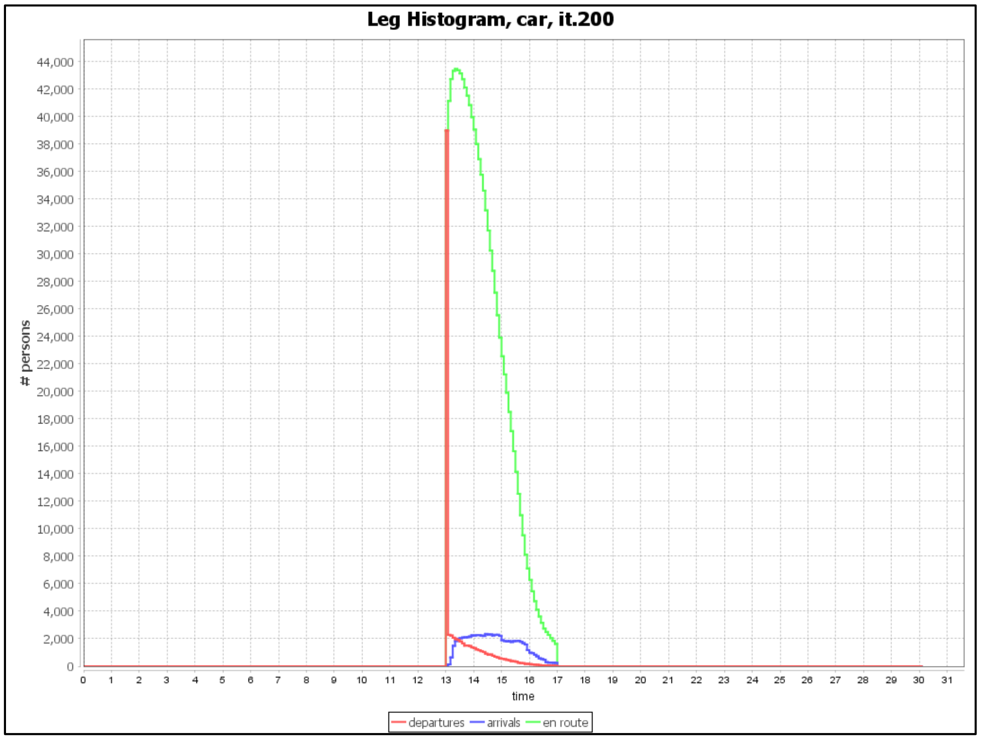


Figure 4.19: Number of users' departures and arrival as well as those who are en-route. (scenario 3-A)

4.3.2 Scenario 3-B: Capacity-, Mode-, Schedule-, and Time-Dependent Shortest-Path Model (75,000 Agents)

In this scenario, we used a population file that has same number of agents as the previous one (scenario 3-A), but the settings in both the population and configuration files are defined to allow agents to use mixed modes of travel. In this section, we report the same metrics as in the previous scenario (car mode only).

4.3.2.1 Run time of the simulation:

The overall simulation's run time was found to be 177 minutes and 38.94 seconds, and the average run time for all iterations is: 51 seconds. It is interestingly less than the car-mode only scenario (scenario 3-A). this is due to the nature of the computation done by the co-evolutionary algorithm of MATSim; more agents, means more traffic and thus more interactions between agents and hence re-planning of their trips. When we have less traffic, because some agents selected other modes in part of their trips, the re-planning process of the re-routing took less run-time.

The result found in this scenario is not expected, but very encouraging. It hints that the major computational issues for such settings of routing models are linked to the number of users who query the navigational applications and their preferences (utility function).

4.3.2.2 Average Utility Score:

The average utility score for the executed plan at the 200th iteration is 40.98 compared to 21.34 in the previous scenario (scenario 3-A). This is a significant improvement in the travel experience of agents who used mixed-mode of travel. Figure 4.20 shows the Average utility score per iteration for this scenario.

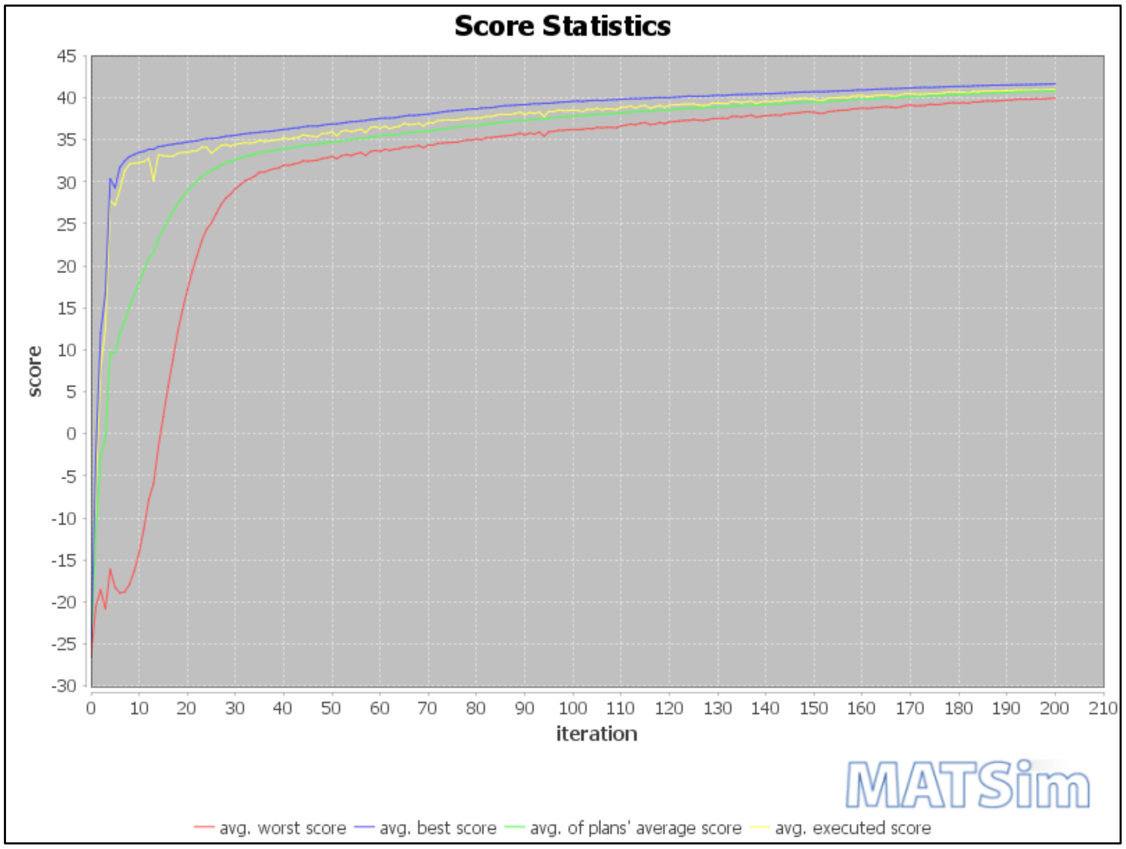


Figure 4.20: Average utility score per iteration (scenario 5.3.2)

4.3.2.3 Average Trip Duration

In this scenario, the average trip duration in the 200th iteration was found to be 27 minutes and 53 seconds. This is about 47 minute and 14 seconds less than the previous scenario (scenario 3-A) where only car mode is allowed. Again, this presents a strong case for why users should consider using a route-guidance application that searches mixed modes of travel for the shortest path.

4.3.2.4 Average Trip Distances

The average trip distances for all the 75,000 agents at the 200th iteration is 18229.66 meters. Figure 4.21 shows average distances for all iterations.

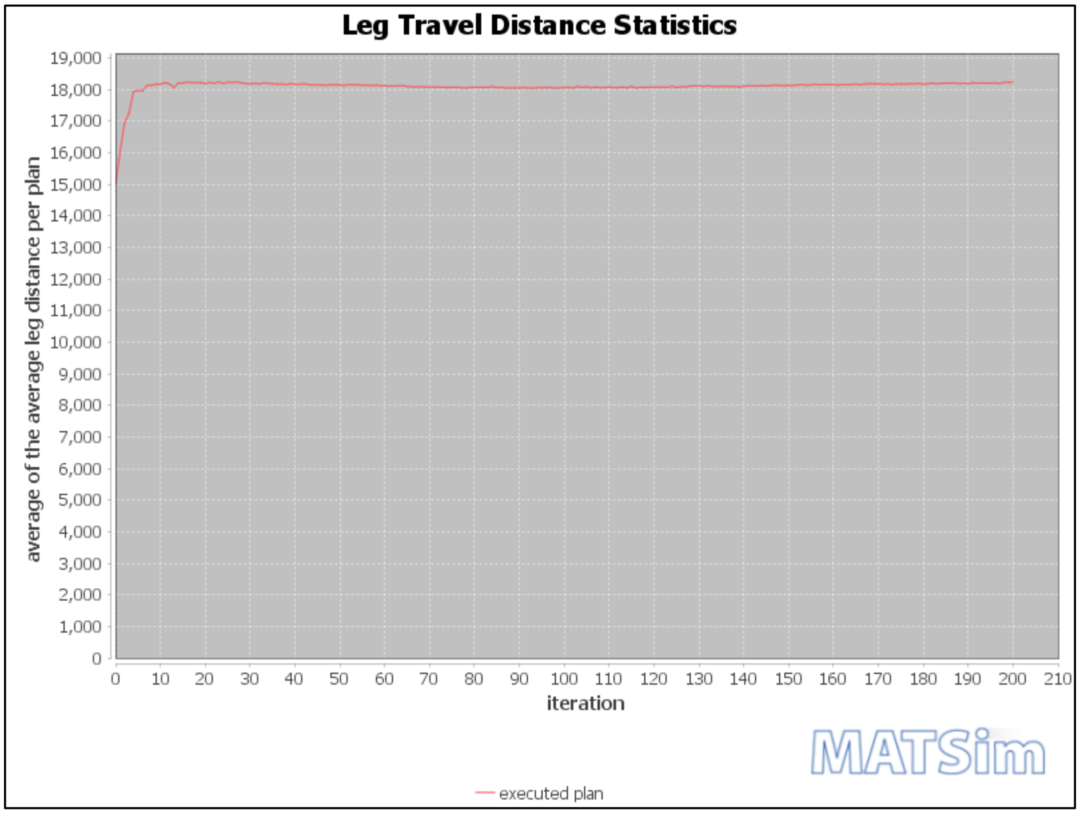


Figure 4.21: Average trips' distances for all the 200 iterations (scenario 3-B)

4.3.2.5 Traffic Assignment and Mode Usage:

In this scenario, specifically in the 200th iteration, the car legs graph shows that 27682 agents (36.91 %) were routed through car mode only, and the remaining agents (47318 agents, 63.09%) were assigned mixed-modes of travel towards their destinations.

Figure 4.22 shows a histogram of iteration 200 for users who were assigned mixed mode routes referred to in the graph as "pt", and figure 4.23 shows a histogram of the same iteration for users who were routed through only car mode only throughout their trips in this scenario.

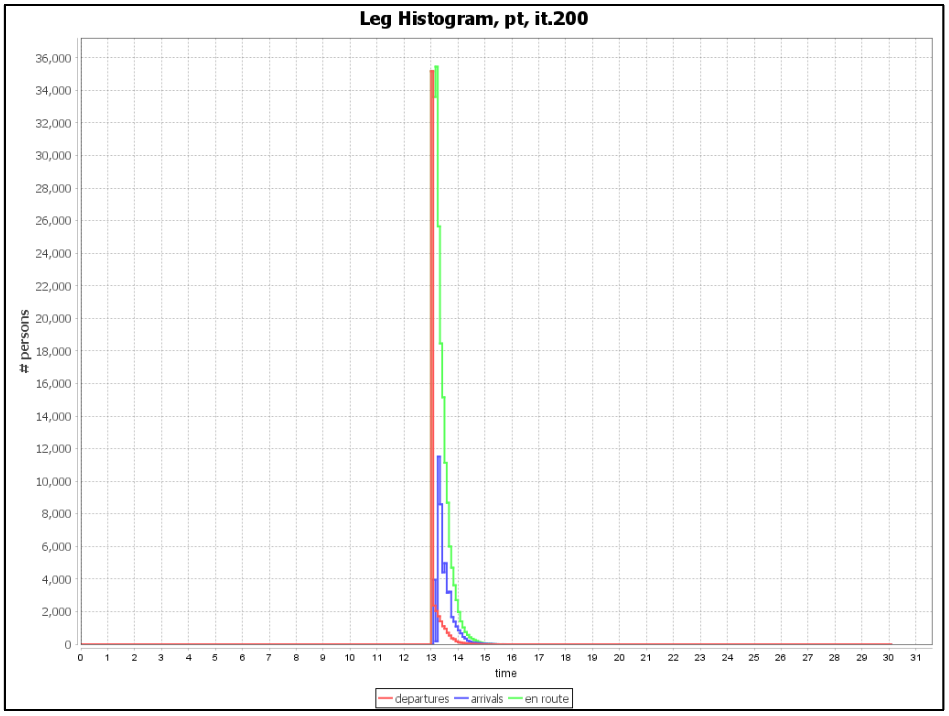


Figure 4.22: iteration 200, Number of users who were assigned mixed mode routes (scenario 4.3.2)

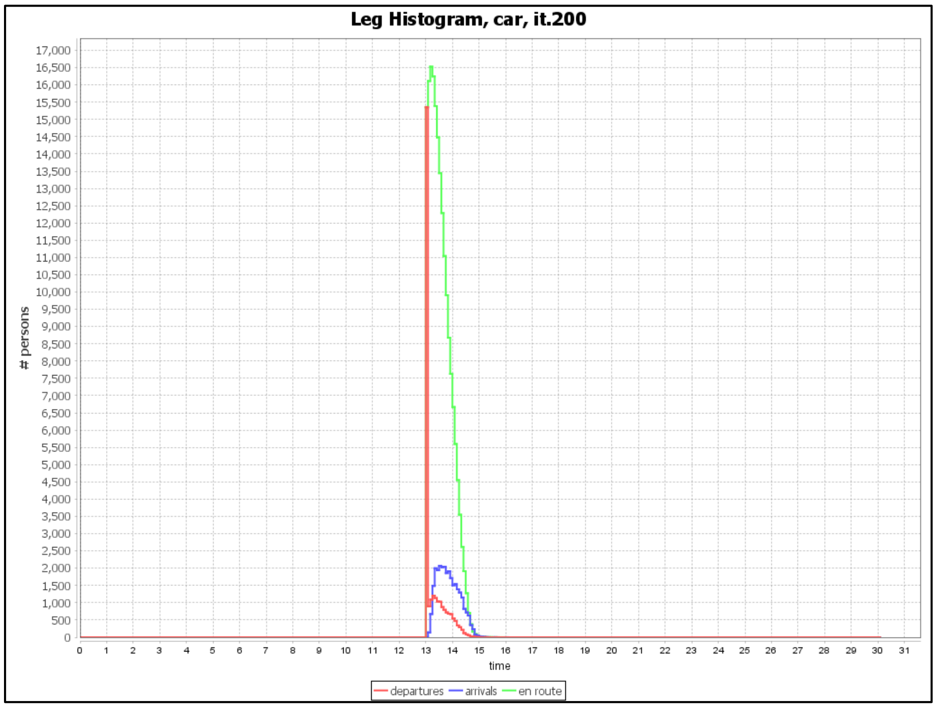


Figure 4.23: iteration 200, number of users who were assigned car-mode only (scenario 4.3.2)

Table 4.6 shows the numbers of agents who have finished departure, arrival, or are still in route, as well as their assigned mode of travel (either car only or mixed) for each time interval since the start of simulation until it finished. We notice that all routes are free-flowing for agents who were assigned either car only mode or mixed mode. While traffic in both scenarios (2-A and 2-B) was shown to be free-flowing, the average arrival duration was faster in scenario 2-B when mixed-modes of travel was searched for the shortest-path. That is why the simulation in scenario 2-A lasted until 16:00 for the last agent to arrive to his destination while in scenario 2-B it lasted until 15:20. Again, this result presents a strong case for implementing mixed-mode routing in the navigational applications. Table 4.7 summarizes all the three scenarios' results.

Table 4.6: Numbers of agents who have finished departure, arrival, or are still in route based on their assigned mode of travel (scenario 3-B)

time	departures_all	arrivals_all	stuck_all	en-route_all	departures_car	arrivals_car	stuck_car	en-route_car	departures_pt	arrivals_pt	stuck_pt	en-route_pt
13:00:00	50530	0	0	50530	15354	0	0	15354	35176	0	0	35176
13:05:00	3267	4094	0	49703	896	139	0	16111	2371	3955	0	33592
13:10:00	3126	849	0	51980	1088	671	0	16528	2038	178	0	35452
13:15:00	2923	13005	0	41898	1206	1487	0	16247	1717	11518	0	25651
13:20:00	2541	10588	0	33851	1138	1998	0	15387	1403	8590	0	18464
13:25:00	2147	6356	0	29642	1037	1943	0	14481	1110	4413	0	15161
13:30:00	1980	7039	0	24583	1030	2067	0	13444	950	4972	0	11139
13:35:00	1582	5189	0	20976	877	2032	0	12289	705	3157	0	8687
13:40:00	1330	5263	0	17043	795	2040	0	11044	535	3223	0	5999
13:45:00	1088	3526	0	14605	721	1857	0	9908	367	1669	0	4697
13:50:00	992	3298	0	12299	682	1913	0	8677	310	1385	0	3622
13:55:00	847	2807	0	10339	669	1715	0	7631	178	1092	0	2708
14:00:00	669	2358	0	8650	543	1504	0	6670	126	854	0	1980
14:05:00	561	2210	0	7001	470	1541	0	5599	91	669	0	1402
14:10:00	425	1841	0	5585	343	1389	0	4553	82	452	0	1032
14:15:00	352	1638	0	4299	293	1296	0	3550	59	342	0	749
14:20:00	257	1367	0	3189	216	1150	0	2616	41	217	0	573
14:25:00	139	990	0	2338	120	822	0	1914	19	168	0	424
14:30:00	101	824	0	1615	84	721	0	1277	17	103	0	338
14:35:00	69	722	0	962	57	629	0	705	12	93	0	257
14:40:00	30	418	0	574	24	352	0	377	6	66	0	197
14:45:00	26	303	0	297	22	234	0	165	4	69	0	132
14:50:00	4	115	0	186	4	75	0	94	0	40	0	92
14:55:00	8	77	0	117	7	50	0	51	1	27	0	66
15:00:00	2	48	0	71	2	25	0	28	0	23	0	43
15:05:00	1	28	0	44	1	17	0	12	0	11	0	32
15:10:00	0	17	0	27	0	4	0	8	0	13	0	19
15:15:00	2	16	0	13	2	6	0	4	0	10	0	9
15:20:00	0	5	0	8	0	0	0	4	0	5	0	4
15:25:00	0	4	0	4	0	1	0	3	0	3	0	1
15:30:00	0	1	0	3	0	1	0	2	0	0	0	1
15:35:00	0	2	0	1	0	2	0	0	0	0	0	1
15:40:00	1	1	0	1	1	0	0	1	0	1	0	0
15:45:00	0	0	0	1	0	0	0	1	0	0	0	0
15:50:00	0	0	0	1	0	0	0	1	0	0	0	0
15:55:00	0	0	0	1	0	0	0	1	0	0	0	0
16:00:00	0	1	0	0	0	1	0	0	0	0	0	0
Total	75000	75000	0		27682	27682	0		47318	47318	0	

Table 4.7: Summary of simulations' results

Number of Agents	Scenario	Average Utility Score (absolute)	Run time (seconds)	Average Trip Duration Minutes: Seconds	Average Trip Distances (meters)	Mode Usage	Executed trips	Last Arrival Time
15,000	Scenario 1-A (car mode only)	43.7	11	23:59	17549.4	Car only	All	2:45 PM
	Scenario 1-B (Mixed mode)	43.85	12	21: 21	17516.43	82.047% Car 17.953% Mixed	All	2:55 PM
45,000	Scenario 2-A (car mode only)	36.9	36	45: 43	17976.4	Car only	All	4:00 PM
	Scenario 2-B (Mixed mode)	41.67	37	26:38	17891.45	54.56 % Car 45.44% Mixed	All	3:20 PM
75,000	Scenario 3-A (car mode only)	21.34	55	75:07	17594.33	Car only	73,371	Simulation ended at 5:00 PM while 1629 agents are either stuck in the network or did not depart
	Scenario 3-B (Mixed mode)	40.98	51	27:53	18229.66	36.91 % Car 63.09% Mixed	All	4:00 PM

To sum up, we found that the computational complexity of both shortest path models discussed in this study is almost the same. This was shown by similar run times for both models when they were examined via simulation using the same set of origin-destination points on the same network. In our reporting of the run time, we focused on the iteration's average run time. We explained in section 4.1.1.2 (the first scenario) that MATSim does more computational tasks than just routing and re-routing the synthetic travelers. For example, MATSim's simulations include writing the results of each iteration to prespecified folder, and later writing the aggregate results of all iterations to another folder. MATSim's computational run time for each iteration can be broken down into more than 20 tasks; where. These tasks and their respective run time are reported in a text file called Stop Watch. However, these tasks are not the same tasks that would be expected from multimodal or unimodal navigational systems.

In figure 3.7, we saw that a MATSim's iteration goes through three main stages: execution, scoring, and replanning. These stages are important in the "simulation" world where a researcher or a traffic planner aims to understanding optimal traffic assignment from a "user equilibrium" perspective. However, navigation systems would not require implementing these stages using large number of iterations. In all scenarios, we reported the average run-time criterion. But, routing processes of a real-world urban environment of a portable navigational device or a cell phone, does not include all the tasks that are performed in MATSim typical traffic assignment simulations. For example, the execution stage in which agents interact and compete over the network resources would be replaced be real-time traffic feeds if MATSim environment is used as a platform for a navigational system. In tables 4.8 and 4.9 we show a breakdown of simulations' average run-time of scenarios 3-A and 3-B.

Table 4.8: Average run-time breakdown of scenario 3-A

Scenario 3-A	Replanning	Execution	Scoring	Iteration (total)
Average Run-Time (seconds)	26.7	26.1	00.01<	54.9

Table 4.9: Average run-time breakdown of scenario 3-A

Scenario 3-A	Replanning	Execution	Scoring	Iterations (total)
Average Run-Time (Seconds)	36.9	12.5	00.01<	51.4

We notice that the total average run-time that we reported is larger than the sum of the average run-time of the three stages of a MATSim iteration. This is due to the fact that we did not include times encountered to fire an iteration after the previous iteration ended or the time needed to write iterations' results to a desk. As mentioned earlier, there are about 20 computational tasks, including the main ones reported in tables 4.8 and 4.9 that all together contribute to the reported iterations average run-time. We notice also that in both scenarios, scoring takes less than 0.01 seconds.

We also found that links' capacities information is very important for a better routing results. In our scenarios, we had issues with routing through the shortest path during the first iterations. Our scenarios were designed to route different number of users from 30 locations within the same business district towards the north and east sides of the city. This caused traffic jams that delayed some users in scenarios 2 and 3.

The more sever delays were at the origin locations themselves because, as mentioned earlier in section 3.2, MATSim uses spatial queues to model vehicles that move on links. Such technique, which is realistic for modeling vehicles' movement, implies that vehicles occupy space and whenever a link is filled with vehicles no more following vehicles will enter this link. That is why trip durations between the same origin and destination locations were different at different traffic loads, especially in the case when the network allows only for car mode of travel.

To better understand the interplay between traffic loads and trips' duration, we need to look into the origin locations and how agents depart these locations towards their destinations. Originally, we distributed the 30 origin locations as points within one area, the downtown. Many of these origin locations are located on a one-lane street. In addition, some of these locations exist on the same street link. We also gave the simulation a time frame of four hours while agents were set to depart as soon as they can within that time frame. When we assigned 500 agents to depart each origin location, the 15,000 agents' scenario, most agents finished their trips within 45 minutes with an average trip duration of about 23 minutes. We noticed that this time decreased by only 2 minutes as we enabled the use of mixed-modes of travel for the same number of agents. Delays on the network became more severe as we assigned larger number of users to depart the origin locations in scenarios 2 and 3. In addition, time savings on average trip durations also became more significant in scenarios 2 and 3.

This happened because capacity constraints of street network do not allow for flowing at the free-flow speed on a link when number of agents exceed the capacity of flow of a street link. The flow capacity of a link was specified as 1,500 vehicles per hour for one-lane streets, 3,000 vehicles per hour for two-lanes streets, and 4,500 vehicles per hour for three-lanes streets. Since most origin locations are located on a one-lane or in two-lanes streets links, vehicles needed to queue up not only on the same departure location, but also on nearby links cause major spill-backs when the number of agents was increased in scenarios two and three.

Therefore, it should not be surprising that we saw time savings as we introduced the public transit search functionality. This allowed some agents to use the additional network resources, namely the added public transit network, for a faster arrival. It also freed the network from vehicles at some critical intersections. The free space of the street network was then used by many who had better routes' plans by using the car mode only. Figure 4.24 shows a screenshot of the simulation video for scenario 5.3.1.

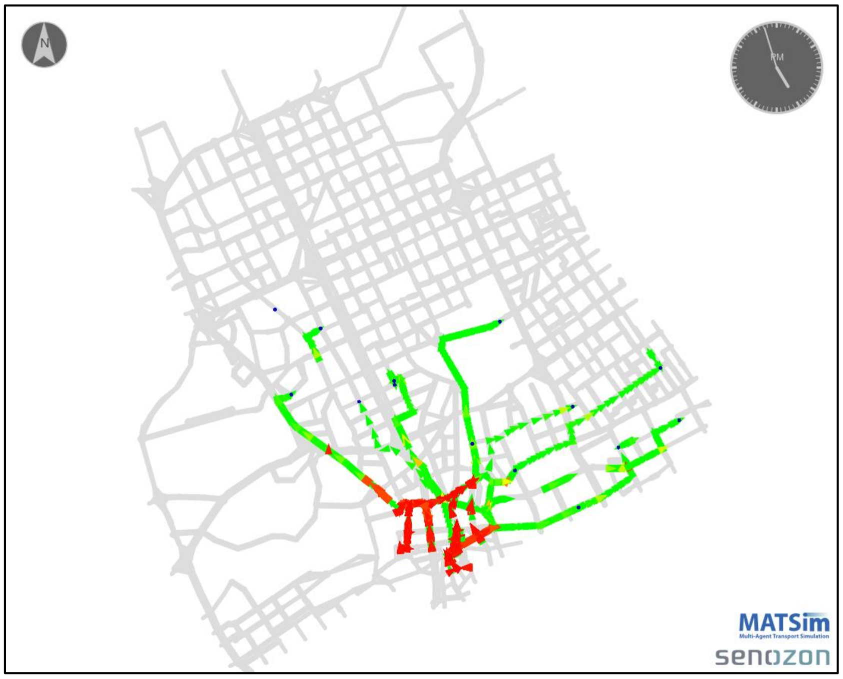


Figure 4.24: Screenshot of the simulation video for scenario 5.3.1. We notice major traffic jams are formed in the links leading to the major (south-north) traffic corridor of the city.

CHAPTER 5

CONCLUSIONS

5.1 Summary and Conclusions

In this study, we tried to gain a new insight on how computationally complicated the multimodal route finding compared to the widely used single mode route finding models. We ran several traffic simulations in which all agents were routed using the famous least-cost algorithm by Dijkstra. To examine the re-routing functionality, we designed a simulation in which agents were re-routed by the software after almost all iterations of the simulation. This process happened because the agents learned that some routes are less congested and thus can be used for a faster arrival. This dynamic shortest-path guidance process is similar in its functionality to route-guidance process of the single-mode routing applications discussed in the literature review section of this study.

This route-guidance process was tested in this study for different loads of traffic (15,000, 45,000, and 75,000 agents) in two scenarios: one in which route-guidance can only route car-mode users only, and the other in which the route-guidance can route users who are flexible to use any mode of travel as long as it gets them to their destination faster. To report realistic conclusions and insights, we modeled the capacity and scheduling constraints of the public transportation mode.

After simulating the two scenarios for the three different traffic loads, we found that the computational complexity in the case of single mode and multimode route-guidance models is almost the same. Thus, the hypothesis that an optimization model for route-guidance application that incorporates both transit mode and car mode in searching for the shortest path will suffer long-run times and hence will not suit the computational power of a portable navigational device or a cell phone could be refuted by the results of this experiment.

Moreover, we’ve seen that the more agents we have, the more time savings in trip-duration will result from adopting multimodal route-guidance. In addition, we saw in Scenario 3 that, when a too-large traffic load causes traffic jams at several points of the network, more interactions occur between agents. This process required more computing processes to re-route the agents and thus more run time to compute the shortest path. Such a pattern needs to be captured by the route-guidance application and the re-routing, or re-planning, should be done in a way that does not cause additional computational complexity.

This research tested whether it is possible to construct a route-finding system that allows for a multimodal travel option (with capacity and scheduling constraints) that takes into account the temporal aspect of travel network behaviors, while limiting the computational requirements to a level at which the system could be implemented in the real-world urban environment of a portable navigational device or a cell phone.

We found that this was possible in a simulation environment that was designed to capture most of the complexities of such system. Thus, this research should be followed by building such a functionality to close this gap in the existing navigational applications. Several implementation issues may arise, but we have demonstrated that the computational complexities and demands will not be major barriers. We also found that such a system would provide, on average, closer-to-optimal routes compared to the standard car-mode, capacity-aware, shortest-path, Dijkstra-based model.

This study succeeded in answering the research questions and in achieving other important contributions as well. These contributions include the following:

1- Preparing a coordinate system conversion module for our study area that can be used by other researchers to create traffic simulation scenarios using MATSim.
2- Preparing and documenting a method to convert ArcGIS data into a MATSim Network XML file. A better contribution would be to create an ArcGIS-based tool that performs this conversion. However, many challenges were encountered, the least is the topological difference in the structure of streets feature classes in ArcGIS (line feature class), whereas the MATSim street network is composed of links and nodes. This contribution helps ArcGIS users to use their data in running dynamic traffic simulations using MATSim.
3- We presented a simple, yet effective transitioning stage between modes. The walkways line feature class is a logical connection between car and train modes, and it was easy to

model them within MATSim. It is suitable to model switching modes in park-and-ride as well as drop-off-and-ride situations. We specified walking speed on these links and we modeled the scheduling and capacities of transit mode. This data model, which is mostly MATSim's network data structure, proved to be simple yet adequate to handle multimodal routing requirements at a very reasonable computational cost.

On the other hand, we could not achieve a very important task that would make this study (and the simulation software) more useful within the time limits of this study. This task is converting the selected routes that MATSim show as a text file describing executed routes into a shapefile format. This task would facilitate a more interesting research task of comparing geometric similarities of the resulting routes. How we plan to pursue this task as well as other future research tasks are discussed in the next section.

5.2 Future Work

As seen in the implementation and results sections, the multiagent traffic simulation software that we used provided great functionality and flexibility to test our model. However, it has some limitations too. The fact that the software we used, MATSim, is an open source that makes further improvements fairly easy. In a few points, I will summarize my personal views on how this study could be extended.

1- At the end of this study, we suggest that the best answer the research questions is to use MATSim's environment, which would be a suitable testing ground, to actually test the quality of a multimodal routes for real trips between origin and destination locations for a sample population of Riyadh city. It is recommended that traffic volume data is used to create a more realistic daily traffic pattern, and then run the multimodal route-guidance for the selected, or surveyed, sample. Such study will provide a more realistic analysis of trips' time savings resulting from using multimodal traffic navigational systems.

2- Limitation in presenting the average executed trip duration per iteration: While the software shows improvements in utility score per iteration, it does not show how this is reflected in time savings. Doing so will help in comparing the utility score graph with the trip durations to confirm that agents eventually cannot improve their selected routes.

3- MATSim's default settings of writing the output agent's plans, or selected routes, is in a text file in which the link IDs of the selected route are shown. However, this setting could be changed to let MATSim write this output as a KML file. KML files can be converted into shapefile format. This could be a starting point for performing routes' geometric similarities analysis of the simulation results. However, this task is not trivial when we have many agents having the same departure and arrival points. The "mode" route that represents the most followed route for every pair of origin and destination points can be used to compare the followed path in the case of "loaded" network against the static shortest path to study the geometric similarities in both cases.

4- This study was completed using the Dijkstra's algorithm as a base-algorithm for the least-cost calculations. Luckily, MATSim has other built-in shortest-path algorithms that we could use for our models and then compare the same metrics, especially the run time. This will help us in gaining insight into how run time would differ at different loads of traffic when we use different shortest-path algorithms. Depending on the produced solutions' quality, we may prefer using a different shortest path algorithm as a basis for building a route-guidance application.

5- This study suggest that we could measure the city's "work commuting" trips time durations, assuming we can derive the proper data from the census data. Therefore, a comparison of such metric in the case of using car mode of travel with mixed modes of travel can provide an important insight on how the city's mobility could be improved.

6- Last but not lease, this study suggests that it is possible to develop a mixed-mode route-guidance application that requires a reasonable computational cost. Therefore, we recommend building such functionality.

APPENDIX A

GEOMETRIC SIMILARITY OF GENERATED SHORTEST PATHS

An important criterion for comparing the different shortest-path algorithms is the similarity of the geometry of the generated paths. Locational proximities between the generated shortest paths provide better insights into the shortcuts followed or modal changes due to different traffic situations that constrain the recommendations made by each algorithm. Geometric similarities between lines have been investigated within the GIScience literature for a variety of applications. Goodchild and Hunter (1997) have developed a method to assess the accuracy of linear features such as coastlines. They used a buffer zone around the lines, whereas the accuracy of the digitization is measured in terms of how much of it lies within the buffer zone.

Dean et al. (2015) built on this method to compare routes' similarities in an interesting way. They viewed the resulting routes generated by different shortest-path algorithms as lines connecting origins and destinations. Then they measured the similarity of these routes generated by different algorithms by constructing a buffer zone around the route of the base algorithm (vector-based route). Furthermore, they evaluated the total percentage of routes that resulted from their proposed algorithm within a buffer-defined adjacency metric of the vector-based route. Such a method is very efficient and intuitive in measuring the geometric similarity between different routes. Other similarity metrics could include the percentage of street or rail segments that are common in different routes.

Many other metrics could be found in the geometric similarity analysis literature, which is covered in a wide variety of fields, including genetic studies (Seyler, et al. 2015). However, to further improve the findings of this research, we suggest the method employed by Dean et al. (2015) because it gives a broader sense of the contextual spatial similarity between different routes. I would construct several buffer zones around the route generated by other algorithms. Then I would test the percentage of the route generated by the proposed algorithm that falls within these buffer zones. Another comparison could be conducted to check if the routes actually coincide (without buffering around reference routes). The percentage of the path length that is the same in the routes generated by the different algorithms will be reported in order to compare the different algorithms. Results can be presented in a chart similar to the one in Figure A-1 that was originally taken from the paper authored by Dean et al. (2015).

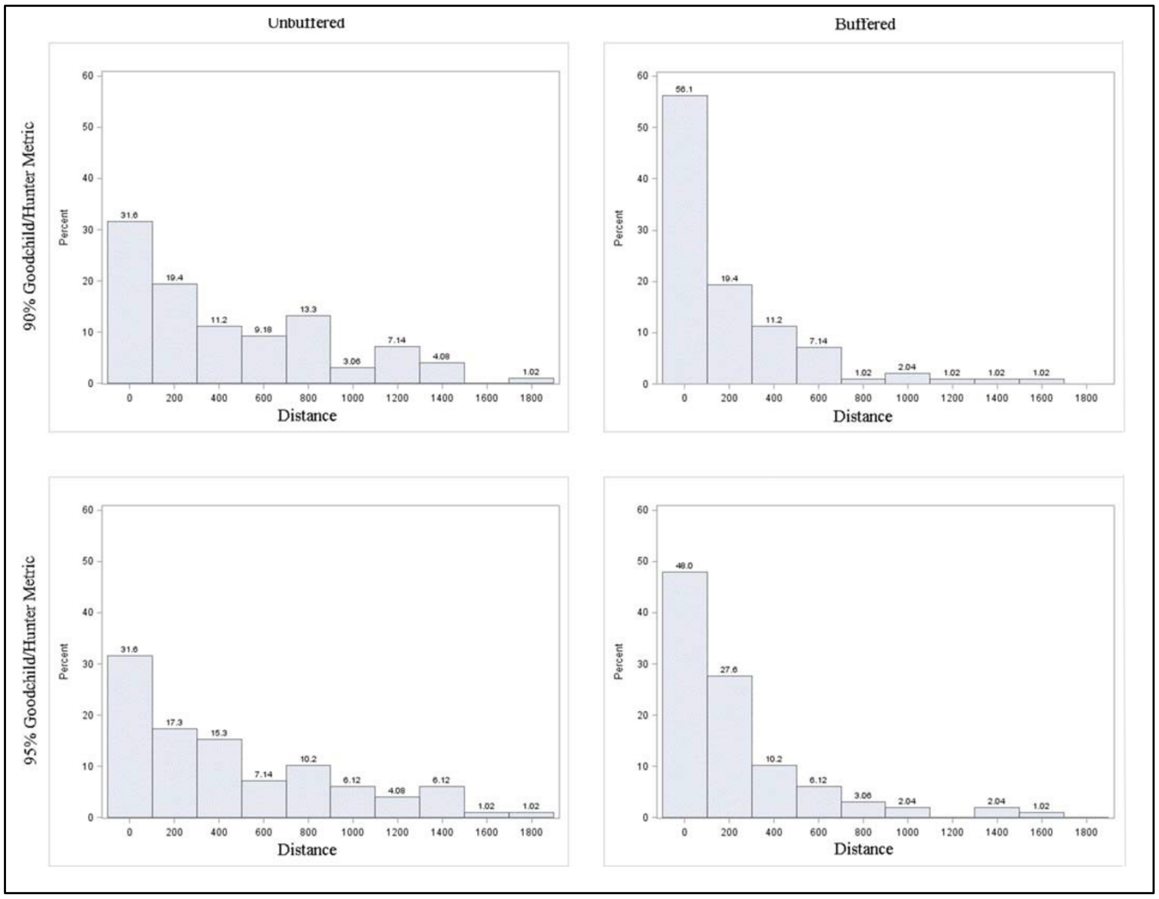


Figure A-1: Dean et al (2015) method of comparing routes (Dean, Thakar, & Sirdeshmukh, Optimal Routefinding Across Landscapes Featuring High-cost Linear Obstacles, 2015)

APPENDIX B

MATSIM NETWORK FILE STRUCTURE, AN EXAMPLE

```
</network>
<!-- ====================================================================== -->
<nodes>
        <!--Riyadh Streets Nodes Start Here-->
                <node id="-100028" x="675796.0" y="2743173.0" />
                <node id="-100029" x="675818.0" y="2743176.0" />
                <node id="-100063" x="676159.0" y="2743254.0" />
                <node id="-100066" x="676172.0" y="2743266.0" />
                <node id="-99905" x="675711.0" y="2743106.0" />
                <node id="-99947" x="675324.0" y="2742743.0" />
        <!--Riyadh Streets Nodes End Here-->
        <!--Riyadh Metro Lines and Walkway Nodes Start Here-->
                <node id="-30470" x="666385.0" y="2737486.0" />
                <node id="-30499" x="663444.0" y="2735491.0" />
                <node id="-30502" x="663441.0" y="2735494.0" />
                <node id="-30533" x="666383.0" y="2737489.0" />
                <node id="-33661" x="674626.0" y="2739314.0" />
                <node id="-33720" x="666053.0" y="2756094.0" />
        <!--Riyadh Metro Lines and Walkway Nodes End Here-->
</nodes>

<!-- ====================================================================== -->
<links capperiod="01:00:00" effectivecellsize="7.5" effectivelanewidth="3.75">
        <!--Riyadh Streets Links Start Here-->
                <link id="1" from="-90110" to="-90094" length="217.10967101056337" freespeed="22.22222222222222"
                capacity="1500.0" permlanes="1.0" oneway="1" modes="car" origid="-90113" type="primary" />
                <link id="10" from="-49153" to="-49119" length="376.60002614998467" freespeed="22.22222222222222"
                capacity="1500.0" permlanes="1.0" oneway="1" modes="car" origid="-49155" type="primary" />
                <link id="9999" from="-67689" to="-68820" length="197.6900839570807" freespeed="22.22222222222222"
                capacity="1500.0" permlanes="1.0" oneway="1" modes="car" origid="-68821" type="primary" />
        <!--Riyadh Streets Links End Here-->
        <!--Walkways that connect streets to stations - Riyadh Streets - Walkways - Metro Station Links Start Here-->
                <link id="6666661" from="-33280" to="-33321" length="3.75839991408" freespeed="0.5" capacity="600.0"
                permlanes="1.0" oneway="1" modes="transit_walk" type="tertiary" />
                <link id="6666662" from="-33321" to="-33280" length="3.75839991408" freespeed="0.5" capacity="600.0"
                permlanes="1.0" oneway="1" modes="transit_walk" type="tertiary" />
                <link id="6666815" from="-33478" to="-33481" length="19.3518055778" freespeed="0.5" capacity="600.0"
                permlanes="1.0" oneway="1" modes="transit_walk" type="tertiary" />
        <!--Riyadh Streets - Walkways - Metro Station Links End Here-->
        <!--Riyadh Metro Lines Links Start Here-->
                <link id="11111107" from="-31309" to="-31155" length="1621.8923412041506" freespeed="22.22222222222222"
                permlanes="1.0" oneway="1" modes="train" origid="-31329"  />
                <link id="11111108" from="-31155" to="-31309" length="1621.8923412041506" freespeed="22.22222222222222"
                permlanes="1.0" oneway="1" modes="train" origid="-31329" />
                <link id="775544332214" from="-5555546606" to="-5555579004" length="6.082762530298219"
                freespeed="22.22222222222222" permlanes="1.0" oneway="1" modes="train" />
                <!--Riyadh Metro Links and Walkways Links End Here-->
</links>
<!-- ====================================================================== -->
</network>
```

APPENDIX C

GLOBAL MODULE'S COORDINATES TRANSFORMATION (1)

```
/* *********************************************************************** *
 * project: org.matsim.*
 * TransformationFactory.java
 *                                                                         *
 * *********************************************************************** *
 *                                                                         *
 * copyright       : (C) 2007 by the members listed in the COPYING,        *
 *                   LICENSE and WARRANTY file.                            *
 * email           : info at matsim dot org                                *
 *                                                                         *
 * *********************************************************************** *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *   See also COPYING, LICENSE and WARRANTY file                           *
 *                                                                         *
 * *********************************************************************** */

package org.matsim.core.utils.geometry.transformations;

import org.matsim.core.utils.geometry.CoordinateTransformation;

/**
 * A factory to instantiate a specific coordinate transformation.
 *
 * @author mrieser
 *
 */
public abstract class TransformationFactory {

	public final static String WGS84 = "WGS84";
	public final static String ATLANTIS = "Atlantis";
	public final static String CH1903_LV03 = "CH1903_LV03"; // switzerland
	public final static String CH1903_LV03_Plus = "CH1903_LV03_Plus"; // switzerland new
	public final static String GK4 = "GK4"; // berlin/germany, own implementation
	public final static String WGS84_UTM47S = "WGS84_UTM47S"; // indonesia
	public final static String WGS84_UTM48N = "WGS84_UTM48N"; // Singapore
public final static String WGS84_UTM35S = "WGS84_UTM35S"; // South Africa (Gauteng)
        public final static String WGS84_UTM36S = "WGS84_UTM36S"; // South Africa (eThekwini, Kwazulu-Natal)
        public final static String WGS84_Albers = "WGS84_Albers"; // South Africa (Africa Albers Equal Conic)
        public final static String WGS84_SA_Albers = "WGS84_SA_Albers"; // South Africa (Adapted version of
        Africa Albers Equal)
        public final static String WGS84_UTM33N = "WGS84_UTM33N"; // Berlin
        public final static String DHDN_GK4 = "DHDN_GK4"; // berlin/germany, for GeoTools
        public final static String WGS84_UTM29N = "WGS84_UTM29N"; // coimbra/portugal
    public final static String WGS84_UTM31N = "WGS84_UTM31N"; // Barcelona/Spain
        public final static String CH1903_LV03_GT = "CH1903_LV03_GT"; //use geotools also for swiss coordinate
        system
        public final static String CH1903_LV03_Plus_GT = "CH1903_LV03_Plus_GT"; //use geotools also for swiss
        coordinate system
        public final static String WGS84_SVY21 = "WGS84_SVY21"; //Singapore2
        public final static String NAD83_UTM17N = "NAD83_UTM17N"; //Toronto, Canada
        public static final String WGS84_TM = "WGS84_TM"; //Singapore3
        public static final String PCS_ITRF2000_TM_UOS = "PCS_ITRF2000_TM_UOS"; // South Korea - but used by
        University of Seoul - probably a wrong one...
    public static final String WGS84_UTM = "WGS84_UTM"; // Saudi Arabia - Middle East - For Riyadh Scenario
        /**
         * Returns a coordinate transformation to transform coordinates from one
         * coordinate system to another one.
         *
         * @param fromSystem The source coordinate system.
         * @param toSystem The destination coordinate system.
         * @return Coordinate Transformation
         */
        public static CoordinateTransformation getCoordinateTransformation(final String fromSystem, final
        String toSystem) {
                if (fromSystem.equals(toSystem)) return new IdentityTransformation();
                if (WGS84.equals(fromSystem)) {
                        if (CH1903_LV03.equals(toSystem)) return new WGS84toCH1903LV03();
                        if (CH1903_LV03_Plus.equals(toSystem)) return new WGS84toCH1903LV03Plus();
                        if (ATLANTIS.equals(toSystem)) return new WGS84toAtlantis();
                      if (WGS84_UTM.equals(toSystem)) return new WGS84toUTM();
                }
                if (WGS84.equals(toSystem)) {
                        if (CH1903_LV03.equals(fromSystem)) return new CH1903LV03toWGS84();
                        if (CH1903_LV03_Plus.equals(fromSystem)) return new CH1903LV03PlustoWGS84();
                        if (GK4.equals(fromSystem)) return new GK4toWGS84();
                        if (ATLANTIS.equals(fromSystem)) return new AtlantisToWGS84();
                }
        return new GeotoolsTransformation(fromSystem, toSystem);
        }
}
```

APPENDIX C

GLOBAL MODULE’S COORDINATES TRANSFORMATION (2)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package org.matsim.core.utils.geometry.transformations;

import org.matsim.api.core.v01.Coord;
import org.matsim.core.utils.geometry.CoordinateTransformation;

/**
 * This is only for one Datum reference i.e., WGS84 with mathematical model that
 * fits the shape of the earth with this datum as reference Transforms
 * coordinates from Datum WGS84 to UTM synthetic coordinate system. The
 * transformed coordinates will lie somewhere in the Middle East, so it's not
 * disturbed by photographic texture. Coordinates in the synthetic coordinate
 * system should be in the range (-100000,-100000)-(100000,100000) to have a
 * useful transformation.
 *
 * @author Abdullah Binthunaiyan and Hassan Ahmad

 */
public class WGS84toUTM implements CoordinateTransformation {
//Class Level Variable Declarations
//WGS84 Mathematical Model Constants
    double DatumEqRad = 6378137.0; //Equatorial Radius in Metres
    double DatumFlat = 298.2572236; //Polar Flatenning
    double k0 = 0.9996; //Scale on Central Meridian
    double a = DatumEqRad; //Equatorial Radius in Metres
    double f = 1 / DatumFlat; //Polar Flattening
    double b = a * (1 - f); // polar axis
    double e = Math.sqrt(1 - b * b / a * a);//eccentricity
    double drad = Math.PI / 180;//Convert degrees to radians)
    double latd = 0;//latitude in degrees
    double phi = 0;//latitude (north +, south -), but uses phi in reference
    double e0 = e / Math.sqrt(1 - e * e);//e prime in reference
    double N = a / Math.sqrt(1 - Math.pow(e * Math.sin(phi), 2));
    double T = Math.pow(Math.tan(phi), 2);
    double C = Math.pow(e * Math.cos(phi), 2);
    double lng = 0;//Longitude (e = +, w = -) - can't use long - reserved word
    double lng0 = 0;//longitude of central meridian
    double lngd = 0;//longitude in degrees
    double M = 0;//M requires calculation
    double x = 0;//x coordinate
    double y = 0;//y coordinate
    double k = 1;//local scale
    double utmz = 30;//default starting utm zone
    double zcm = 0;//zone central meridian
    boolean OOZok = false;
    @Override
    public Coord transform(Coord coord) {
        //Convert Latitude and Longitude to UTM
        k0 = 0.9996;//scale on central meridian
        b = a * (1 - f);//polar axis.
        e = Math.sqrt(1 - (b / a) * (b / a));//eccentricity
        //Input Geographic Coordinates
        //Decimal Degree Option
        double latd0 = coord.getY();
        double lngd0 = coord.getX();
        lngd = lngd0;
        latd = latd0;
        phi = latd * drad;//Convert latitude to radians
        lng = lngd * drad;//Convert longitude to radians
        utmz = 1 + Math.floor((lngd+180)/6);//calculate utm zone
        double latz = 0;//Latitude zone: A-B S of -80, C-W -80 to +72, X 72-84, Y,Z N of 84
        if (latd > -80 && latd < 72){latz = Math.floor((latd + 80)/8)+2;}
        if (latd > 72 && latd < 84){latz = 21;}
        if (latd > 84){latz = 23;}
        zcm = 3 + 6*(utmz-1) - 180;//Central meridian of zone
       //Calculate Intermediate Terms
        e0 = e / Math.sqrt(1 - e * e);//Called e prime in reference
        double esq = (1 - (b / a) * (b / a));//e squared for use in expansions
        double e0sq = e * e / (1 - e * e);// e0 squared - always even powers
        N = a / Math.sqrt(1 - Math.pow(e * Math.sin(phi), 2));
        T = Math.pow(Math.tan(phi), 2);
        C = e0sq * Math.pow(Math.cos(phi), 2);
```

```
        double A = (lngd - zcm) * drad * Math.cos(phi);
       //Calculate M
        M = phi * (1 - esq * (1 / 4 + esq * (3 / 64 + 5 * esq / 256)));
        M = M - Math.sin(2 * phi) * (esq * (3 / 8 + esq * (3 / 32 + 45 * esq / 1024)));
        M = M + Math.sin(4 * phi) * (esq * esq * (15 / 256 + esq * 45 / 1024));
        M = M - Math.sin(6 * phi) * (esq * esq * esq * (35 / 3072));
        M = M * a;//Arc length along standard meridian
        double M0 = 0;//M0 is M for some origin latitude other than zero. Not needed for standard
UTM
        //Calculate UTM Values
        x = k0 * N * A * (1 + A * A * ((1 - T + C) / 6 + A * A * (5 - 18 * T + T * T + 72 * C -
58 * e0sq) / 120));//Easting relative to CM
        x = x + 500000;//Easting standard
        y = k0 * (M - M0 + N * Math.tan(phi) * (A * A * (1 / 2 + A * A * ((5 - T + 9 * C + 4 * C
* C) / 24 + A * A * (61 - 58 * T + T * T + 600 * C - 330 * e0sq) / 720))));//Northing from
equator
        if (y < 0) {
            y = 10000000 + y;
        }
        //Output
         return new Coord(Math.round(10 * (x)) / 10, Math.round(10 * y) / 10);
    }


}
```

APPENDIX D

CONFIGURATION FILE

```
?xml version="1.0" ?>
<!DOCTYPE config SYSTEM "http://www.matsim.org/files/dtd/config_v1.dtd">
<config>
        <module name="global">
                <param name="randomSeed" value="4711" />
                <param name="coordinateSystem" value="WGS84_UTM" />
        </module>
        <module name="network">
                <param name="inputNetworkFile" 
value="riyadhMultimodeWithScheduleScenario/multimodalnetwork.xml" />
        </module>
        <module name="plans">
                <param name="inputPlansFile" 
value="riyadhMultimodeWithScheduleScenario/population.xml" />
        </module>
        <module name="controler">
                <param name="outputDirectory" value="./output/riyadhMultimodeWithScheduleScenario" 
/>
                <param name="firstIteration" value="0" />
                <param name="lastIteration" value="50" />
                <param name="eventsFileFormat" value="xml" />
                <param name="mobsim" value="qsim" />
                <param name="createGraphs" value="true"/>
                <param name="routingAlgorithmType" value="Dijkstra"/>
        </module>
        <module name="qsim">
                <!-- "start/endTime" of MobSim (00:00:00 == take earliest activity time/ run as 
long as active vehicles exist) -->
                <param name="startTime" value="00:00:00" />
                <param name="endTime" value="30:00:00" />
                <param name = "snapshotperiod"       value = "00:00:00"/> <!-- 00:00:00 means NO 
snapshot writing -->
        </module>
<module name="planCalcScore">
                <param name="learningRate" value="1.0" />
                <param name="BrainExpBeta" value="2.0" />
                <param name="lateArrival" value="-18" />
                <param name="earlyDeparture" value="-0" />
                <param name="performing" value="+6" />
                <param name="traveling" value="-6" />
                <param name="waiting" value="-0" />
                <param name="activityType_0"            value="work" /> <!-- work -->
                <param name="activityPriority_0"        value="1" />
                <param name="activityTypicalDuration_0" value="12:00:00" />
                <param name="activityMinimalDuration_0" value="08:00:00" />
                <param name="activityType_1"            value="home" /> <!-- home -->
                <param name="activityPriority_1"        value="1" />
                <param name="activityTypicalDuration_1" value="08:00:00" />
                <param name="activityMinimalDuration_1" value="06:00:00" />
                <param name="activityOpeningTime_1"     value="07:00:00" />
                <param name="activityLatestStartTime_1" value="09:00:00" />
                <param name="activityEarliestEndTime_1" value="" />
                <param name="activityClosingTime_1"     value="18:00:00" />
        </module>
        <module name="strategy">
                <param name="maxAgentPlanMemorySize" value="5" /> <!-- 0 means unlimited -->
                <param name="ModuleProbability_1" value="0.7" />
                <param name="Module_1" value="BestScore" />
                <param name="ModuleProbability_2" value="0.1" />
                <param name="Module_2" value="ReRoute" />
                <param name="ModuleProbability_3" value="0.1" />
                <param name="Module_3" value="TimeAllocationMutator" />
                <param name="ModuleProbability_4" value="0.1" />
                <param name="Module_4" value="ChangeTripMode" />
        </module>
        <module name="transit">
                <param name="useTransit" value="true" />
                <param name="transitScheduleFile"
                value="riyadhMultimodeWithScheduleScenario/transitschedule.xml" />
                <param name="vehiclesFile"
                value="riyadhMultimodeWithScheduleScenario/transitVehicles.xml" />
                <param name="transitModes" value="train" />
        </module>

</config>
```

REFERENCES

American Public Transportation Association. 2016. *Shared Mobility and the Transformation of Public Transit.* Washington, DC: American Public Transportation Association.

Andreas, Horni, Kay W. Axhausen, and Kai Nagel. 2016. *The Multi-Agent Transport Simulation.* London: Ubiquity Press.

Apple. 2017. *The future is here: iPhone X.* 09 12. Accessed 10 22, 2017. https://www.apple.com/newsroom/2017/09/the-future-is-here-iphone-x/.

Arriyadh Development Authority. 2014. *(ADA Programs and Projects).* 12 1. Accessed 12 1, 2014. http://www.ada.gov.sa/ADA_A/DocumentShow/?url=/res/ADA/Ar/Projects/RiyadhMetro/index.html.

Axhausen, Kay, Andreas Horni, and Kai Nagel. 2016. *The Multi-Agent Transport Simulation.* London: Ubiquity press.

Badii, Anush. 2014. *A Practical Introduction to Traffic Flow Theory and on Ramp Flow Control.* San Diego: California Department of Transportation.

Bertsekas, Dimitri P. 1998. *Network optimization: continuous and discrete models.* . Belmont: Athena Scientific.

Bhaskar, Umang, Lisa Fleischer, Darrell Hoy, and Chien-Chung Huang. 2009. "Equilibria of Atomic Flow Games are not Unique." *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms* 748-757.

Bondy, J. A., and U.S.R. Murty. 1976. *Graph Theory With Applications.* New York: The Macmillan Press Ltd.

Cascetta, Ennio. 2009. "Transportation Supply Models." In *Transportation systems analysis: models and applications,* by Ennio Cascetta, 29-88. Springer Science & Business Media.

Chen, S. K., S. Liu, X. Xiao, J. Hong, and B. H Mao. 2012. "M/G/C/C-Based Model of Passenger Evacuation Capacity of Stairs and Corridors in Metro Stations." *Journal of the China Railway Society* 7-12.

Coello, Carlos A., Gary B. Lamont, and David A. Van Veldhuizen. 2007. *Evolutionary algorithms for solving multi-objective problems.* New York: Springer.

Correa, Jose., and Nicol Stier-Moses. 2010. "Wardrop Equilibria." *Wiley Encyclopedia of Operations Research and Management Science.*

Dean, Denis J. 2015. "Spatial Optimization Class at UTD." 10 1.

Dean, Denis J., Vaishnavi Thakar, and Neeraj Sirdeshmukh. 2015. "Optimal Routefinding Across Landscapes Featuring High-cost Linear Obstacles." *Transactions in GIS.*

Dutch, Steven. 2017. *Convert Geographical Units.* 08 01. Accessed 08 01, 2017. http://www.rcn.montana.edu/resources/converter.aspx.

ESRI. 2017. "Network Analyst Tutorial." 01 01. http://help.arcgis.com/en/arcgisdesktop/10.0/pdf/network-analyst-tutorial.pdf.

—. 2010. *Network elements.* 03 02. http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Network_elements.

Fisk, C. S. 1984. "Game theory and transportation systems modelling." *Methodological* 301-313.

Flotterod, Gunnar. 2016. "MATSim as aMonte-Carlo Engine." In *The Multi-Agent Transport Simulation MATSim*, by A Horni, K Nagel and K W. Axhausen, 327–336. London: Ubiquity Press.

Gao, Wenli. 2009. "Comparisons between MATSim and EMME/2 on the Greater Toronto and Hamilton Area Network." Toronto.

Goodchild, Michael F., and Gary J. Hunter. 1997. "A simple positional accuracy measure for linear Features." *International Journal of Geographical Information Science* 299-306.

Goodchild, Michael. 2000. "GIS and Transportation: Status and Challenges." *GeoInformatica* 127–139.

Google. 2017. *Google Pixel 2.* 10 22. Accessed 10 22, 2017. https://store.google.com/product/pixel_2_learn.

Google Inc. 2015. *Google Maps Content Providers.* 12 20. https://maps.google.com/help/maps/mapcontent/transit/faq.html#genq1.

Google Official Blog. 2009. *Google Official Blog.* August 25. https://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html.

Google, Inc. 2016. *Success Stories.* 03 01. http://maps.google.com/help/maps/mapcontent/transit/success-story.html.

Hollander, Y., and J. N. Prashker. 2006. "The applicability of non-cooperative game theory in transport analysis." *Transportation* 481-496.

Iacobucci, Joe, Kirk Hovenkotter, and Jacob Anbinder. 2017. "Transit Systems and the Impacts of Shared Mobility." In *Disrupting Mobility, Impacts of Sharing Economy and Innovative*, by Gereon Meyer and Susan Shaheen, 75. Berlin: Springer International Publishing .

intel. 2017. *50 Years of Moore's Law.* 10 22. Accessed 10 22, 2017. https://www.intel.com/content/www/us/en/silicon-innovations/moores-law-technology.html.

Kaufman, David E., Nonis Jason, and Robert L. Smith. 1998. "A mixed integer linear programming model for dynamic route guidance." *Transportation Research Part B: Methodological 32.6* 431-440.

Kleinrock, Leonard. 1975. *Queuing Systems Volum I: Theory.* Los Angelese: John Wiley & Sons.

Liu, Lu. 2010. *Data Model and Algorithms for Multimodal Route Planning with Transportation Networks.* Munich: Technical University of Munich.

Macfarlane, Greg. 2017. *A script to convert a lines shapefile into a MATSim network.* 07 01. https://gist.github.com/gregmacfarlane/eb5a9299b1532f4b631c.

Maerivoet, Sven, and Bart De Moor. 2005. "Traffic flow theory." *physics/0507126.*

Magzhan, Kairanbay, and Hajar Mat Jani. 2013. "A Review And Evaluations Of Shortest Path Algorithms." *International Journal of Scientific & Technology Research* 99-104.

MATSim. 2017. *Documentation.* 01 01. http://www.matsim.org/docs.

McNally, Michael G. 2007. "THE FOUR STEP MODEL." In *Handbook of Transport Modeling*, by Hensher and Button. eds.

Microsoft. 2017. *Microsoft Lumia 950 XL.* 10 22. Accessed 10 22, 2017. https://www.microsoft.com/en-us/mobile/phone/lumia950-xl-dual-sim/.

Nagel, Kai. 2017. *Testing for Nash equilibrium?* 09 02. https://matsim.atlassian.net/wiki/questions/75268118/testing-for-nash-equilibrium.

Nagel, Kai, Kay W. Axhausen, Benjamin Kickhofer, and Andreas Horni. 2016. "Research Avenues." In *The Multi-Agent Transport Simulation MATSim*, by Kai Nagel, Kay W. Axhausen and Andreas Horni, 533–542. London: Ubiquity Press.

Osborne, Martin J., and Ariel Rubinstein. 1994. *A course in Game Theory.* MIT Press.

Popovici, Elena, Anthony R Bucci, Paul Wiegand, and Edwin D. de Jong. 2012. "Coevolutionary Principles." In *Handbook of Natural Computing*, by Grzegorz Rozenberg, Thomas Bck and Joost N. Kok, 987-1033. Berlin : Springer Berlin Heidelberg.

Qualcomm. 2017. 10 22. Accessed 10 22, 2017. https://www.qualcomm.com/products/snapdragon.

Rapino, Melanie A., and Alison K. Fields. 2013. "Mega Commuting in the U.S: Time and Distance in Defining the Long Commute using the American Comminty Survey." *Association for Public Policy Analysis and Management Fall 2013 Conference* . Washington, DC.

Rodrigue, Jean-Paul, Claude Comtois, and Brian Slack. 2006. *The Geography of Transport.* New York: Routledge.

Sam Drew Takes On. 2015. *Waze Social GPS Maps & Traffic – What's so Awesome about This Navigation App?* 05 31. http://www.samdrewtakeson.com/2015/05/waze-social-gps-maps-traffic-whats-awesome-navigation-app/.

Schulz, Andreas S., and Nicolás Stier-Moses. 2003. "On the performance of user equilibria in traffic networks." *Fourteenth annual ACM-SIAM symposium on Discrete algorithms.* Baltimore. 86-87.

Seyler, Sean L., Avishek Kumar, M. F. Thorpe, and Oliver Beckstein. 2015. "Path Similarity Analysis: A Method for Quantifying Macromolecular Pathways." *Computational Biology.*

The Linley Group. 2015. *A Guide to Mobile Processors.* 08 01. Accessed 22 2017, 10. http://www.fiercewireless.com/wireless/report-qualcomm-led-smartphone-application-processor-market-1h-2016.

Thunig, Theresa, and Kai Nagel. 2017. "The structure of user equilibria: Dynamic coevolutionary simulations vs. cyclically expanded networks." *Procedia Computer Science* 648–655.

Thuniga, Theresa, and Kai Nagel. 2017. "The structure of user equilibria: Dynamic coevolutionary simulations vs. cyclically expanded networks." *Procedia Computer Science* 648–655.

Toyota. 2016. *Camry Specifications.* https://www.toyota.co.nz/our-range/camry/camry/specifications/gl/.

Trainline. 2016. *Mobile App from Trainline.* 04 01. http://www.bucksfreepress.co.uk/beaconsfield/14482182.New_idea_brings_end_to_long_queues_at_railway_stations/.

Uber. 2014. *Our Committment to Safety.* 12 17. https://newsroom.uber.com/our-commitment-to-safety/.

Wardrop, John. 1952. " Some theoretical aspects of road traffic research." *ICE Proceedings: Part II, Engineering Divisions* 325-362.

Xu, Xin-yue, Liu Jun, Hai-ying Li, and Jian-Qiang Hu. 2014. "Analysis of subway station capacity with the use of queueing theory." *Transportation research part C: emerging technologies 38* 28-43.

Yazicioglu, Ahmet Yasin, Xiaoli Ma, and Yucel Altunbasak. 2011. "Analyzing the Dynamics of Evolutionary Prisoner's Dilemma on Structured Networks." *International Conference on Game Theory for Networks.* Berlin Heidelberg: Springer . 190-204.

Zakharo, Victor, Alexander Krylatov, and Dmitry Ivanov. 2013. "Equilibrium Traffic Flow Assignment in Case of Two Navigation Providers." In *Collaborative Systems for Reindustrialization*, by Luis M. Camarinha-Matos and Raimar J. Scherer, 156-163. Berlin Heidelberg: Springer.

Zhan, F. Benjamin, and Charles E. Noon. 1998. "Shortest Path Algorithms: An Evaluation Using Real Road Network." *Transportation Science* 65-73.

BIOGRAPHICAL SKETCH

Abdullah Binthunaiyan was born and grew up in Durma- Saudi Arabia. In 2003, he obtained his bachelor’s degree in electrical engineering from King Fahd University of Petroleum and Minerals-Dhahran. He joined the National Information Center in the same year and worked in the research and development department. Then he got his master’s degree in Geographical Information Systems from the University of Redlands in 2010. As an employee in the National Information Center, he supervised and managed several projects and data products. In 2018, he graduated from The University of Texas at Dallas with a PhD. in Geospatial Information Sciences.

CURRICULUM VITA

ABDULLAH N. BINTHUNAIYAN

Education

2003	B.Sc. in Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran.
2008	GIS Certificate, University of California-Riverside
2010	MS GIS, University of Redlands, California, USA
2013- 2018	PhD-GISC, University of Texas at Dallas

Work Experience

● **National Information Center- Saudi Arabia**

R & D specialist, and GIS specialist. Researched on: GIS, APIS, RFID, Smart Cities, Data Mining, Spatial Optimization, and Agent-Based Modeling. Surveyed and researched: Social, Public, Labor and Political spatial-based data. Evaluated RFP responses on the following projects: GIS, AVL, RFID, Oracle Licensing, Data Mining. Worked as MACA-GIS project Manager, GIS application manager for Traffic and Security Patrol Departments, conducted a pilot GIS project for the Higher Commission of Industrial Security Saudi Arabia. Served in the Information Security Auditing Committee-NIC.

Memberships

● American Geographers Association, GIS Corp, Academy of Science and Engineering, the International Geographical Honor Society, and the Association for Information Systems.

Research

● Smart Cities, Opportunity, Challenges, and Economic Implications, In Progress

● Spatial Optimization and Multimodal Shortest Path Algorithm

● Multilevel Modeling of HVC Knowledge in Egypt

● Mapping the Feeling Thermometer of American Voters in the 2016 Election.

Personal Background

Grew up in Riyadh, Kingdom of Saudi Arabia. Interests include online gaming

and sports (soccer and squash rackets). Love horse racings.