DESIGN OF AN INTELLIGENT TRAFFIC MANAGEMENT SYSTEM

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Civil Engineering

By

Amin Azimian

Dayton, Ohio

December, 2011

DESIGN OF AN INTELLIGENT TRAFFIC MANAGEMENT SYSTEM

Name: Azimian, Amin

APPROVED BY:

Deogratias Eustace, Ph.D., P.E., PTOE Advisory Committee Chairman Associate Professor Department of Civil and Environmental And Engineering Mechanics	Arthur Busch, Ph.D. Committee Member Assistant Professor Department of Mathematics
Maher Qumsiyeh, Ph.D. Committee Member Assistant Professor Department of Mathematics	Paul Goodhue, P.E., PTOE Committee Member Transportation Manager CESO, Inc.
John G. Weber, Ph.D. Associate Dean School of Engineering	Tony E. Saliba, Ph.D. Dean, School of Engineering & Wilke Distinguished Professor

ABSTRACT

DESIGN OF AN INTELLIGENT TRAFFIC MANAGEMENT SYSTEM

Name: Azimian, Amin
University of Dayton

Advisor: Dr. Deogratias Eustace

Due to present-day significant increases in population and consequently in traffic congestion in most metropolitan cities in the world, designing of an intelligent traffic management system (ITMS) in order to detect the path with the shortest travel time is critical for emergency, health, and courier services. The aim of this thesis study was to develop a theoretical traffic detection system and capable of estimating the travel time associated with each street segment based on the traffic data updated every 20 seconds, which successively finds the path with the shortest travel time in the network by using a dynamic programming technique. Furthermore, in this study we model the travel time associated with each street segment based on the historical and real time data considering that the traffic speed on each road segment is piecewise constant. It would be useful to implement such algorithms in GIS systems such as Google map in such a way that the service delivery drivers can avoid congested routes by receiving real time traffic information.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Deogratias Eustace for his support, advice and endless patience in improving my writing. Without his brilliant guidance, this thesis could not have been accomplished. I would also like to express my appreciation to my committee members, Dr. Arthur Busch, Dr. Maher Qumsiyeh and Mr. Paul Goodhue for their constructive comments and suggestions. Finally, I would like to thank all those who supported me in one way or another, especially my parents for supporting my goals, and my brothers and sisters for believing that I could make it through.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Over several decades, traffic congestion has become a serious problem in the U.S. major cities. Congestion is particularly associated with motorization and the diffusion of the automobile, which has increased the demand for transportation infrastructure. However, the supply of the transportation infrastructure has often not been able to keep up with the growth of mobility.

Traffic congestion problems consist of incremental delay, vehicle operating costs such as fuel consumption, pollution emissions and stress that result from interference among vehicles in the traffic stream, particularly as traffic volumes approach a road’s capacity. Across the U.S., more people are spending more time sitting in traffic jams than ever before. According to the U.S. Census Bureau (2000), nationwide, the average commute increased 14 percent in the last ten years, from 22.4 minutes in 1990 to 25.5 minutes in 2000.

Dear et al. (2001) reported that California already has five of the nation's 20 most congested metro areas. In California, traffic congestions statewide cost $21 billion due to lost time and wasted fuel every year. In this regard the state's official forecast shows the number of miles driven on Los Angeles and Orange County roads will increase 40 percent by 2020. Furthermore, Wasserman reported that in Sacramento, even with $15 billion in planned road improvements, congestion will increase by 400 percent in the next 20 years.

According to American Society of Civil Engineers, "*2001 Report Card for America's Infrastructure*", in Texas, 26 percent of freeways is congested and traffic volumes on the state's highways have increased by one-third in ten years. Also Rubin and Cox (2001) reported that Texas traffic is growing so quickly that even if public transit use were to double, the gain would be canceled out by population growth in as little as three months.

Traffic congestion occurs when the demand is greater than the available road capacity. There are many reasons that cause congestion; most of them reduce the capacity of the road at a given point or over a certain length, for example people parking on the roads or increase in the number of vehicles. As shown in Table 1.1, below the Federal Highway Administration, (Margiotta et al, 2009) mentioned that about half of U.S. traffic congestion is recurring,

Table 1.1 The causes of traffic jams

Bottlenecks	40% of total congestion
Traffic Incidents	25% of total congestion
Work zones	10% of total congestion
Bad weather	15% of total congestion
Poor signal timing	5% of total congestion
Special events/Other	5% of total congestion

and is attributed to sheer volume of traffic; most of the rest depends on crashes, road construction works and severe weather events. Consequently, all of these factors affect our communities both mentally as well as economically. The Ohio Department of Transportation (2009) reported that traffic congestion prevents Honda's employees from arriving on time which may threaten Honda's low-inventory strategy in Ohio. There are always concerns that traffic congestion may delay emergency vehicles during critical moments when they need to arrive at the scenes as quickly as possible.

1.2 Data Collection Techniques

According to the Federal Highway Administration (Turner et al., 1998), the travel time data collection techniques can be categorized into the following groups:

1) **Test vehicle technique**: in this technique an observer records cumulative travel time at predefined checkpoints along a travel route, then this information can be converted to travel time, speed for each segment along the proposed route. In this technique it is not possible to store a large amount of data. In addition, it requires quality control considering that it is associated with human or electric errors.

2) **License plate matching technique:** in this method the plate numbers and arrival times will be recorded at various checkpoints, and finally the travel times will be computed by matching the license plate numbers or from the difference in arrival times. This method allows us to obtain travel times from a large sample of motorists, and it is possible to find out the variability of travel times among vehicles within the traffic stream. Furthermore, it is possible to transport data

collection equipment between observation sites. However the travel time data is only available to the area where video cameras are installed.

3) **ITS probe vehicle technique:** this technique can collect travel times data by employing passive instrumented vehicles in the traffic stream and remote sensing devices. By using this technique it is possible to collect data electronically for 24 hours, but such system requires skilled software designer and high implementation cost, furthermore this technique is not recommended for small scale data collection efforts.
4) **Emerging technique:** this technique estimates travel times by using a variety of methods, such as inductance loops, weigh-in-motion stations or aerial video. Some of the methods used in this technique are still in testing stages.

1.3 Objectives of Study

Developing a new system such as an intelligent traffic management system in order to reduce traffic problems is essential. The objectives of this study are three-fold:

1- Developing traffic detection and estimation of traffic mean speeds and travel times associated with each street segment based on the data provided by traffic detectors;

2- Identifying specific zones in which all possible (or reasonable) routes are located (classification method and defining a coordinate system); and

3- Designing of a route-finding algorithm.

1.4 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 presents a literature review on previous research studies concerning with shortest path finding problems. Chapter 3 presents a description of the timetable and speedtable updating system and the mathematical formulation of the dynamic programming and travel time modeling. Chapter 4 presents the results and a numerical example. Finally, Chapter 5 presents the conclusions and recommendations of the study.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Dynamic shortest path finding problems are a subset of dynamic transportation problems including dynamic traffic assignment, dynamic fleet management, etc. The term “dynamic” is also called “online” or “real-time,” which means the problem-related solutions are time-dependent. A shortest path problem is a problem of finding the shortest path from one source point to every other node in a directed graph in which each link has a particular weight. Every node and link stands for street intersection and road segment respectively. Yen (1975) split the shortest path algorithms into the following three groups.

1- Finding the shortest path from a particular point to every other node in a directed network.

2- Finding the shortest path between all pairs of the nodes in the directed network.

3- Finding the short path between two particular nodes. This group is the most popular in the transportation area.

2.2 Shortest Path Problem

In the last few years, some research efforts have focused on developing different algorithms for Dynamic Vehicle Routing Problems (DVRPs); many of these applications can be found in practice, for example dial-a-ride systems for transportation-on-demand, courier services, emergency services, pick-up and delivery of goods, and many others.

As a pioneering work, Bertsimas and Van Ryzin (1991) reported a model for stochastic and dynamic vehicle routing with a single vehicle, traveling at a constant velocity in its region of service, whose time of arrival, location and on-site service are stochastic. The objective was to find a policy to determine service demands over an infinite horizon that minimizes the expected system time, however a challenging problem in a different direction is to investigate dynamic routing in a network environment rather than under some Euclidean metric.

Gendreau et al. (2006) developed an algorithm for a pick-up and delivery problem in which vehicles travel at a constant velocity in regard to new request locations that uses heuristic and numerical calculations to minimize the expected system time. Their numerical results show the benefits of such techniques in real-time situation.

Taniguchi and Shimamoto (2004) cited a comprehensive planning strategy for dynamic routing of traffic in a city. They proposed a routing system, which uses a routing algorithm based on the ant colony optimization algorithm. This algorithm follows the behavior of living ants that lay pheromone trail on the ground in order to find shortest way from the nest toward the food location. In their proposed model pheromone is a time function related to the vehicles that travel across the networks. The time function can be affected by congestion. Each node has a probability table in which there are entries for each neighboring node that can be reached via one connecting link, therefore the probabilities influence the driver's selection of the next route. Each time the driver will be notified about next route (node) and this process is repeated until the driver reaches his destination. In Taniguchi and Shimamoto's model congestion can be detected using detectors and since its algorithm is based on the ant colony it takes a long time to find the shortest route.

Bellman (1957) developed an algorithm that calculates the shortest path to all nodes from a single source in the graph with non-negative edge weights which are constant functions. His approach seeks to solve each sub problem only once, thus reducing the number of computations. This is especially useful when the number of repeating sub problems is exponentially large.

Some of the research efforts (e.g., Bertsekas and Tsitsiklis, 1991; Frank, 1969; Loui, 1983; Ji, 2005) focused on stochastic shortest path problems on static road network. The stochastic shortest path problem is a generalization where either the network isn't completely known to the driver, and each link is associated with a probability of independently being in the network, while in deterministic process the weight associated with each link is known. Bertsekas and Tsitsiklis (1991) defined a stochastic shortest path problem in which each node has a transition matrix consisting of probability distributions over the set of successor nodes so as to reach the destination node with the minimum expected cost. Moreover, Frank (1969) applied shortest-path probability distributions in graphs in which the weights associated with each link are replaced with their expected values; subsequently Loui (1983) improved this model by using a "utility function" to determine the optimal path.

Ji (2005) studied the shortest path problem with stochastic arc length. She proposed the following three concepts of stochastic shortest path and formulated three models for the stochastic shortest path according to different decision criteria:
(1) Expected shortest path that finds a path with the minimum expected time to help them make a decision; (2) alpha-shortest path that finds a path, which satisfies some chance constraints with at least some, given confidence level α; (3) the most shortest path that finds a path which maximizes the chance functions of some events (i.e., the probabilities of satisfying the events). Furthermore, it was assumed that there is only one directed arc *(i,j)* from *i* to *j*.

Orda et al. (1993) formulated a stochastic model for the shortest path problem on a dynamic network whose link delays change probabilistically according to Markov chains and they assumed that the routing decisions at a node are based on the current state of links emanating from that node and on the statistics of other links. However their stochastic model considers the change of network to be predictable.

Gonzalez et al. (2007) presented an adaptive fastest path algorithm based on the (1) hierarchy of roads, (2) path segments traveled frequently, and (3) significant speed advantage. Major roads are more preferable than minor roads except if there are minor roads with significant speeds over the major ones. They defined different speed patterns for various conditions such as time of day, weather or vehicle type. They employed A* algorithm to find the shortest path. A* is a shortest path finding algorithm that uses a best-first search algorithm to find the least-cost path from a given initial node to one destination node.

Similarly, Kanoulas et al. (2006) developed another method based on the A* algorithm to find the fastest paths on a road network with speed patterns for any time interval. For example, it can find the shortest path for a person who wants to go to work sometime between 7:00 and 7:45 AM. In their proposed method each day belongs to exactly one category; workday or non-workday and for the days in the same category a road segment has the same speed at the same time of the day. However, they assumed that speed on each street segment is piecewise constant. In addition, Ding et al (2008) studied a time dependent shortest path problem in which a graph (or a road network) has an edge-delay function (travel time function) associated with each edge. They proposed a new Dijkstra-based algorithm to find the least total travel time.

Peeta et al. (2011) studied the problem of dynamic routing operations in the emergency response context of the routing of response vehicles and evacuees. The study focuses on identifying the paths used for routing response vehicles and the evacuees in disaster situations, as a result two modules are developed: (1) the K-shortest paths module that allows more flexible options for routing response vehicles under the dynamic network conditions due to a disaster, and (2) the multiple-stop routing module that enables the delivery of relief resources to several locations using a single response vehicle.

Orda and Rom (1990) formulated a model of shortest path problem for communication systems in which link weights changes over the time according to arbitrary functions. They developed algorithms for finding the shortest-path and minimum-cost under various weighting constraints and investigated the properties of the derived path.

Effati and Jafarzadeh (2007) presented a nonlinear neural network for solving the shortest path on a static network. In their research effort they defined a directed graph, whose edges have fixed costs; the cost coefficient can be either positive or negative. A positive cost coefficient represents a loss, whereas a negative one represents a gain.

Chitra and Subbaraj (2010) presented a non-dominated sorting genetic algorithm for shortest path routing problem. They used a multiobjectives evolutionary algorithm based on the Non-dominated Sorting Genetic Algorithm (NSGA), for solving the dynamic shortest path routing problem in computer networks. The problem is formulated as a multiobjective mathematical programming that attempts to minimize both delay and cost simultaneously. In their research effort the topology of the network was specified by an undirected graph and the cost associated with each link is predefined. Furthermore, Ahn and Ramakrishna (2002) used the same method, but the topology of the network was specified by a directed graph.

As mentioned by Wang (2003), shortest path problems are mostly studied by computer scientists, and almost none of them present a proper method of finding the weights associated with each link. However, in some GPS technologies speed limits posted on each street segment are used in estimating the travel times, but we cannot ignore the traffic changes or congestion in the street networks. Today intelligent transportation systems (ITS) applications can run over the wireless broadband network, which can allow us to measure the number of vehicles successively, transfer data, and consequently find the average travel times related to each link. Definitely such technologies can improve the efficiency of vehicle movement throughout the entire urbanized area.

2.3 Algorithm Efficiency

In computer science, efficiency is used to describe properties of an algorithm relating to how much of various types of resources it consumes. Yen (1975) explained the following factors that can be used to evaluate algorithm efficiency:

- The number of operations required to execute an algorithm (i.e. additions, subtractions, comparisons, etc.)
- The running time that each computer requires to execute the algorithm
- The amount of memory that each computer requires to store the raw data or results
- The amount of memory that each computer requires to store the computer program of the algorithm.

Although the efficient use of both runtime and space (memory) is important, runtime is usually more important than space. Wang (2003) mentioned that the runtime can be evaluated in two ways: (1) the asymptotic or worst-case runtime: the time that an algorithm requires to run if it were given the most insidious of all possible inputs, and (2) the average-case runtime: the average time that an algorithm requires to run if it were given all possible inputs. In computer science the Big-O notation represents the asymptotic time class of an algorithm, which provides an upper bound on the growth rate of the function. For example if a problem of size *n* requires time that is directly proportional to *n*, the upper bound on the number of operations for that problem is in class g(n) and O(g(n)) stands for an algorithm that has order g(n). Table 2.1 shows the intuitive interpretations for some growth-rate functions (Rinker, 2002).

Table 2.1 Interpretation of some growth-rate functions

O(1)	The time requirement for this growth rate function is constant and independent of the problem's size.
O(log n)	The time requirement increases slowly as the problem size increases. The binary search algorithm has this behavior.
O(n)	The time requirement increases directly with the size of the problem.
O(n.log n)	The time requirement increases more rapidly than a linear algorithm.
$O(n^2)$	The time requirement increases rapidly with the size of the problem. Algorithms that use two nested loops are often quadratic.
$O(n^3)$	Compared to the quadratic algorithm the time requirement for the cubic algorithm increases more rapidly with the size of the problem. Algorithms that use three nested loops are often cubic and are practical only for small problems.
$O(2^n)$	As the size of a problem increases, the time requirement for an exponential algorithm usually increases too rapidly.

For a supercomputer that performs 21 trillion operations per second, the approximate completion time for the following algorithms with different values of *n* are as given in Table 2.2.

Table 2.2 Completion time for algorithms with different values of *n*

n	**g(n)**					
	N	***n* log *n***	n^2	n^3	n^4	2^n
10	$4.8*10^{-13}$	$4.8*10^{-13}$	$4.8*10^{-12}$	$4.8*10^{-11}$	$4.8*10^{-10}$	$4.8*10^{-11}$
20	$9.5*10^{-13}$	$1.2*10^{-12}$	$1.9*10^{-11}$	$3.8*10^{-10}$	$7.6*10^{-9}$	$5.0*10^{-8}$
30	$1.4*10^{-12}$	$2.1*10^{-12}$	$4.3*10^{-11}$	$1.3*10^{-9}$	$3.90*10^{-8}$	$5.1*10^{-5}$
40	$1.9*10^{-12}$	$3.1*10^{-12}$	$7.6*10^{-11}$	$3.0*10^{-9}$	$1.2*10^{-7}$	$5.2*10^{-2}$
50	$2.4*10^{-12}$	$4.0*10^{-12}$	$1.2*10^{-10}$	$6.0*10^{-9}$	$3.0*10^{-7}$	$5.0*10^{1}$
10^2	$4.8*10^{-12}$	$9.5*10^{-12}$	$4.8*10^{-10}$	$4.8*10^{-8}$	$4.8*10^{-7}$	$6.0*10^{16}$
10^3	$4.8*10^{-11}$	$1.4*10^{-12}$	$4.8*10^{-8}$	$4.8*10^{-5}$	$4.80*10^{-2}$	$5*10^{287}$
10^4	$4.8*10^{-10}$	$1.9*10^{-12}$	$4.8*10^{-6}$	$4.8*10^{-2}$	$5*10^{2}$	
10^5	$4.8*10^{-9}$	$2.4*10^{-12}$	$4.8*10^{-4}$	48	$5*10^{6}$	
10^6	$4.8*10^{-8}$	$2.9*10^{-12}$	$4.8*10^{-2}$	$4.8*10^{4}$	$5*10^{10}$	

2.4 Summary of Literature Review

According to the literature review, it is known that the study of the shortest path finding problems has been extensively conducted. Most of the computer scientists focused on developing stochastic or deterministic models in transportation and particularly in communication systems, and a limited part of research was found to formulate the travel time associate with each street segment and also they mostly do not take into account the events that may affect traffic condition periodically.

CHAPTER 3

PROBLEM FORMULATION

3.1 Traffic Detection

The ultimate objective of this thesis is to develop a comprehensive system to route the traffic in an urbanized area. The development of this system stems from the combination of two distinctive sections namely traffic detection and routing system. The typical work flow is depicted in Figure 3.1.

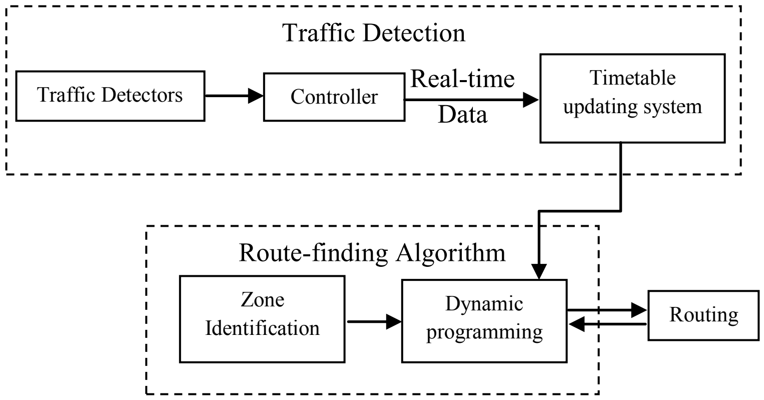


Figure 3.1 Work flow of a typical detection and routing system

In order to route the traffic within a city, it is required to access dynamic data that shows traffic flow condition using detectors located in all possible routes within a street network. According to the Federal Highway Administration (Mimbela et al, 2000), there are different forms of traffic detectors that include:

- Inductive Loop Detection System: The loop detector forms a tuned electrical circuit of which the loop wire is the inductive element. When a vehicle passes over the loop, it results in a decrease in loop inductance. The detector senses the change in inductance and causes the electronics unit to send a pulse to the controller, indicating the presence or passage of a vehicle.
- Video Image Processing System: It allows the user to define a limited number of linear detection zones on the roadway in the field-of-view of the video camera. When a vehicle crosses one of these zones, it is identified by noting changes in the properties of the affected pixels relative to their state in the absence of a vehicle. it estimates vehicle speed and then measure the time that an identified vehicle needs to traverse a detection zone of known length.
- Microwave Radar Based Traffic Detection System: It transmits energy toward an area of a roadway and when a vehicle crosses the roadway, a portion of the transmitted energy is reflected back toward the receiver and consequently it calculates volume, speed, vehicle length and occupancy.
- GPS-Based Vehicle Tracking System: A GPS data logger can be used to collect a vehicle's position data periodically. Typically, a GPS data logger comprises of three parts: data storage media, GPS receivers, and power supply devices
- Acoustic Traffic Detection System: it measures vehicle passage, presence, and speed by detecting acoustic energy or audible sounds produced by vehicular traffic from a variety of sources within each vehicle and from the interaction of a vehicle tires with the road.

- Infrared Traffic Detection System: it detects the changes of temperature due to vehicle presence.
- Magnetic traffic detection system: magnetic sensors are passive devices that detect the changes in magnetic field due to presence of a metal object.

All of the above mentioned detector systems can provide both the number of vehicles (volume) and speed of each vehicle in order to calculate the time each vehicle needs to cover a particular route. Currently, loop detectors are still the dominant detectors in use, because they have been commodity-priced while the alternative detectors have not due to the short history of these detectors (Lees, 2008). Hence in this research study it is assumed that inductance loop detectors are installed on each street segment. The components of a loop detector shown in Figure 3.2 are as follows:

- One or more turns of insulated loop wire wound in a shallow slot sawed in the pavement.
- A lead-in cable from the curbside pull box to the controller cabinet.
- A detector electronic unit (DEU) housed in the controller.

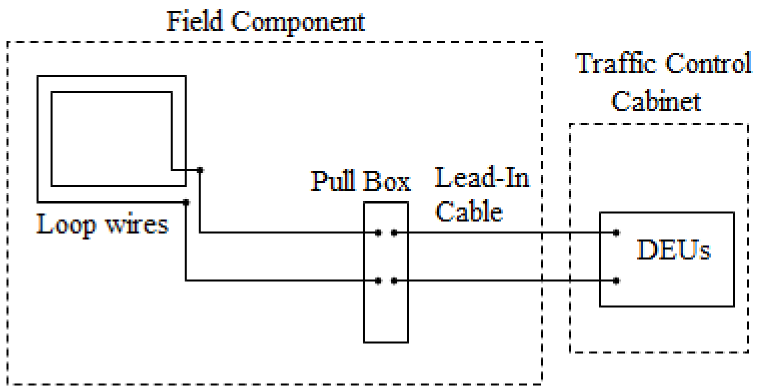


Figure 3.2 The components of a loop detector

An inductance loop detector will measure two important parameters for each 20 seconds time interval, namely:

- Occupancy; proportion of time during which the loop is occupied by a vehicle, (sec)
- Number of vehicles entering (*I*) or exiting (*E*) a street segment, (Veh/20 sec/ln)

Let's consider a model in which the average traffic speed associated with each street segment, $L(i,j)$ can be measured every 20 seconds and it is assumed that traffic detectors measure the number of vehicles every day for 24 hours continuously.

Let $D(i,,j,t_k)$ be density or the number of vehicles on street segment, $L(i,j)$, also let $I(i,,j,t_k)$ and $E(i,,j,t_k)$ be the number of vehicles (observed volumes), entering and exiting the street segment during time slot $\Delta t_k = t_k - t_{k-1}$ respectively. Now if we assume that there is no vehicle on each street segment at $t = t_0$, the number of vehicles during a defined time slot $\Delta t_k = 20$ seconds can be given as follows:

$$\Delta t_1 = (t_1 - t_0) = 20 \text{ sec.}; \quad D(i,j,t_1) = I(i,j,t_1) - E(i,j,t_1), \quad I(i,j,t_1) \geq E(i,j,t_1)$$

$$\Delta t_2 = (t_2 - t_1) = 20 \text{ sec.}; \; D(i,j,t_2) = D(i,j,t_1) + I(i,j,t_2) - E(i,j,t_2), \; D(i,j,t_2) + I(i,j,t_2) \geq E_2(i,j,t_2)$$

$$\vdots$$

$$\Delta t_k = (t_k - t_{k-1}) = 20 \text{ sec.}; \; D(i,j,t_k) = D(i,j,t_{k-1}) + I(i,j,t_k) - E(i,j,t_k), \; D(i,j,t_{k-1}) + I(i,j,t_k) \geq E(i,j,t_k)$$

Hence the average traffic speed $S(i,j,t_k)$ on each street segment at time t_k can be expressed as shown in Equation 3.1:

$$S(i,j,t_k) = S_f(i,j) * \left(1 - \frac{D(i,j,t_k)}{J(i,j)}\right) \qquad (3.1)$$

Where:

$J(i,j)$ = Jam density on street segment i, j, Veh/L(i,j)

$S_f(i,j)$ = Speed limit on street segment i, j, mi/h

Consequently the needed time to cover the street segment *L(i,j)* can be defined by Equations 3.2:

$$T(i,j,t_k) = \begin{cases} \dfrac{L(i,j)}{S(i,j,t_k)}, & D(i,j,t_k) \neq 0 \\ \\ \dfrac{L(i,j)}{S_f(i,j)}, & D(i,j,t_k) = 0 \end{cases} \tag{3.2}$$

Where:

t_k= the time at which the latest time table has been created

$T(i,j,t_k)$ = Average travel time on street segment i,j at time slot Δt_k

$S(i,j,t_k)$ = Traffic mean speed on street segment i,j at time slot Δt_k, mi/h

$L(i,j)$ = Length of street segment i,j, miles

It is noteworthy to mention that the traffic mean speeds or the average travel times related to different directions of a particular street segment as shown in Figure 3.3 are not identical.

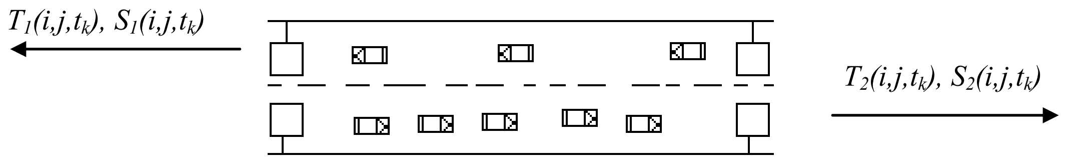

$T_1(i,j,t_k)$, $S_1(i,j,t_k)$
$T_2(i,j,t_k)$, $S_2(i,j,t_k)$

Figure 3.3 Traffic mean speed and the average travel time related to each lane

3.2 Timetable and Speedtable Updating System

As shown in Figure 3.4 timetable and speedtable updating system stores all estimated travel times and traffic mean speeds related to each route in previous days. A new time table and speed table with updated data will be created every 20 seconds. On the other hand the time interval between two timetables or speedtables is 20 second, hence in this study $t_0, t_1, \ldots, t_k$ are the times at which timetables and speedtables have been created. Such tables involve the characteristics of a street network, for example travel time and traffic mean speed are set to ∞ and 0 respectively for nodes not directly linked to each other ($T(i,j,t_k)= \infty, S(i,j,t_k)= 0 \text{ for } i \neq j$) . In additions the estimated travel time and traffic mean speed from one node to itself are equal to 0 and ∞ respectively ($T(i,j,t_k)=0, S(i,j,t_k)= \infty \text{ for } i=j$). Therefore each vehicle driver located at a node can communicate with the route finding system and ask for the shortest path toward his/her destination. This system will take care of the information provided by the timetable and speedtable updating systems and will select the shortest route among all possible routes. In this study $T(i,j,t)$ and $S(i,j,t)$ respectively stand for the travel time and traffic mean speed in current day and $Y'(i,j,t)$ and $S_y(i,j,t)$ respectively represent the travel time and traffic mean speed for the same day and same time in last week.

Speed Table, $S(i,j,t_k)$ *mi/hr*

Nodes	1	2	3	4
1	∞	0	S(1,3)	S(1,4)
2	S(2,1)	∞	0	0
3	S(3,1)	0	∞	S(3,4)
4	0	S(4,2)	S(4,3)	∞

Time Table, $L(i,j)$ *mile*, $T(i,j,t_k)$ *minute*

Nodes	1	2	3	4
1	0	∞	$60.\frac{L(1,3)}{S(1,3)}$	$60.\frac{L(1,4)}{S(1,4)}$
2	$60.\frac{L(2,1)}{S(2,1)}$	0	∞	∞
3	$60.\frac{L(3,1)}{S(3,1)}$	∞	0	$60.\frac{L(3,4)}{S(3,4)}$
4	∞	$60.\frac{L(4,2)}{S(4,2)}$	$60.\frac{L(4,3)}{S(4,3)}$	∞

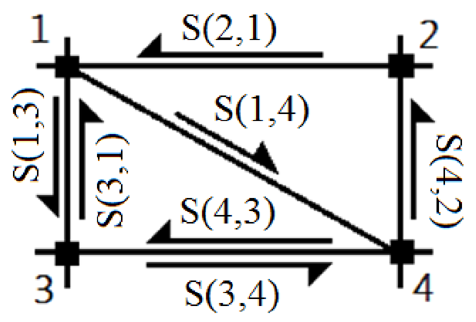


Figure 3.4 Speedtable and timetable in the proposed network

3.3 Design of a Network

A street network *G(N,l)* is a type of directed, weighted graph consisting of the following elements:

- **A set of nodes (*N*):** A node is a terminal point or an intersection point of a graph. In this study it is the abstraction of a street intersection.
- **A set of links (*l*):** A link is a connection between nodes *i* and *j* and it is the abstraction of a street segment.

- **Sub-graph:** A sub-graph is a subset of a particular graph. In this study **each street network** consists of many sub-networks. The street network itself is a sub-network of regional transportation network.
- **Buckle:** A link that makes a node correspond to itself is a buckle. In this study the cost (travel time) of the buckles are zero.

In addition a street network lets the traffic flows move across the network, therefore movements should be represented as linkages, which can be considered over several aspects:

- **Path:** A sequence of links that are traveled in the same direction. For a path to exist between two nodes, it must be possible to travel an uninterrupted sequence of links.
- **Length of a link:** It is the distance associated with a link, a connection or a path.

It is noteworthy to mention that that the position of nodes or intersections should be identified based on the state plane coordinate system. The state plane coordinate system is a coordinate systems designed for a specific region of the United States. Each state may have one or two state plane coordinate systems (e.g., Ohio south state plane and Ohio north state plane coordinate system). In addition, it is a system for specifying positions of geodetic stations using Cartesian or plane rectangular coordinates rather than spherical coordinates (the geographic coordinate system of latitude and longitude).

3.4 Zone Identification and Nodes Selection Algorithm

The shortest path finding algorithm is a heuristic algorithm that tries to examine a finite set of nodes between source and destination points in order to identify the nodes that creates the path with the shortest travel time. Due to the large volume of street intersections in a network, it is useful to design a zone identification algorithm that limits or reduces the number of nodes in order to increase the algorithm efficiency explained in Section 3.2. Reference to Figure 3.5, assume that a driver requests a path with a travel time less than 5 minutes between a source node *s* and a destination node *d*, then the zone identification algorithm takes care of the nodes located in the smallest area (blue rectangular zone) that encompasses source $s(x_1,y_1)$ and destination $d(x_2,y_2)$, the dynamic programming technique will examine the selected nodes in order to detect the shortest path. If the detected shortest path has a travel time longer than the travel time requested by the driver (5 minutes), the dynamic programming technique will incrementally expand the search radius (R) and look for the shortest path in a larger sub-area (see Figure 3.6). For high speed algorithms there is no real advantage to using search radius.

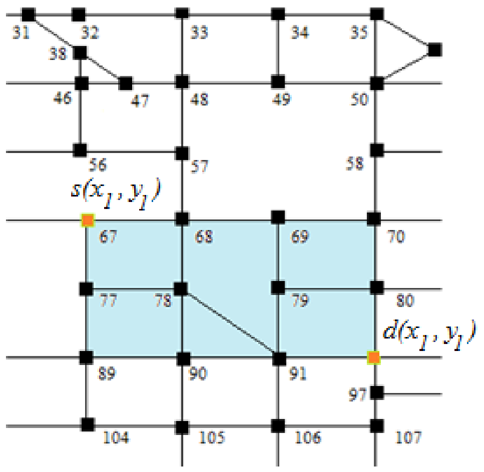


Figure 3.5 An example of street network

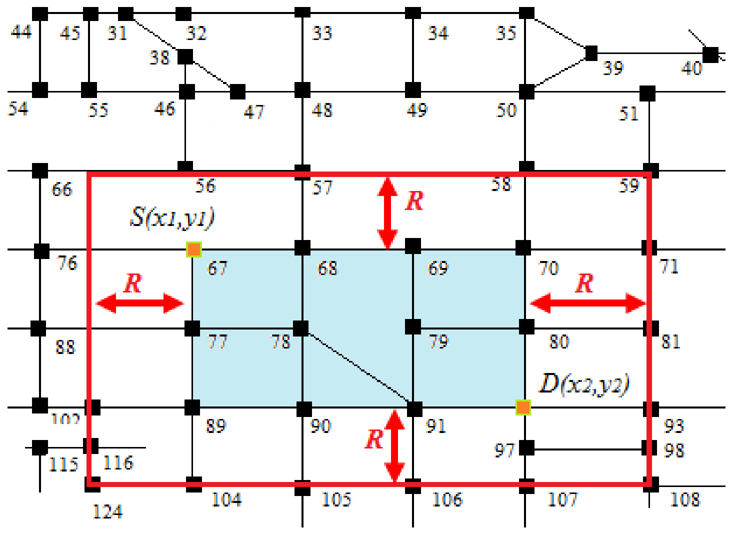


Figure 3.6 Extended zone

In order to define a strategy to identify the zone in which desired nodes are located, the following four possible trip patterns are taken into account:

- If $x_1 \leq x_2$ and $y_1 \geq y_2$ then the location of the selected node $N_i(x_i,y_i)$, can be defined by:

$$x_1 - R \leq x_i \leq x_2 + R \text{ and } y_2 - R \leq y_i \leq y_1 + R \quad \text{(See Figure 3.7a)}$$

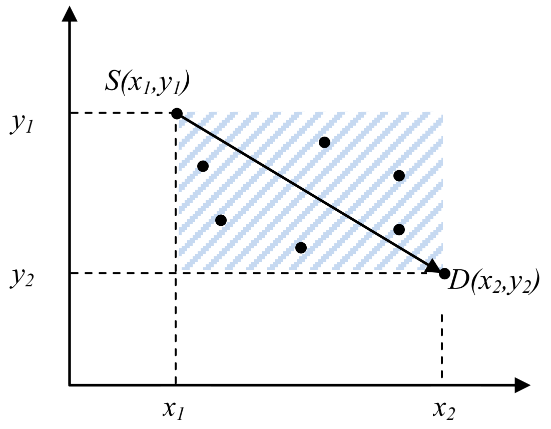


Figure 3.7a Travel pattern I

- If $x_2 \leq x_1$ and $y_1 \leq y_2$ then:

$x_2-R \leq x_i \leq x_1+R$ and $y_1-R \leq y_i \leq y_2+R$ (See Figure 3.7b)

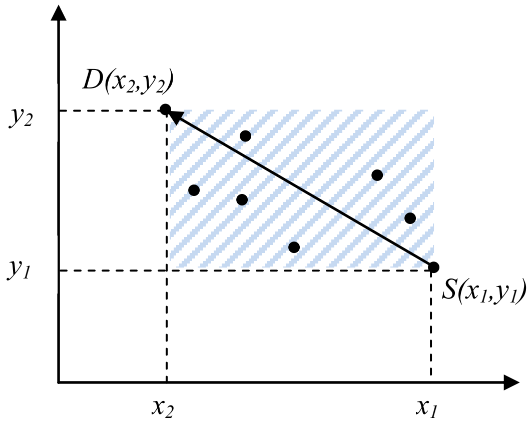


Figure 3.7b Travel pattern II

- If $x_1 \leq x_2$ and $y_1 \leq y_2$ then:

$x_1-R \leq x_i \leq x_2+R$ and $y_1-R \leq y_i \leq y_2+R$ (See Figure 3.7c)

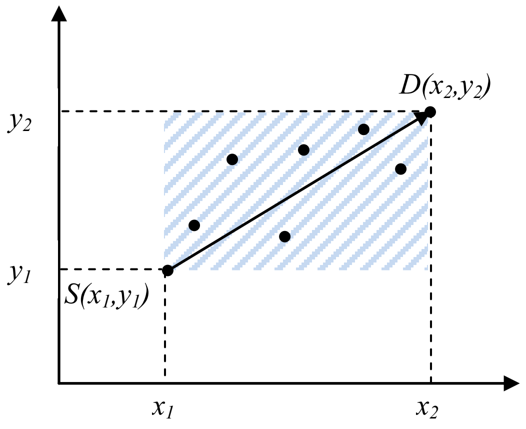


Figure 3.7c Travel pattern III

- And finally, If $x_1 \geq x_2$ and $y_1 \geq y_2$ then:

$$x_2-R \leq x_i \leq x_1+R \text{ and } y_2-R \leq y_i \leq y_1+R \quad \text{(See Figure 3.7d)}$$

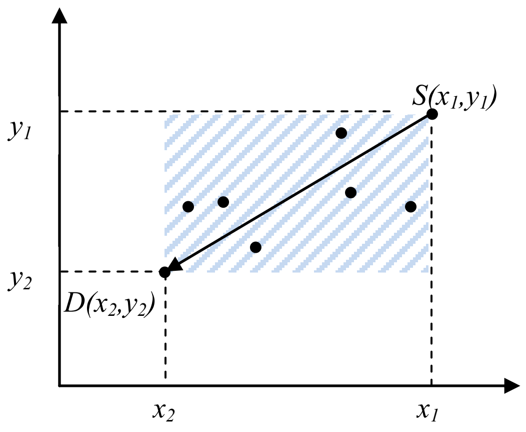


Figure 3.7d Travel pattern IV

Since each node N_i has its unique address $i=1,2,\ldots n$, therefore the set of selected nodes in the identified zone encompassing the initial point *"S"* and destination point *"D"* is:

$$\text{N} = \{S, N_2, N_3, \ldots, D\}, \quad N_1 = S \text{ and } N_{n+1} = D$$

A typical pseudo code used for the identification and selection of nodes for solving such problems mentioned above is presented in Figure 3.8.

```
Start;
Declare Source Point, S($x_s$,$y_s$)
Declare Destination Point, D($x_d$,$y_d$)
Let k=1
for i=1 to n do
Declare Point  $P_i$($x_i$,$y_i$)
If ($x_s$-R≤$x_i$≤R+ $x_d$) and ($y_d$-R≤$y_i$≤R+ $y_s$) then
$N_k$=i
Let  k=k+1
Else if ($x_d$-R ≤$x_i$ ≤ R+ $x_s$) and ($y_s$-R≤$y_i$ ≤R+$y_s$)  then
$N_k$ =i;
Let  k=k+1;
Else if ($x_s$ -R≤$x_i$ ≤R+ $x_d$) and ($y_s$-R≤$y_i$ ≤ R+$y_d$) then
$N_k$ =i;
Let k=k+1;
Else
$N_k$ =i;
Let k=k+1;
End if
End
```

Figure 3.8 A typical pseudo code of zone identification and node selection

3.5 Numerical Methods of Optimization

In mathematics, optimization means finding a value of *x* which maximizes or minimizes a given function *f (x)* (Gordon, 1998). Some of the commonly used optimization methods available include the following:

(i) Linear programming: studies the case in which the objective function *f* is linear and set *A* is specified using only linear equalities and inequalities (*A* is the design variable space).

(ii) Integer programming: studies linear programs in which some or all variables are constrained to take on integer values.

(iii) Quadratic programming: allows the objective function to have quadratic terms, while the set A must be specified with linear equalities and inequalities

(iv) Nonlinear programming: studies the general case in which the objective function or the constraints or both contain nonlinear parts.

(v) Stochastic programming: studies the case in which some of the constraints depend on random variables.

(vi) Combinatorial optimization: is concerned with problems where the set of feasible solutions is discrete or can be reduced to a discrete one.

(vii) Dynamic programming**:** studies the case in which the optimization strategy is based on splitting the problem into smaller overlapping sub-problems. Dynamic programming is the method employed in this research study to find the shortest path problem, which is explained in more in details in Section 3.6.

3.6 Dynamic Programming and Real Time Routing

As previously mentioned, the zone identification algorithm takes into account appropriate nodes in order to be used in route finding algorithm, then the route finding algorithm examines all possible paths passing through the selected nodes by considering the link weights (travel times) stored in time tables. Furthermore, this algorithm uses a dynamic programming (DP) technique implemented by Bellman (1957) in selecting the path with shortest travel time based on the estimated travel times. It is noteworthy to mention that one of the advantages of dynamic programming is that the travel time modeling concept explained in Section 3.7 is applicable to DP technique.

Dynamic programming was systematized by Bellman (1957) for efficiently solving a broad range of search and optimization problems. It can be used when the solution to a problem such as shortest path finding problem can be viewed as the result of a sequence of decisions (tree) as shown in Figure 3.9 below.

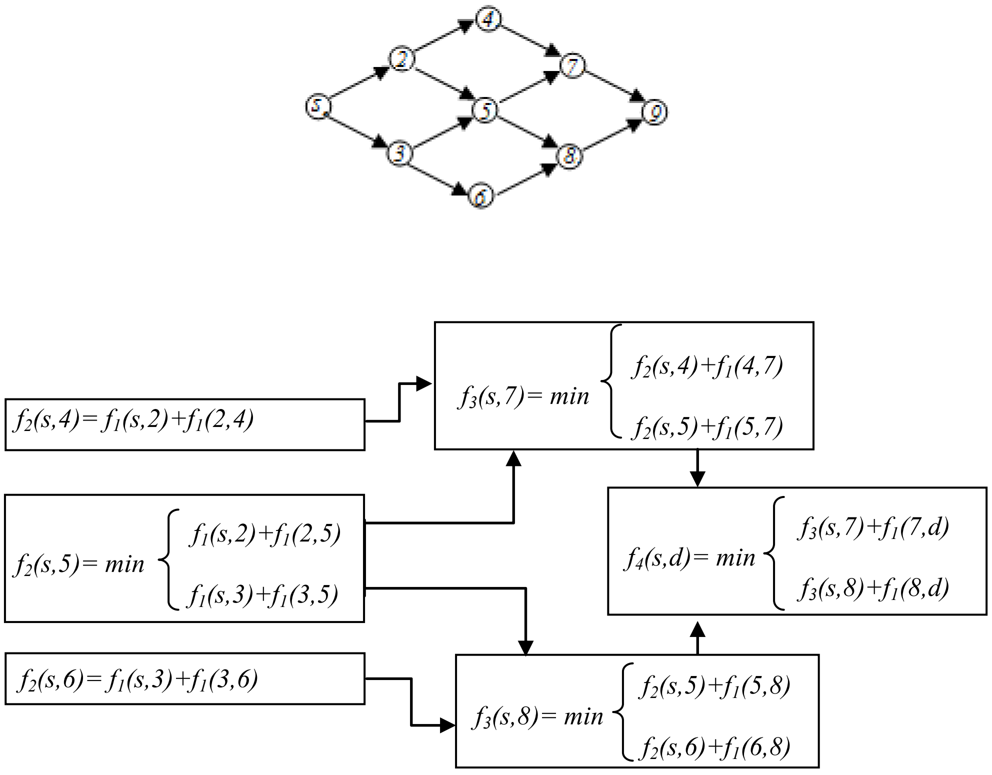

s
2
3
4
5
6
7
8
9
$f_2(s,4)=f_1(s,2)+f_1(2,4)$
$f_3(s,7)=\min\begin{cases}f_2(s,4)+f_1(4,7)\\f_2(s,5)+f_1(5,7)\end{cases}$
$f_2(s,5)=\min\begin{cases}f_1(s,2)+f_1(2,5)\\f_1(s,3)+f_1(3,5)\end{cases}$
$f_4(s,d)=\min\begin{cases}f_3(s,7)+f_1(7,d)\\f_3(s,8)+f_1(8,d)\end{cases}$
$f_2(s,6)=f_1(s,3)+f_1(3,6)$
$f_3(s,8)=\min\begin{cases}f_2(s,5)+f_1(5,8)\\f_2(s,6)+f_1(6,8)\end{cases}$

Figure 3.9 An example of a decision tree indicating overlapping sub-problems

As an important point this method is based on the principle of optimality, expressed in the form of a functional equation. According to Chinneck (2010) the overview of the dynamic programming method can be outlined as follows:

1. Break down a complex problem into simpler sub problems and try to solve them.
2. Enlarge each small part slightly and find the optimum solution to the new problem using the previously found optimum solution.
3. Continue with Step 2 until you have sufficiently enlarged the sub-problem such that the current problem encompasses the original problem. When this problem is solved, then the stopping conditions have been met.
4. Track back the solution to the whole problem from the optimum solutions to the small problems solved along the way.

Now by reconsidering Figure 3.9, suppose that a driver is on node *s* and he/she decides to get to node *j* in *n* steps. First, the driver needs to proceed optimally in *n-1* steps in order to get to some node *i,* and then he/she needs to choose node *j* to go to; the total travel time of going from *s* to *i* in *n-1* steps using an optimal policy is $f_{n-1}(s,i)$ and the cost of going from *i* to *j* is $T(i,j,t_k)$ which uses just one step. Therefore the shortest travel time based on real time data on a dynamic network with a finite set of nodes N = {s,...,d}, where the number of nodes are n+1, can be estimated as follows:

For a *1-step path* with minimum travel time, $f_1(s,j) = T\,(s,j,t_k)$

For a *2-step path* with minimum travel time, $f_2(s,j) = \min_i\ [f_1(s,i)+T(i,j,t_k)]$

For a *3-steps path*, $f_3(s,j)=\min_i [f_2(s,i)+T(i,j,t_k)]$

-
-
-

For an *n-steps path*, $f_n(s,j)=\min_i [f_{n-1}(s,i)+T(i,j, t_k)]$

Finally we can develop Equations 3.3 and 3.4 as follows:

$$f_1(s,j)= \begin{cases} T(s,j,t_k) & if\ s\ is\ connected\ to\ j \\ 0 & if\ s=j \\ \infty & if\ s\ is\ not\ connected\ to\ j \end{cases} \tag{3.3}$$

$$\underset{n\geq 2}{f_n(s,j)}= \min_i \begin{cases} f_{n-1}(s,i)+T(i,j,t_k) & if\ i\ is\ connected\ to\ j \\ f_{n-1}(s,i) & if\ i=j \\ \infty & if\ i\ is\ not\ connected\ to\ j \end{cases} \tag{3.4}$$

As explained in Section 3.1, $T(i,j,t_k)$ is a travel time or a weight associated with the street segment $L(i,j)$, by replacing $T(i,j,t_k)$ with Equation 3.2 in the above equations, we obtain Equations 3.5 and 3.6:

$$f_1(s,j) = \begin{cases} \dfrac{L\,(s,j)}{S\,(s,j,t_k)} & if\ \ s\ is\ connected\ to\ j \\ 0 & if\ \ s=j \\ \infty & if\ s\ is\ not\ connected\ to\ j \end{cases} \tag{3.5}$$

$$f_n(s,j)=\min_{i,\ n\geq 2} \begin{cases} f_{n-1}(s,i)+\dfrac{L\,(i,j)}{S\,(i,j,t_k)} & \text{if } i \text{ is connected to } j \\ f_{n-1}(s,i) & \text{if } i=j \\ \infty & \text{if } i \text{ is not connected to } j \end{cases} \qquad (3.6)$$

It is worthy to note that if $i = j$ then $S(i,j,t_k) = \infty$ or $T(i,j,t_k) = 0$, and if nodes i and j are two different points not directly connected to each other, then $S(i,j,t_k) = 0$ or $T(i,j,t_k) = \infty$. Figure 3.10 represents a typical pseudo code of real time routing algorithm. In General, monitoring the traffic changes and consequently getting access to the real time data is widely applicable in real time routing evacuation operations or for dynamic logistics routing and scheduling.

```
Start;
Declare Selected Nodes, N=[s $N_2$ $N_3$ … d]    {$N_1$=s} and {$N_n$=d}
Declare Travel Times Matrix, Y
Declare Clock Time Matrix, Time=[H M S]  {H: hour, M: minute, S=second}
u=floor[(H*3600)+(M*60)+(S) /20]  {floor rounds the value  element to the nearest integer equal
or less than  [(H*3600)+(M*60)+(S) /20] , u addresses the last timetable created at the time of
request}
for  j=1:size(N) do        {Size(N) indicates the number of nodes selected}
   $f_1(1,j)=T_u (N_1,N_j)$;
end
for  j=1:size(N,2) do
   J(1,j)=1;
end
for n=1:size(N)-2 do
    for d=1:size(N) do
    a=inf;
        for j=1:size(N,2) do
            if $f_n(1,j)+ T_u(N_j,N_d)\leq a$ then
                a= $f_n(1,j)+ T_u(N_j,N_d)$;
                $J_{n+1}(i)=j$
                $f_{n+1}(1,i)=a$;
            end
        end
    end
end
v=size(N)
for k=1:size(N)-1 do
   $v_k$=J(,v);
end
p=fliplr(p)
```

Figure 3.10 A typical pseudo code of real time routing algorithm.

3.7 Travel Time Modeling

Travel time modeling is used to estimate the travel time associated with each street segment by taking into account the events that periodically cause traffic congestion (e.g., going to work at a specific time). Here days are categorized, e.g. Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday. It is reasonable to assume that for two days in the same category, a particular road segment has approximately the same traffic volume or travel time at the same time of the day. Furthermore, the travel time can be adjusted by employing a multiplier, which will be explained in this section.

By referring to Figure 3.11, assume that a week ago on Monday a driver on point *1* decided to go to point *n*. Now, we want to estimate the travel time he/she spent in traveling from point *1* at time u_1 to point *n*.

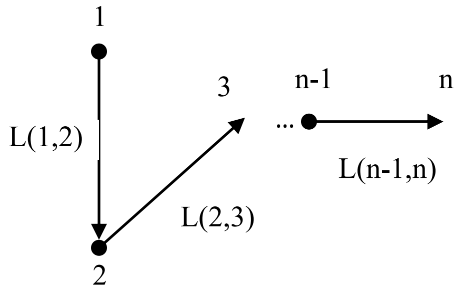


Figure 3.11 An example of street segments traveled by the driver

Segment L(1,2)

Consider a street segment *L(1,2)* shown in Figure 3.12 and suppose that a driver was at point *1* at time u_1 (i.e., $t_0 < u_1 < t_1$). Therefore, $\acute{Y}(1,2,u_1)$, which is the time that takes the driver to arrive at point *2* at time u_2 ($t_{m-1} < u_2 < t_m$), can be shown by Equation 3.7. Note that $t_0, t_1, \ldots, t_m, \ldots, t_k$ are the times at which timetables and speedtables have been created.

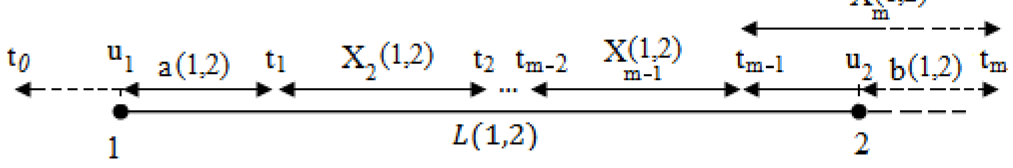


Figure 3.12 Street segment L(1,2)

$$\acute{Y}(1,2,u_1) = (t_1 - u_1) + (t_2 - t_1) + \cdots + (t_m - t_{m-1}) - (t_m - u_2) \quad \textit{(3.7)}$$

$$\acute{Y}(1,2,u_1) = (t_1 - u_1) + (t_m - t_1) - (t_m - u_2)$$

$$\acute{Y}(1,2,u_1) = (t_1 - u_1) + 20 * (m - 1) - (t_m - u_2) \quad (3.8)$$

Where:

$$(t_m - t_1) = 20 * (m - 1) \text{ , and}$$

$$(t_m - u_2) = \frac{b(1,2)}{S(1,2,t_m)}$$

In these equations t_1 and u_1 are known but *b* and *m* are unknown. First, *m* should be identified in order to find *b,* finding the value of *m* allows us to find the time t_m, the timetable created at time t_m and finally, the traffic mean speed $S(1,2,t_m)$ during $(t_m\text{-}t_{m\text{-}1})$.

Let $S_y(1,2,t_1)$ be the average traffic mean speed (*miles/hr*) and $a(1,2)$ be the distance (*miles*) traveled during $(t_1 - u_1)$ on segment $L(1,2)$, therefore $a(1,2)$ can be given as depicted in Equation 3.9:

$$a(1,2)=(t_1 - u_1)*S_y(1,2,t_1) \quad (3.9)$$

Then, let $X_2(1,2)$ be the distance traveled during the second time slot $(t_1 - t_2) = 20$ seconds and $S_y(1,2,t_2)$ be the average traffic mean speed on segment $L(1,2)$ during $(t_2 - t_1)$. Then we can get Equation 3.10.

$$X_2(1,2)=20*S_y(1,2,t_2) \quad (3.10)$$

Finally, let $S_y(1,2,t_m)$ be the average traffic mean speed and $X_m(1,2)$ be the distance traveled during (t_m - t_{m-1}) on segment $L(1,2)$, then we can get Equation 3.11:

$$X_m(1,2) =(t_m- t_{m-1})*S_y (1,2,t_m) \qquad (3.11)$$

Where m is the smallest possible value such that:

$$a(1,2) + X_2(1,2) + X_3(1,2) + \cdots + X_m(1,2) \geq L(1,2)$$

This leads into Equation 3.12.

$$b(1,2) = a(1,2) + X_2(1,2) + X_3(1,2) + \cdots + X_m(1,2) - L(1,2) \qquad (3.12)$$

$$or$$

$$b(1,2) = \ a(1,2) + \sum_{k=2}^{m} X_k\,(1,2) - L(1,2)$$

Therefore, by replacing $X_1(1,2)$, $X_2(1,2)$ and $X_3(1,2)$ by Equations 3.9, 3.10 , and 3.11, respectively, then $b(1,2)$ can be given as follows:

$$b(1,2) = [\,S_y\,(1,2,t_1) * (t_2 - u_1) + 20 * S_y\,\,(1,2,t_2) + \cdots + 20 * S_y\,(1,2,t_m)] - L(1,2)$$

$$or$$

$$b(1,2) = [\,S_y\,(1,2,t_1) * (t_2 - u_1) + 20\textstyle\sum_{k=2}^{m} S\,(1,2,t_k)] - L(1,2) \qquad (3.13)$$

Thus, if last Monday a driver arrived at point *2* at time $u_2 = u_1 + \acute{Y}(1,2,u_1)$, now this Monday (today) if the driver arrives at point *1* at time u_1, it is expected that this driver will arrive at point *2* at time $u_2' = u_1 + \hat{T}(1,2,u_1)$, where $\hat{T}(1,2,u_1)$ is forecasted (adjusted) travel time, needed to cover street segment *L(1,2)*.

$$\hat{T}(1,2,u_1) = \lambda(1,2,u_1) * \acute{Y}(1,2,u_1) \qquad (3.14)$$

$$\lambda(1,2,u_1) = \frac{\bar{T}(1,2,u_1)}{\bar{Y}(1,2,u_1)} \qquad (3.15)$$

Where:

$\acute{Y}(1,2,u_1)$ = computed travel time spent to cover street segment *L(1,2)* on Monday last week.

$\lambda(1,2,u_1)$ = a travel time adjustment factor associated with street segment *L(1,2)* at the time of requesting the shortest path (u_1).

$\bar{T}(1,2,u_1)$ = the hourly average travel time associated with street segment *L(1,2)* based on the last timetables established today (Monday) before the time of requesting the shortest path (u_1). Since each time table will be created every 20 seconds, the total number of timetables in one hour is 180.

$\bar{Y}(1,2,u_1)$ = the average travel time associated with street segment *L(1,2)* based on the last timetables established before time u_1 on Monday last week.

For example, if right now a driver is on point *1* at time *11:30:14*, so the last timetable and speedtable have been created at *=11:30:00*, therefore $\bar{T}$ *(1,2,11:30:14)* is the hourly average travel time related with segment *L(1,2)* based on the timetables created at t_1=*10:30:00* , t_2=*10:30:20,…,* t_{180}=*11:30:00* .

$$\bar{T}(1,2,11{:}30{:}14) = \frac{T(1,2,t_1) + T(1,2,t_2) + \cdots + T(1,2,t_{180})}{180} = \frac{1}{180}\sum_{k=1}^{180} T(1,2,t_k)$$

Furthermore, $\bar{Y}$*(1,2,11:30:14)* is the hourly average travel time related with segment *L(1,2)* based on the timetables created at t_1*=10:30:00*, t_2*=10:30:20,…,* t_{180}*=11:30:00* in last week.

$$\bar{Y}(1,2,11{:}30{:}14) = \frac{Y(1,2,t_1) + Y\,(1,2,t_2) + \cdots + Y\,(1,2,t_{180})}{180}$$

$$= \frac{1}{180}\sum_{k=1}^{180} Y\,(1,2,t_k)$$

If we assume that:

L(1,2)=300 meters,

$S(1,2,t_1)= S(1,2,t_2)=\ldots= S(1,2,t_{180})=15\ m/s,$ and

$S_y\,(1,2,t_1)= S_y(1,2,t_2)=\ldots= S_y(1,2,t_{180}))=10\ m/s$

Then:

$$T(1,2,t_1)= T(1,2,t_2)=\ldots= T(1,2,t_{180})= \frac{300}{15}= 20\ sec.$$

$$Y(1,2,t_1)= Y(1,2,t_2)=\ldots= Y(1,2,t_{180})= \frac{300}{10} = 30\ sec$$

$$\bar{T}(1,2,11{:}30{:}14) = \frac{1}{180}\sum_{k=1890}^{2070} T_k(1,2) = 20\ Sec.$$

$$\bar{Y}(1,2,11{:}30{:}14) = \frac{1}{180}\sum_{k=1890}^{2070} Y_k(1,2) = 30\ Sec.$$

$$\lambda(1,2,11{:}30{:}14) = \frac{\bar{T}(1,2,11{:}30{:}14)}{\bar{Y}(1,2,11{:}30{:}14)} = 0.67$$

Then, using Equation 3.8, $\acute{Y}(1,2,11{:}30{:}14)$ can be estimated as follows:

$$\acute{Y}(1,2,11{:}30{:}14) = (t_1 - u_1) + 20 * (m-1) - \left(\frac{b(1,2)}{S(1,2,t_m)}\right)$$

Where $t_1 = 11{:}30{:}20$ and $u_1 = 11{:}30{:}14$, then $(t_1 - u_1) = 6\ sec.$

In order to find the *m* value we should look for the smallest possible value such that:

$$a(1,2) + X_2(1,2) + \cdots + X_m(1,2) \geq L(1,2) = 300$$

The distance traveled by the driver during time period $(t_1 - u_1) = 6\ seconds$, i.e., *a(1,2)* is given by:

$$a(1,2) = (t_1 - u_1) * Sy(1,2,t_1) = 6*10 = 60, \quad \text{where } a(1,2) = 60 < L(1,2) = 300$$

The distance traveled by the driver during time period $(t_2 - t_1) = 20\ sec.$, i.e., $X_2(1,2)$:

$$X_2(1,2) = (t_2 - t_1) * Sy(1,2,t_2) = 20*10 = 200, \quad \text{where } a(1,2) + X_2(1,2) = 260 < L(1,2) = 300$$

But, the distance traveled by the driver during time period $(t_3 - t_2) = 20\ sec.$, was X_3:

$$X_3(1,2) = (t_3 - t_2) * Sy(1,2,t_3) = 20*10 = 200, \quad \text{where } a(1,2) + X_2(1,2) + X_3(1,2) = 460 > L(1,2) = 300$$

The sum of $a(1,2) + X_2(1,2) + X_3(1,2)$ exceeds the length of street segment *L(1,2)*, therefore $m = 3$, and by using Equation 3.12 we can get:

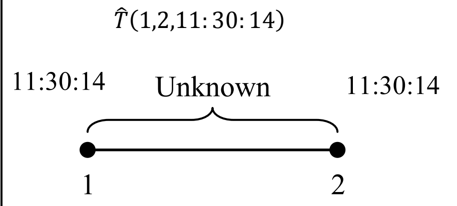
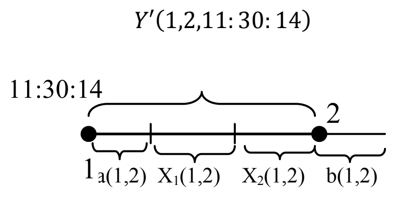
$b = a(1,2) + X_2(1,2) + X_3(1,2) - L(1,2) = 460 - 300 = 160$, therefore the estimated travel time is as follows:

$$\acute{Y}(1,2,11{:}30{:}14) = 6 + 20 * (3-1) - \left(\frac{160}{10}\right) = 30$$

Therefore, as shown in Figure 3.13, the forecasted travel time is given as:

$$\hat{T}(1,2,11{:}30{:}14) = \lambda(1,2,11{:}30{:}14) * \acute{Y}(1,2,11{:}30{:}14) = (0.67) * (30) = 20$$

Today (Monday)	Monday (Last week)

 $\hat{T}(1,2,11{:}30{:}14)$
11:30:14
Unknown
11:30:14
1
2
	 $Y'(1,2,11{:}30{:}14)$
	11:30:14
	2
	1
	a(1,2)
	X1(1,2)
	X2(1,2)
	b(1,2)

Average travel time From t_1=10:30:00 to t_{180}=11:30:00 $\bar{T}(1,2,11{:}30{:}14) = \frac{T(1,2,10{:}30{:}00) + \cdots + T(1,2,11{:}30{:}00)}{180}$	Average travel time From t_1=10:30:00 to t_{180}=11:30:00 $\bar{Y}(1,2,11{:}30{:}14) = \frac{Y(1,2,10{:}30{:}00) + \cdots + Y(1,2,11{:}30{:}00)}{180}$

$$\hat{T}(1,2,11{:}30{:}14) = \frac{\bar{T}(1,2,11{:}30{:}14)}{\bar{Y}(1,2,11{:}30{:}14)} * Y'(1,2,11{:}30{:}14)$$

Figure 3.13 Forecasted travel time computation

It is noteworthy to mention that in this study all days are assumed to be normal typical days; therefore, calculating the adjustment factor and the expected travel time based on historical information allows us to take into account traffic congestion that happens during a defined time period, although expected travel time is in our interest for long-distance trips, but for short-distance trips, it may be better to rely more on real time data and real time routing as described in Section 3.6.

Segment L(2,3)

Referring to Figure 3.14, if last week a driver was at point *2* at time u_2', he/she would have arrived at point 3 at time $u_3 = u_2' + \acute{Y}(2,3,u_2')$, and:

$$\acute{Y}(2,3,u_2') = (t_m - u_2') + 20(q-m) - \left(\frac{b(2,3)}{S_y(2,3,t_q)}\right) \qquad (3.16)$$

Where q is the possible smallest value such that;

$$S_y(2,3,u_m) * (t_m - u_2') + 20 \sum_{k=m+1}^{q} S_y\ (2,3,t_k) \geq L(2,3)$$

and:

$$b(2,3) = [S_y(2,3,u_m) * (t_m - u_2') + 20 \sum_{k=m+1}^{q} S_y\ (2,3,t_k)] - L(2,3) \qquad (3.17)$$

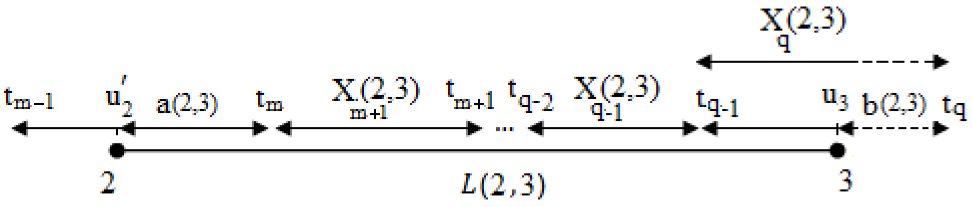


Figure 3.14 Street segment L(2,3)

Hence, the driver arrived at point *3* at time $u_3 = u_2' + \acute{Y}(2,3,u_2')$, but today it is expected that the driver will arrive at point *3* at time $u_3' = u_2' + \hat{T}(2,3,u_2')$, where $\hat{T}(2,3,u_2')$ is forecasted (adjusted) travel time, needed to cover street segment *L(2,3)*

$$\hat{T}(2,3,u_2') = \lambda(2,3,u_1) * \acute{Y}(2,3,u_2') \qquad (3.18)$$

$$\lambda(2,3,u_1) = \frac{\bar{T}(2,3,u_1)}{\bar{Y}(2,3,u_1)} \qquad (3.19)$$

Segment L(n-1,n)

Referring to Figure 3.15, assuming the same day last week a driver was at point *n-1* at time u_{n-1}', he/she would arrive at point *n* at time $u_n = u_{n-1}' + \acute{Y}(n-1,n,u_{n-1}')$, and:

$$\acute{Y}(n-1,n,u'_{n-1}) = (t_r - u'_{n-1}) + (t_w - t_r) - \left(\frac{b(n,n-1)}{S_y\,(n-1,n,t_w)}\right) \qquad (3.20)$$

Where *w* is the smallest possible value such that:

$$S_y(n-1,n,\ u'_{n-1}) * (t_r - u'_{n-1}) + 20\sum_{k=r}^{w} S_y\ (n-1,n,t_k) \geq L(n-1,n)$$

And;

$$b(n,n-1) = [S_y(n-1,n,\ u'_{n-1}).\,(t_r - u'_{n-1}) + 20\sum_{k=r+1}^{w} S_y\ (n-1,n,t_k)] - L(n,n-1) \quad (3.21)$$

Or

$$b(n,n-1) = [a(n-1,n) + \sum_{k=r+1}^{w} X_k\ (n-1,n)] - L(n,n-1)$$

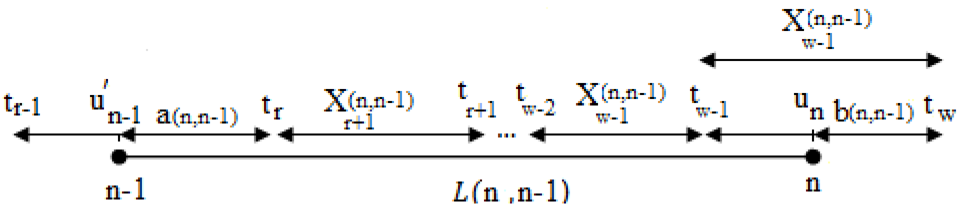


Figure 3.15 Street segment L(n,n-1)

. Therefore, today the driver may arrive at point *n* at time $u'_n = u'_{n-1} + \hat{T}(n-1,n,u'_{n-1})$.

$$\hat{T}(n-1,n,u'_{n-1}) = \lambda(n-1,n,u_1) * \acute{Y}(n-1,n,u'_{n-1}) \qquad (3.22)$$

$$\lambda(n-1,n,u_1) = \frac{\bar{T}(n-1,n,u_1)}{\bar{Y}(n-1,n,u_1)} \qquad (3.23)$$

Consequently, by applying the travel time modeling to Equations 3.3 and 3.4, we can get:

$$f_1(s,j,u_1) = \begin{cases} \hat{T}(s,j,u_1) & \text{if } s \text{ is connected to } j \\ 0 & \text{if } s = j \\ \infty & \text{if } s \text{ is not connected to } j \end{cases} \tag{3.24}$$

$$\underset{n\geq 2}{f_n(s,j,u_1)} = \min_i \begin{cases} \left[f_{n-1}(s,i,\ u_1) + \hat{T}(i,j,\alpha)\right] & \text{if } i \text{ is connected to } j \\ f_{n-1}(s,i,u_1) & \text{if } i = j \\ \infty & \text{if } i \text{ is not connected to } j \end{cases} \tag{3.25}$$

Where $\alpha = u_1 + f_{n-1}(s,i,\ u_1)$. A typical pseudo code for route finding algorithm based on the travel time modeling is depicted in Figure 3.16.

```
Start;
Declare Selected Nodes, N=[s $N_2$ $N_3$ … d]     {$N_2$=s}
Declare Speedtable  Matrix, Sy
Declare Clock Time Matrix, Time=[H M S]  {H: hour, M: minute, S=second}
$u_1$=(H*60)+M+(S/60)  {$u_1$ is clock time in terms if Minutes at the time of request}
u=(ceil($u_1$*60/20))     {u addresses the speedtable created  at $t_1>u_1$ in the same day in last week, ceil
rounds the element to the nearest integer greater than or equal  to ($u_1$*60/20)}
$t_1$=u*20/60               {the time that speedtable created}
dt=($t_1$-$u_1$)/3600
for  j=1:size(N) do       {Size(N) indicates the number of nodes selected}
   if $Y_u$ ($N_1$,$N_j$)=inf then
     $f_l$(1,j)=inf
   elseif $Y_u$ ($N_1$,$N_j$)=0
    $f_l$(1,j)=0
   else
     X=$Sy_u$ ($N_1$,$N_j$).dt
     L=||$N_1$-$N_j$||              {distance between node 1 and node j}
   If  X≤ L  then
       Let  k=u+1
       Y'($N_1$,$N_j$)=dt
     while X≤ L do
         X=X+(20/3600)* $Sy_k$ ($N_1$,$N_j$)
         Y'($N_1$,$N_j$)= Y'($N_1$,$N_j$)+(20/3600)
         Let  k=k+1
     end
         Let k=k-1
        Y'($N_1$,$N_j$)= Y'($N_1$,$N_j$)-[(L-X)/$Sy_k$($N_1$,$N_j$)
    end
        v=u
        $\bar{Y} = 0$
        $\bar{T} = 0$
   for k=v-181:v-1 do
        $\bar{Y} = \bar{Y}$+ $Y_k$ ($N_1$,$N_j$)
        $\bar{T} = T$+ $T_k$ ($N_1$,$N_j$)
   end
  $f_l$(1,j)= $\bar{Y}$ *Y'($N_1$,$N_j$)*60/$\bar{T}$
  end
for  j=1:size(N) do
   J(1,j)=1;
end
for n=1:size(N)-2 do
    for d=1:size(N) do
    a=inf;
```

```
 for j=1:size(N) do
  if $Y_u$ ($N_j$,$N_d$)=inf then
     $f_l$(j,d)=inf
  elseif $Y_u$ ($N_j$,$N_d$)=0
    $f_l$(j,d)=0
  else
     b=$f_n$(1,j)
     u'=$u_l$+ $f_n$(1,j)
     u=(ceil(u'*60/20))
     $t_l$=u*20/60
    dt=($t_l$-u')/3600
   X=$Sy_u$ ($N_j$,$N_d$).dt
   L=||$N_j$-$N_d$||             {distance between node j and node d}
  If  X≤ L
      k=u+1
     Y'($N_j$,$N_d$)=dt
   while X≤ L
       X=X+(20/3600)* $Sy_k$ ($N_j$,$N_d$)
       Y'($N_j$,$N_d$)= Y'($N_1$,$N_j$)+(20/3600)
        k=k+1
   end
       k=k-1
      Y'($N_j$,$N_d$)= Y'($N_1$,$N_j$)-[(L-X)/$Sy_k$($N_j$,$N_d$)
  end
      v=u
      $\bar{Y} = 0$
     $\bar{T} = 0$
  for k=v-181:v-1
      $\bar{Y} = \bar{Y}$+ $Y_k$ ($N_j$,$N_d$)
      $\bar{T} = T$+ $T_k$ ($N_1$,$N_j$)
  end
    $\hat{T} = \bar{Y}$ *Y'($N_j$,$N_d$)*60/$\bar{T}$
 end
    if $f_n$(1,j)+ $f_l$(j, d)≤a then
           a= $f_n$(1,j)+ $f_l$(j, d);
           $J_{n+1}$(i)=j
           $f_{n+1}$(1,i)=a;
        end
     end
   end
end
v=size(N)
for k=1:size(N)-1 do
   $v_k$=J(,v);
end
p=fliplr(p)
```

Figure 3.16 A typical pseudo code for the route finding algorithm based on the travel time modeling

CHAPTER 4

RESULTS AND NUMERICAL EXAMPLE

A simulation process has been designed for the proposed route finding algorithm, which can represent the shortest path and estimated travel time. Let's consider a shortest path problem for our proposed network shown in Figure 4.1, within this network the origin-destination (OD) pair (1,5) is considered and it is assumed that a driver is at point *1* at time u_1 =*11:30:14*.

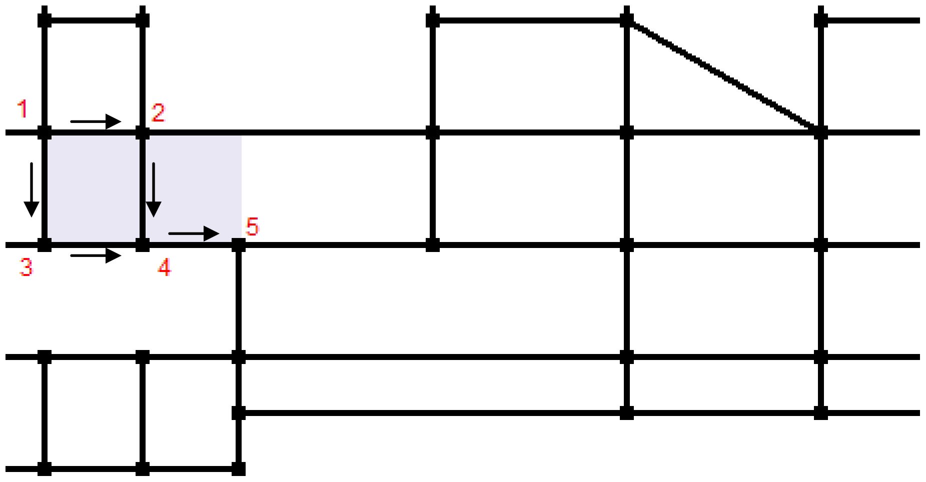


Figure 4.1 Proposed street network

From the network it can be seen that 2 paths exist between nodes 1 and 5 if R = 0; these are listed in Table 4.1. The characteristics of the links are listed in Table 4.2.

Table 4.1 List of paths of OD pair (1,5)

Path	**1**	**2**
Node	1-2-4-5	1-3-4-5

Table 4.2 Network characteristics

Link	**Length, *L* *(meters)***	**Jam Density, *J (Veh/500 meters)***	**Speed limit, S_f**	**Adjustment Factor, λ**
1-2	500	20	35 *mi/hr* ≈16 *m/sec.*	0.9
1-3	500	20	35 *mi/hr* ≈16 *m/sec.*	0.5
2-4	500	20	35 *mi/hr* ≈16 *m/sec.*	0.8
3-4	500	20	35 *mi/hr* ≈16 *m/sec.*	1
4-5	500	20	35 *mi/hr* ≈16 *m/sec.*	1.3

1-step path

First, we need to find a 1-step path from point 1 to every other node $f_1(1, j, 11{:}30{:}14), j = 1,2,3,4,5$ using Equation 3.25:

$f_1(\mathbf{1},\mathbf{1},11{:}30{:}14) = 0$, considering that there is no cost to get from point *1* to itself.

- If a driver selects point *2*, then based on the speedtables created at t_1 = 11:30:20, t_2 = 11:30:40, t_3 = 11:31:00, and t_4 = 11:31:20 (see Table 4.3), $\acute{Y}(1,2,11{:}30{:}14)$ can be estimated as follows:

$a(1,2) = (t_1 - u_1)*S_y(1,2,t_1) = (6)*(13) = 78\ m$, where $a(1,2) = 78 < 500$

$X_2(1,2) = (t_2 - t_1)*Sy(1,2,t_2) = (20)*(9) = 180\ m$, where $a(1,2) + X_2(1,2) = 258 < 500$

$X_3 = (t_3 - t_2)*Sy(1,2,t_3) = (20)*(6) = 120\ m,$ where $a(1,2) + X_2(1,2) + X_3(1,2) = 378 < 500$

$X_4 = (t_4 - t_3) * Sy\,(1,2,t_4) = (20)*(8) = 160\ m$, where

$a(1,2) + X_2(1,2) + X_3(1,2) + X_4(1,2) = 538 > 500$, which renders that $m = 4$ and $b(1,2) = 538 - 500 = 38$.

$$\acute{Y}(1{,}2{,}11{:}30{:}14) = (6) + 20 * (4-1) - \left(\frac{38}{8}\right) = 61.25\ Seconds\ or\ 1.02\ minutes$$

Therefore, by applying the adjustment factor we get:

$f_1(\mathbf{1{,}2},u_1) = \hat{T}(1{,}2,u_1) = \lambda(1{,}2,u_1)\acute{Y}\,(1{,}3{,}11{:}30{:}14) = (0.9)*(61.24) = 55$ sec. If a driver selects point *2*, he/she will arrive at point *2* at 11:31:14 + 00:00:55 =11:31:09.

- If a driver selects point *3*, then based on the speedtables created at $t_1 = 11{:}30{:}20$, $t_2 = 11{:}30{:}40$, $t_3 = 11{:}31{:}00$, and $t_4 = 11{:}31{:}20$ (see Table 4.3), $\acute{Y}(1{,}3{,}11{:}30{:}14)$ is computed as follows:

$a(1,3) = (t_1 - u_1)*S_y\,(1,3,t_1) = (6)*(12) = 72\ m$, where: $a(1,3) = 72 < 500$

$X_2(1,3) = (t_2 - t_1)*S_y\,(1,3,t_2) = (20)*(7) = 140\ m$, where: $a(1,3) + X_2(1,3) = 212 < 500$

$X_3(1,3) = (t_3 - t_2)*S_y\,(1,3,t_3) = (20)*(11) = 220\ m$, where:

$a(1,3) + X_2(1,3) + X_3(1,3) = 432 < 500$

$X_4(1,3) = (t_4 - t_3)*S_y\,(1,3,t_4) = (20)*(11) = 220\ m$, where:

$a(1,3) + X_2(1,3) + X_3(1,3) + X_4(1,3) = 652 > 500$, which renders that $m = 4$ and $b(1,3) = 652 - 500 = 38$

$$\acute{Y}(1{,}3,u_1) = (6) + 20 * (4-1) - \left(\frac{152}{8}\right) = 47\ Seconds\ or\ 0.78\ minutes$$

Therefore, $f_1(1,3,u_1) = \hat{T}(1,3,u_1) = \lambda(1,3,u_1)\acute{Y}(1,3,u_1) = (0.5)*(47) = 23.5\ sec.$ This means that if a driver selects point 3, he/she will arrive there at time 11:30:14 + 00:00:47 = 11:30:38.

- $f_1(1,4) = \hat{T}\ (1,4,u_1) = \infty$, since no direct connection between points *1* and *4*.
- $f_1(1,5) = \hat{T}\ (1,4,\ u_1) = \infty$, since no direct connection between *1* and *5*.

2-step paths

We need to use Equation 3.26 to establish 2-step paths from point 1 to every other nodes based on the 1-step paths. Possible 2-step paths from node *1* to itself are as follows:

$$f_2(\mathbf{1},\mathbf{1},u_1)=min\begin{cases} f_1(\mathbf{1},\mathbf{1},u_1) + \lambda(\mathbf{1},\mathbf{1},u_1) * Y'\ (\mathbf{1},\mathbf{1}, 11{:}30{:}14) = 0+0 = 0 \\ f_1(\mathbf{1},\mathbf{2},\ u_1) + \lambda(2,1,u_1) * Y'\ (\mathbf{2},\mathbf{1}, 11{:}31{:}09) = 55.13 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{3},u_1) + \lambda(3,1,u_1) * Y'\ (\mathbf{3},\mathbf{1}, 11{:}30{:}38) = 23.5 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{4},u_1) + \lambda(4,1,u_1) * Y'\ (\mathbf{4},\mathbf{1},\infty\ ) = \infty + \infty = \infty \\ f_1(\mathbf{1},\mathbf{5},u_1) + \lambda(5,1,u_1) * Y'\ (\mathbf{5},\mathbf{1},\infty) = \infty + \infty = \infty \end{cases}$$

Possible 2-step paths from node *1* to 2 are as follows:

$$f_2(\mathbf{1},\mathbf{2},u_1)= min\begin{cases} f_1(\mathbf{1},\mathbf{1},u_1) + \lambda(1,2,u_1) * Y'_2(\mathbf{1},\mathbf{2}, 11{:}30{:}14) = 0 + 55.13 = 55.13 \\ f_1(\mathbf{1},\mathbf{2}, 1u_1) + \lambda(2,2,u_1) * Y'\ (\mathbf{2},\mathbf{2}, 11{:}31{:}09) = 55.13+0 = 55.13 \\ f_1(\mathbf{1},\mathbf{3},u_1) + \lambda(3,2,u_1) * Y'\ (\mathbf{3},\mathbf{2}, 11{:}30{:}38) = 23.5 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{4},u_1) + \lambda(4,2,u_1) * Y'\ (\mathbf{4},\mathbf{2},\infty) = \infty + \infty = \infty \\ f_1(\mathbf{1},\mathbf{5},u_1) + \lambda(5,2,u_1) * Y'\ (\mathbf{5},\mathbf{2},\infty) = \infty + \infty = \infty \end{cases}$$

- $f_1(\mathbf{1},\mathbf{1},u_1) = 0$, which means that there is no cost to get from point *1* to itself, therefore the driver is still at point *1*: $\lambda(\mathbf{1},\mathbf{2},u_1) * Y'(\mathbf{1},\mathbf{2}, 11{:}30{:}14) = 55.13$ sec.
- The second choice is $f_1(\mathbf{1},\mathbf{2},u_1) + \lambda(\mathbf{2},\mathbf{2},u_1) * Y'(\mathbf{2},\mathbf{2}, 11{:}31{:}09)$, in this case is $f_1(\mathbf{1},\mathbf{2},u_1) = 55.13$ *sec.*, which means that at time u_1 = 11:30:14 the driver is at point *1* and he/she will arrive at point *2* in next 55.13 seconds (i.e., at time 11:31:09), and $\hat{T}(\mathbf{1},\mathbf{2}, 11{:}31{:}09) = \lambda(\mathbf{2},\mathbf{2},u_1) * Y'(\mathbf{2},\mathbf{2}, 11{:}31{:}09) = 0$ because there is no cost associated with getting from point 2 to itself.
- Similarly, in the third choice $f_1(\mathbf{1},\mathbf{3}, u_1) = 23.5$, which means the driver will arrive at point 3 at time *11:30:38,* but $\lambda(\mathbf{3},\mathbf{2},u_1) * Y'(\mathbf{3},\mathbf{2}, 11{:}30{:}38) = \infty$, because there is no direct connection between points *3* and *2*.
- In the fourth and fifth choices node 1 is not directly linked to nodes 4 and 5 and as a result their estimated travel times are also infinity.

Possible 2-step paths from node *1* to node *3* are as follows:

$$f_2(\mathbf{1},\mathbf{3},u_1) = \min \begin{cases} f_1(\mathbf{1},\mathbf{1},u_1) + \lambda(1,3,u_1) * Y'(\mathbf{1},\mathbf{3}, 11{:}30{:}14) = 0 + 23.5 = 23.5 \\ f_1(\mathbf{1},\mathbf{2},u_1) + \lambda(2,3,u_1) * Y'(\mathbf{2},\mathbf{3}, 11{:}31{:}09) = 55.13 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{3},u_1) + \lambda(3,3,u_1) * Y'(\mathbf{3},\mathbf{3}, 11{:}30{:}38) = 23.5 + 0 = 23.5 \\ f_1(\mathbf{1},\mathbf{4},u_1) + \lambda(4,3,u_1) * Y'(4,3,\infty) = \infty + \infty = \infty \\ f_1(\mathbf{1},\mathbf{5},u_1) + \lambda(5,3,u_1) * Y'(5,3,\infty) = \infty + \infty = \infty \end{cases}$$

Possible 2-step paths from node 1 to node 4 are as follows:

$$f_2(\mathbf{1},\mathbf{4},u_1) = \min \begin{cases} f_1(\mathbf{1},\mathbf{1},u_1) + \lambda(1,4,u_1) * Y'(\mathbf{1},\mathbf{4},11{:}30{:}14) = 0 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{2},u_1) + \lambda(2,4,u_1) * Y'(\mathbf{2},\mathbf{4},11{:}31{:}09) = 55.13 + 53 = 108.13 \\ f_1(\mathbf{1},\mathbf{3},u_1) + \lambda(3,4,u_1) * Y'(\mathbf{3},\mathbf{4},11{:}30{:}38) = 23.5 + 60.75 = 84.25 \\ f_1(\mathbf{1},\mathbf{4},u_1) + \lambda(4,4,u_1) * Y'(\mathbf{4},\mathbf{4},\infty) = \infty + 0 = \infty \\ f_1(\mathbf{1},\mathbf{5},u_1) + \lambda(5,4,u_1) * Y'(\mathbf{5},\mathbf{4},\infty) = \infty + \infty = \infty \end{cases}$$

- In the second choice $f_1(1,2,u_1) = 55.13$ sec., which means that the driver will arrive at point *2* at time *11:31:09*. At this instant by leaving point *2* at $u_2 = 11{:}31{:}09$ in order to go to point *4*, it takes $\hat{T}_2(\mathbf{2},\mathbf{4},11{:}31{:}09) = \lambda(1,2,u_1) * Y'(\mathbf{2},\mathbf{4},11{:}31{:}09)$ seconds to arrive at point *4*. First, we need to estimate $Y'(\mathbf{2},\mathbf{4},11{:}31{:}09)$ based on the speedtables created at times $t_4 = 11{:}31{:}20$, $t_5 = 11{:}31{:}40$, $t_6 = 11{:}32{:}00$, $t_7 = 11{:}32{:}20$, and $t_8 = 11{:}32{:}40$. Then the following computations follow:

$a(2,4) = (t_4 - u_2) * S_y(2,4,t_4) = (11)*(9) = 99\ m$, where: $a(2,4) = 99 < 500$

$X_5(2,4) = (t_5 - t_4) * S_y(2,4,t_5) = (20)*(8) = 160\ m$, where: $a(2,4) + X_5(2,4) = 259 < 500$

$X_6(2,4) = (t_6 - t_5) * S_y(2,4,t_6) = (20)*(6) = 120\ m$, where:

$a(2,4) + X_5(2,4) + X_6(2,4) = 379 < 500$

$X_7(2,4) = (t_7 - t_6) * S_y(2,4,t_7) = (20)*(5) = 100\ m$, where:

$a(2,4) + X_5(2,4) + X_6(2,4) + X_7(2,4) = 479 < 500$

$X_8(2,4) = (t_8 - t_7) * S_y(2,4,t_8) = (20)*(11) = 220\ m$, where:

$a(2,4) + X_5(2,4) + X_6(2,4) + X_7(2,4) + X_8(2,4) = 699 > 500$

Which renders that $m = 5$ and $b(2,4) = 699 - 500 = 199$, therefore:

$$\acute{Y}(2,4,u_1) = (11{:}31{:}20 - 11{:}31{:}09) + 20 * (8 - 4) - \left(\frac{199}{8}\right) = 66.13\,sec.$$

$\hat{T}_2(\mathbf{2},\mathbf{4},11{:}31{:}09) = \lambda(1,2,u_1) * Y'\,(\mathbf{2},\mathbf{4},11{:}31{:}09) = (0.8)*(66.13) = 53$

- Similarly, in the third choice $f_1(\mathbf{1},\mathbf{3},u_1) = 23.5$ seconds, which means that if the driver selects point *3* he/she will arrive at point *3* at $u_3 = 11{:}30{:}38$. Therefore, the forecasted travel time from *3* to *4* is $\hat{T}(\mathbf{3},\mathbf{4},11{:}30{:}38) = \lambda(1,3,u_1) * Y'\,(\mathbf{3},\mathbf{4},11{:}30{:}38)$ based on the speedtables created at times $t_2 = 11{:}30{:}40$, $t_3 = 11{:}31{:}00$, $t_4 = 11{:}31{:}20$, and $t_5 = 11{:}31{:}40$. Then the following computations follow:

$a(3,4) = (t_2 - u_3)*S_y(3,4,t_2) = (2)*(5) = 10\ m$, where: $a(3,4) = 10 < 500$

$X_3(3,4) = (t_3 - t_2)*S_y(3,4,t_3) = (20)*(9) = 180\ m$, where: $a(3,4) + X_3(3,4) = 190 < 500$

$X_4(3,4) = (t_4 - t_3)*S_y(3,4,t_4) = (20)*(8) = 160\ m$, where:

$a(3,4) + X_3(3,4) + X_4(3,4) = 350 < 500$

$X_5(3,4) = (t_5 - t_4)*S_y(3,4,t_5) = (20)*(8) = 160\ m$, where:

$a(3,4) + X_3(3,4) + X_4(3,4) + X_5(3,4) = 510 > 500$, therefore, $m = 4$ and $b(3,4) = 10$.

$$\acute{Y}(3,4,u_1) = (11{:}30{:}40 - 11{:}30{:}38) + 20 * (4 - 1) - \left(\frac{10}{8}\right) = 60.75\,Sec.$$

$\hat{T}_2(3,4,u_1) = \lambda(3,4,u_1)*Y_2(2,4,11{:}30{:}38) = (1)*(60.75) = 60.75$

Possible 2-steps paths from 1 to 5 are as follows (Note: no direct link from *1* to *5*):

$$f_2(\mathbf{1,5},u_1) = min \begin{cases} f_1(\mathbf{1},\mathbf{1},u_1) + \lambda(1,5,u_1) * Y'(\mathbf{1},\mathbf{5},11{:}30{:}14) = 0 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{2},u_1) + \lambda(2,5,u_1) * Y'(\mathbf{2},\mathbf{5},11{:}31{:}09) = 55.13 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{3},u_1) + \lambda(3,5,u_1) * Y'(\mathbf{3},\mathbf{5},11{:}30{:}38) = 23.5 + \infty = \infty \\ f_1(\mathbf{1},\mathbf{4},u_1) + \lambda(4,5,u_1) * Y'(\mathbf{4},\mathbf{5},\infty) = \infty \\ f_1(\mathbf{1},\mathbf{5},u_1) + \lambda(5,5,u_1) * Y'(\mathbf{5},\mathbf{5},\infty) = \infty + 0 = \infty \end{cases}$$

3-step paths

Here we need to establish 3-step paths from node 1 to every other nodes based on the 2-step paths previously created. Possible 3-step paths from node *1* to itself are as follows:

$$f_3(\mathbf{1,1},u_1) = min \begin{cases} f_2(\mathbf{1},\mathbf{1},u_1) + \lambda(1,1,u_1) * Y'(\mathbf{1},\mathbf{1},11{:}30{:}14) = 0+0 = 0 \\ f_2(\mathbf{1},\mathbf{2},u_1) + \lambda(2,1,u_1) * Y'(\mathbf{2},\mathbf{1},11{:}31{:}09) = 55.13 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{3},u_1) + \lambda(3,1,u_1) * Y'(\mathbf{3},\mathbf{1},11{:}30{:}38) = 23.5 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{4},u_1) + \lambda(4,1,u_1) * Y'(\mathbf{4},\mathbf{1},\infty) = 84.25 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{5},u_1) + \lambda(5,1,u_1) * Y'(\mathbf{5},\mathbf{1},\infty) = \infty + \infty = \infty \end{cases}$$

Possible 3-step paths from node 1 to node 2 are as follows:

$$f_3(\mathbf{1,2},u_1) = min \begin{cases} f_2(\mathbf{1},\mathbf{1},u_1) + \lambda(1,2,u_1) * Y'(\mathbf{1},\mathbf{2},11{:}30{:}14) = 0+55.13 = 55.13 \\ f_2(\mathbf{1},\mathbf{2},u_1) + \lambda(2,2,u_1) * Y'(\mathbf{2},\mathbf{2},11{:}31{:}09) = 55.13 + 0 = 5\ 5.13 \\ f_2(\mathbf{1},\mathbf{3},u_1) + \lambda(3,2,u_1) * Y'(\mathbf{3},\mathbf{2},11{:}30{:}38) = 23.5 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{4},u_1) + \lambda(4,2,u_1) * Y'(\mathbf{4},\mathbf{2},\infty) = 84.25 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{5},u_1) + \lambda(5,2,u_1) * Y'(\mathbf{5},\mathbf{2},\infty) = \infty + \infty = \infty \end{cases}$$

Possible 3-step paths from node 1 to node 3 are as follows:

$$f_3(\mathbf{1},\mathbf{3},u_1)=min \begin{cases} f_2(\mathbf{1},\mathbf{1},u_1) + \lambda(1,3,u_1) * Y'(\mathbf{1},\mathbf{3},11:30:14) = 0 + 23.5 = 23.5 \\ f_2(\mathbf{1},\mathbf{2},u_1) + \lambda(2,3,u_1) * Y'(\mathbf{2},\mathbf{3},11:31:09) = 55.13 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{3},u_1) + \lambda(3,3,u_1) * Y'(\mathbf{3},\mathbf{3},11:30:38) = 23.5 + 0 = 23.5 \\ f_2(\mathbf{1},\mathbf{4},u_1) + \lambda(4,3,u_1) * Y'(\mathbf{4},\mathbf{3},\infty) = 84.25 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{5},u_1) + \lambda(5,3,u_1) * Y'(\mathbf{5},\mathbf{3},\infty) = \infty + \infty = \infty \end{cases}$$

Possible 3-steps paths from *1* to *4* are as follows:

$$f_3(\mathbf{1},\mathbf{4},u_1)=min \begin{cases} f_2(\mathbf{1},\mathbf{1},u_1) + \lambda(1,4,u_1) * Y'(\mathbf{1},\mathbf{4},11:30:14) = 0 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{2},u_1) + \lambda(2,4,u_1) * Y'(\mathbf{2},\mathbf{4},11:31:09) = 55.13 + 53 = 108.13 \\ f_2(\mathbf{1},\mathbf{3},u_1) + \lambda(3,4,u_1) * Y'(\mathbf{3},\mathbf{4},11:30:38) = 23.5 + 60.75 = 84.25 \\ f_2(\mathbf{1},\mathbf{4},u_1) + \lambda(4,4,u_1) * Y'(\mathbf{4},\mathbf{4},\infty) = \infty \\ f_2(\mathbf{1},\mathbf{5},u_1) + \lambda(5,4,u_1) * Y'(\mathbf{5},\mathbf{4},\infty) = \infty + 0 = \infty \end{cases}$$

Possible 3-step paths from node 1 to node 5 are as follows:

$$f_3(\mathbf{1},\mathbf{5},u_1)=min \begin{cases} f_2(\mathbf{1},\mathbf{1},u_1) + \lambda(1,5,u_1) * Y'(\mathbf{1},\mathbf{5},11:30:14) = 0 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{2},u_1) + \lambda(2,5,u_1) * Y'(\mathbf{2},\mathbf{5},11:31:09) = 55.13 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{3},u_1) + \lambda(3,5,u_1) * Y'(\mathbf{3},\mathbf{5},11:30:38) = 23.5 + \infty = \infty \\ f_2(\mathbf{1},\mathbf{4},u_1) + \lambda(4,5,u_1) * Y'(\mathbf{4},\mathbf{5},11:31:38) = 84.25 + 69 = 153.25 \\ f_2(\mathbf{1},\mathbf{5},u_1) + \lambda(5,5,u_1) * Y'(\mathbf{5},\mathbf{5},\infty) = \infty + 0 = \infty \end{cases}$$

- In the fourth choice $f_2(1,4,u_1) = 84.25$ indicates that the driver will arrive at point *4* at time u_4*=11:31:38*. Then the estimated travel time from node *4* to node *5* is:

$\hat{T}(\mathbf{4},\mathbf{5}, 11{:}31{:}38)$ $=\lambda(1,3, u_1)* Y'$ *(4,5, 11:31:38)* based on the speedtables created at times t_5*= 11:31:40,* t_6 *= 11:32:00,* t_7*= 11:32:20,* t_8*= 11:32:40,* and t_9*= 11:33:00*

$a(4,5) = (t_5 - u_4)*S_y(4,5,t_5)$ = (2)*(8) = 16 m, where: $a(4,5) = 16 < 500$

$X_6(4,5) = (t_6 - t_5)*S_y(4,5,t_6)$ = (20)*(7) = 140 m, where: $a(4,5) + X_6(4,5) = 156 < 500$

$X_7(4,5) = (t_7 - t_6)*S_y(4,5,t_7)$ = (20)*(6) = 120 m, where:

$a(4,5) + X_6(4,5) + X_7(4,5) = 276 < 500$

$X_8(4,5) = (t_8 - t_7)*S_y(4,5,t_8)$ = (20)*(7) = 140 m, where:

$a(4,5) + X_6(4,5) + X_7(4,5) + X_8(4,5) = 416 < 500$

$X_9(4,5) = (t_9 - t_8)*S_y(4,5,t_7)$ = (20)*(7) = 140 m, where:

$X_1\ a(4,5) + X_6(4,5) + X_7(4,5) + X_8(4,5) + X_9(4,5) = 556 > 500$

Which means that $m = 5$ and $b(4,5) = 556 – 500 = 56$

$$\acute{Y}(4,5,u_1) = (11{:}31{:}40 - 11{:}31{:}38) + 20 * (9 - 5) - \left(\frac{56}{7}\right) = 74\ sec.$$

$\hat{T}_3(4,5,u_1) = \lambda(4,5,u_1)*Y_3(4,511{:}31{:}38,) = (1.3)*(53)=69\ sec.$

4-step paths

All possible 4-step paths from node 1 to node 1 are as follows:

$$f_4(\mathbf{1},\mathbf{1},u_1)=\min\begin{cases} f_3(\mathbf{1},\mathbf{1},u_1)\ +\ \lambda(1,1,u_1)*Y'\ (\mathbf{1},\mathbf{1},11:31:38)=0+0=0 \\ f_3(\mathbf{1},\mathbf{2},u_1)\ +\lambda(2,1,u_1)*Y'\ (\mathbf{2},\mathbf{1},11:31:09)=55.13+\infty=\infty \\ f_3(\mathbf{1},\mathbf{3},u_1)\ +\ \lambda(3,1,u_1)*Y'\ (\mathbf{3},\mathbf{1}:11:30:38)=23.5+\infty=\infty \\ f_3(\mathbf{1},\mathbf{4},u_1)\ +\ \lambda(4,1,u_1)*Y'\ (\mathbf{4},\mathbf{1},11:31:38)=84.25+\infty=\infty \\ f_3(\mathbf{1},\mathbf{5},u_1)\ +\ \lambda(5,1,u_1)*Y'\ (\mathbf{5},\mathbf{1},11:31:47)=153.25+\infty \end{cases}$$

All possible 4-step paths from node 1 to node 2 are as follows:

$$f_4(\mathbf{1},\mathbf{2},u_1)=\min\begin{cases} f_3(\mathbf{1},\mathbf{1},u_1)\ +\ \lambda(1,2,u_1)*Y'_4(\mathbf{1},\mathbf{2},11:30:14)=0+55.13=55.13 \\ f_3(\mathbf{1},\mathbf{2},u_1)\ +\lambda(2,2,u_1)*Y'_4(\mathbf{2},\mathbf{2},11:31:09)=55.13+0=55.13 \\ f_3(\mathbf{1},\mathbf{3},u_1)\ +\ \lambda(3,2,u_1)*Y'_4(\mathbf{3},\mathbf{2},11:30:38)=23.5+\infty=\infty \\ f_3(\mathbf{1},\mathbf{4},u_1)\ +\ \lambda(4,2,u_1)*Y'_4(\mathbf{4},\mathbf{2},11:31:38)=84.25+\infty=\infty \\ f_3(\mathbf{1},\mathbf{5},u_1)\ +\ \lambda(5,2,u_1)*Y'_4(\mathbf{5},\mathbf{2},11:31:47)=153.25+\infty=\infty \end{cases}$$

All possible 4-step paths from node 1 to node 3 are as follows:

$$f_4(\mathbf{1},\mathbf{3},u_1)=\min\begin{cases} f_3(\mathbf{1},\mathbf{1},u_1)\ +\ \lambda(1,3,u_1)*Y'\ (\mathbf{1},\mathbf{3},11:30:14)=0+23.5=23.5 \\ f_3(\mathbf{1},\mathbf{2},u_1)\ +\lambda(2,3,u_1)*Y'\ (\mathbf{2},\mathbf{3},11:31:09)=55.13+\infty=\infty \\ f_3(\mathbf{1},\mathbf{3},u_1)\ +\ \lambda(3,3,u_1)*Y'\ (\mathbf{3},\mathbf{3},11:30:38)=23.5+0=23.5 \\ f_3(\mathbf{1},\mathbf{4},u_1)\ +\ \lambda(4,3,u_1)*Y'\ (\mathbf{4},\mathbf{3},11:31:38)=84.25+\infty=\infty \\ f_3(\mathbf{1},\mathbf{5},u_1)\ +\ \lambda(5,3,u_1)*Y'\ (\mathbf{5},\mathbf{3},11:31:47)=153.25+\infty=\infty \end{cases}$$

All possible 4-step paths from node 1 to node 4 are as follows:

$$f_4(\mathbf{1},\mathbf{4},u_1)=min \begin{cases} f_3(\mathbf{1},\mathbf{1},u_1)\ +\ \lambda(1{,}4,u_1)*Y'\,(\mathbf{1},\mathbf{4},11{:}30{:}14) = 0+\infty\ =\infty \\ f_3(\mathbf{1},\mathbf{2},u_1)\ +\lambda(2{,}4,u_1)*Y'\,(\mathbf{2},\mathbf{4},11{:}31{:}09) = 55.13+53 = 108.13 \\ f_3(\mathbf{1},\mathbf{3},u_1)+\ \lambda(3{,}4,u_1)*Y'\,(\mathbf{3},\mathbf{4},11{:}30{:}38) = 23.5+60.75 = 84.25 \\ f_3(\mathbf{1},\mathbf{4},u_1)\ +\ \lambda(4{,}4,u_1)*Y'\,(\mathbf{4},\mathbf{4},11{:}31{:}38) = 84.25+0 = 84.25 \\ f_3(\mathbf{1},\mathbf{5},u_1)\ +\ \lambda(5{,}4,u_1)*Y'\,(\mathbf{5},\mathbf{4},11{:}31{:}47) = \infty+\infty = \infty \end{cases}$$

All possible 4-step paths from node 1 to node 5 are as follows:

$$f_4(\mathbf{1},\mathbf{5},u_1)=min \begin{cases} f_3(\mathbf{1},\mathbf{1},u_1)\ +\ \lambda(1{,}5,u_1)*Y'\,(\mathbf{1},\mathbf{5},11{:}30{:}14) = 0+\infty=\infty \\ f_3(\mathbf{1},\mathbf{2},u_1)\ +\lambda(2{,}5,u_1)*Y'\,(\mathbf{2},\mathbf{5},11{:}31{:}09) = 55.13+\infty = \infty \\ f_3(\mathbf{1},\mathbf{3},u_1)\ +\ \lambda(3{,}5,u_1)*Y'\,(\mathbf{3},\mathbf{5},11{:}30{:}38) = 23.5+\infty = \infty \\ f_3(\mathbf{1},\mathbf{4},u_1)\ +\ \lambda(4{,}5,u_1)*Y'\,(\mathbf{4},\mathbf{5},11{:}31{:}38) = 84.25+69 = 153.25 \\ f_3(\mathbf{1},\mathbf{5},u_1)\ +\ \lambda(5{,}5,u_1)*Y'\,(\mathbf{5},\mathbf{5},11{:}31{:}47) = 153.25+0 = \infty \end{cases}$$

As a final point, the shortest path requested at time u_1 = 11:30:14 can be found by using the backward approach. From the last solution, $f_4(\mathbf{1},\mathbf{5},u_1)$, we can find the last segment of the shortest path, street segment (4,5), then $f_3(\mathbf{1},\mathbf{4},u_1)$ indicates that the next segment of the shortest path is segment (3,4), and finally, from $f_2(\mathbf{1},\mathbf{3},u_1)$, we can understand that the first segment of the shortest path is segment (1,3). Therefore, the shortest path consists of nodes 1, 3, 4, and 5 as shown in Figure 4.2.

$$f_3(\mathbf{1},\mathbf{4},u_1) + \hat{T}(\mathbf{4},\mathbf{5},11:31:38)$$

$$\downarrow$$

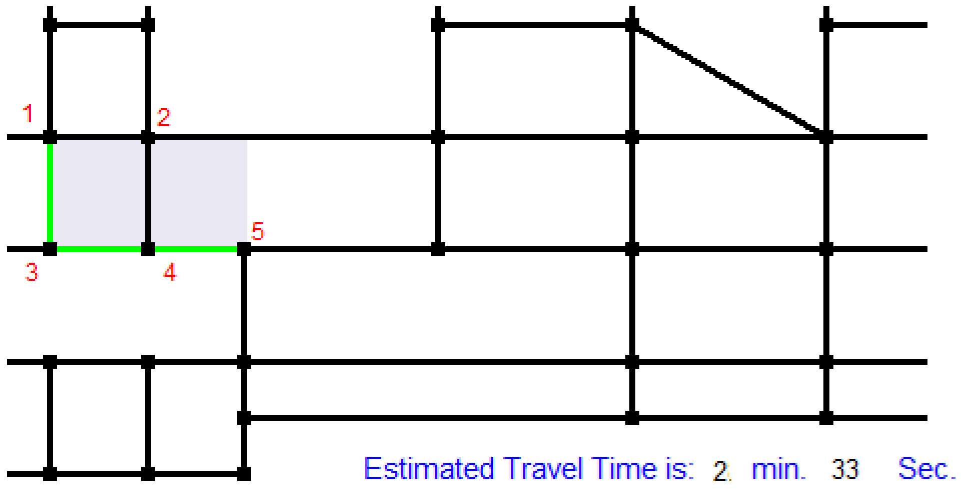
$$\overbrace{f_2(\mathbf{1},\mathbf{3},u_1) + \hat{T}(\mathbf{3},\mathbf{4},11:30:38)}$$

$$\downarrow$$

$$\overbrace{f_1(\mathbf{1},\mathbf{1},u_1) + \hat{T}(\mathbf{1},\mathbf{3},11:30:14)}$$

$$\downarrow$$

$$\overbrace{\hat{T}(\mathbf{1},\mathbf{1},11:30:14)}$$



Estimated Travel Time is: 2. min. 33 Sec.

Figure 4.2 Detected shortest path

Table 4.3 Observed data at different times

t_1 = 11:30:00

Observed Density at 11:30:00

Node	1	2	3	4	5
1	-	5	3	-	-
2	-	-	-	6	-
3	-	-	-	4	-
4	-	-	-	-	7
5	-	-	-	-	-

Speedtable created at 11:30:00

Node	1	2	3	4	5
1	∞	11	13	0	0
2	0	∞	0	11	0
3	0	0	∞	12	0
4	0	0	0	∞	10
5	0	0	0	0	∞

Timetable created at 11:30:00

Node	1	2	3	4	5
1	0	0.74	0.65	∞	∞
2	∞	0	∞	0.79	∞
3	∞	∞	0	0.69	∞
4	∞	∞	∞	0	0.85
5	∞	∞	∞	∞	0

t_2 = 11:30:20

Observed Density at 11:30:20

Node	1	2	3	4	5
1	-	3	4	-	-
2	-	-	-	2	-
3	-	-	-	7	-
4	-	-	-	-	5
5	-	-	-	-	-

Speedtable created at 11:30:20

Node	1	2	3	4	5
1	∞	13	12	0	0
2	0	∞	0	14	0
3	0	0	∞	10	0
4	0	0	0	∞	11
5	0	0	0	0	∞

Timetable created at 11:30:20

Node	1	2	3	4	5
1	0	0.65	0.69	∞	∞
2	∞	0	∞	0.62	∞
3	∞	∞	0	0.85	∞
4	∞	∞	∞	0	0.74
5	∞	∞	∞	∞	0

t_3 = 11:30:40

Observed Density at 11:30:40

Node	1	2	3	4	5
1	-	8	11	-	-
2	-	-	-	6	-
3	-	-	-	13	-
4	-	-	-	-	5
5	-	-	-	-	-

Speedtable created at 11:30:40

Node	1	2	3	4	5
1	∞	9	7	0	0
2	0	∞	0	11	0
3	0	0	∞	5	0
4	0	0	0	∞	11
5	0	0	0	0	∞

Timetable created at 11:30:40

Node	1	2	3	4	5
1	0	0.93	1.23	∞	∞
2	∞	0	∞	0.79	∞
3	∞	∞	0	1.59	∞
4	∞	∞	∞	0	0.74
5	∞	∞	∞	∞	0

t_4 = 11:31:00

Observed Density at 11:31:00

Node	1	2	3	4	5
1	-	12	5	-	-
2	-	-	-	6	-
3	-	-	-	8	-
4	-	-	-	-	7
5	-	-	-	-	-

Speedtable created at 11:31:00

Node	1	2	3	4	5
1	∞	6	11	0	0
2	0	∞	0	11	0
3	0	0	∞	9	0
4	0	0	0	∞	10
5	0	0	0	0	∞

Timetable created at 11:31:00

Node	1	2	3	4	5
1	0	1.39	0.74	∞	∞
2	∞	0	∞	0.79	∞
3	∞	∞	0	0.93	∞
4	∞	∞	∞	0	0.85
5	∞	∞	∞	∞	0

t_5 = 11:31:20

Observed Density at 11:31:20

Node	1	2	3	4	5
1	-	10	6	-	-
2	-	-	-	8	-
3	-	-	-	9	-
4	-	-	-	-	8
5	-	-	-	-	-

Speedtable created at 11:31:20

Node	1	2	3	4	5
1	∞	8	11	0	0
2	0	∞	0	9	0
3	0	0	∞	8	0
4	0	0	0	∞	9
5	0	0	0	0	∞

Timetable created at 11:31:20

Node	1	2	3	4	5
1	0	1.11	0.79	∞	∞
2	∞	0	∞	0.93	∞
3	∞	∞	0	1.01	∞
4	∞	∞	∞	0	0.93
5	∞	∞	∞	∞	0

t_6 = 11:31:40

Observed Density at 11:31:40

Node	1	2	3	4	5
1	-	11	7	-	-
2	-	-	-	9	-
3	-	-	-	9	-
4	-	-	-	-	9
5	-	-	-	-	-

Speedtable created at 11:31:40

Node	1	2	3	4	5
1	∞	7	10	0	0
2	0	∞	0	8	0
3	0	0	∞	8	0
4	0	0	0	∞	8
5	0	0	0	0	∞

Timetable created at 11:31:40

Node	1	2	3	4	5
1	0	1.23	0.85	∞	∞
2	∞	0	∞	1.01	∞
3	∞	∞	0	1.01	∞
4	∞	∞	∞	0	1.01
5	∞	∞	∞	∞	0

t_7 = 11:32:00

Observed Density at 11:32:00

Node	1	2	3	4	5
1	-	13	5	-	-
2	-	-	-	12	-
3	-	-	-	10	-
4	-	-	-	-	11
5	-	-	-	-	-

Speedtable created at 11:32:00

Node	1	2	3	4	5
1	∞	5	11	0	0
2	0	∞	0	6	0
3	0	0	∞	8	0
4	0	0	0	∞	7
5	0	0	0	0	∞

Timetable created at 11:32:00

Node	1	2	3	4	5
1	0	1.59	0.74	∞	∞
2	∞	0	∞	1.39	∞
3	∞	∞	0	1.11	∞
4	∞	∞	∞	0	1.23
5	∞	∞	∞	∞	0

t_8 = 11:32:20

Observed Density at 11:32:20

Node	1	2	3	4	5
1	-	15	5	-	-
2	-	-	-	13	-
3	-	-	-	11	-
4	-	-	-	-	12
5	-	-	-	-	-

Speedtable created at 11:32:20

Node	1	2	3	4	5
1	∞	4	11	0	0
2	0	∞	0	5	0
3	0	0	∞	7	0
4	0	0	0	∞	6
5	0	0	0	0	∞

Timetable created at 11:32:20

Node	1	2	3	4	5
1	0	2.22	0.74	∞	∞
2	∞	0	∞	1.59	∞
3	∞	∞	0	1.23	∞
4	∞	∞	∞	0	1.39
5	∞	∞	∞	∞	0

t_9 = 11:32:40

Observed Density at 11:32:40

Node	1	2	3	4	5
1	-	12	5	-	-
2	-	-	-	6	-
3	-	-	-	5	-
4	-	-	-	-	11
5	-	-	-	-	-

Speedtable created at 11:32:40

Node	1	2	3	4	5
1	∞	6	11	0	0
2	0	∞	0	11	0
3	0	0	∞	11	0
4	0	0	0	∞	7
5	0	0	0	0	∞

Timetable created at 11:32:40

Node	1	2	3	4	5
1	0	1.39	0.74	∞	∞
2	∞	0	∞	0.79	∞
3	∞	∞	0	0.74	∞
4	∞	∞	∞	0	1.23
5	∞	∞	∞	∞	0

t_{10} = 11:33:00

Observed Density at 11:33:00

Node	1	2	3	4	5
1	-	13	5	-	-
2	-	-	-	12	-
3	-	-	-	10	-
4	-	-	-	-	11
5	-	-	-	-	-

Speedtable created at 11:33:00

Node	1	2	3	4	5
1	∞	5	11	0	0
2	0	∞	0	6	0
3	0	0	∞	8	0
4	0	0	0	∞	7
5	0	0	0	0	∞

Timetable created at 11:33:00

Node	1	2	3	4	5
1	0	1.59	0.74	∞	∞
2	∞	0	∞	1.39	∞
3	∞	∞	0	1.11	∞
4	∞	∞	∞	0	1.23
5	∞	∞	∞	∞	0

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This thesis presented a new method for dynamic vehicle routing in order to find the shortest path using real time data and historical data collected from traffic detectors installed in all street segments. The raw data provided by these detectors is transferred to the control unit, which converts them into traffic mean speeds and needed travel time to traverse each street segment. For this purpose speedtables and timetables are used to store traffic mean speeds and traffic mean travel times during each time slot. Furthermore, the position of each node is defined in a local and Cartesian coordinate system, which allows the zone identification and node selection algorithm to look for the desired nodes between the source and destination points which reduces the data size and speed up the process of finding the shortest path or extend the area that encompasses the source and destination points to take into account more nodes.

In this study the solving strategy is based on the dynamic programming, which takes care of the speedtable and timetable updating system to find the shortest path among possible alternatives, by considering each node toward the destination only once.

In this study two different models have been developed based on different decision criteria:

- Finding the shortest path based on the real time data collected from the street network.
- Finding the shortest path using travel time modeling method based on historical and real time data that incorporates both concepts of short-term travel time forecasting and shortest path finding. Therefore, this research effort opens many interesting and practical issues for future work.

Finally, such algorithms can be implemented in GIS systems such as Google map in order to help commercial sectors move goods and services quickly or to better and efficiently link health and human service providers with users/customers.

5.2 Recommendations

This study opens many practical issues for future works. In this section we provide some recommendations that can improve the existing algorithm in order to be implemented in the real world.

Signal timing: In this study we assumed that waiting at intersections is very minimal or negligible while in real life this is often not the case. Furthermore, for modern traffic activated signals, the detection of vehicles in the approach lanes affects the sequence and duration of green phases provided at the intersection and hence the right-of-way provision. Therefore, it would be better to state or model an appropriate expected waiting time or delay at each intersection in the network.

Road hierarchy: The number of intersections within a street network may be a limiting factor for route finding algorithms. For this purpose, we employed a heuristic method that can limit the number of nodes and consequently increasing the algorithm speed. Another way to improve algorithm runtime is to take into account the road hierarchy, where highways connect multiple large regions; interstate roads connect locations within a region and small roads reach into individual houses. As a result, a vehicle will first be routed via major roads to get access to the region in which its destination point is located and after that it will be routed via minor roads.

Implementation: The last challenge will be implementing the algorithms in a software, hardware and wireless communication system integrated with traffic detectors, control units and monitoring instrumentations in vehicles that work together and share real time information and subsequently display and report the route to be taken to the users (drivers).

REFERENCES

Ahn, W.C., Ramakrishna, R.S., 2002. “Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations.” In *IEEE Transactions on Evolutionary Computation*. 6(6), pp. 566-579.

Bellman, R.E., 1957. “*Dynamic Programming.*” Princeton University Press, Princeton, NJ.

Bertsekas, D.P., Tsitsiklis, J.J., 1991. “An Analysis of Stochastic Shortest Path Problems.” In *Mathematics of Operations Research*. 16(3), pp. 580-595.

Bertsimas D.J. and Van Ryzin G., 1991. “A Stochastic and Dynamic Vehicle Routing Problem in the Euclidean Plan.” In *Operations Research*, 39(4), pp. 601-615.

Bonet, B., Geffner, H., 2002. “Solving Stochastic Shortest-path Problems with Real Time Dynamic Programming.” Online publication accessed on October 2011 at http://ldc.usb.ve/ ~bonet/reports/rtdp.pdf.

Chinneck, W., 2010. “*Practical Optimization: A Gentle Introduction.*” Carleton University, Ottawa, Canada.

Chitra, C., Subbaraj. C., 2010. “A Nondominated Sorting Genetic Algorithm for Shortest Path Routing Problem.” In *International Journal of Electrical and Computer Engineering*, 5(1), pp. 53-63.

Dear, M., Fulton, W., Wolch, J., 2001. “*Sprawl Hits the Wall.* ” Southern California Studies Center, University of Southern California, Los Angeles, CA.

Dean, B.C., 2004. “Algorithms for Minimum-Cost Paths in Time-Dependent Networks with Waiting Policies.” In *Networks*. 44(1), pp. 41-46.

Ding, B., Yu, J.Y., Qin, L., 2008. “Finding Time-dependent Shortest Paths over Large Graphs.” In *Proceedings of The 11th International Conference & Extending Database Technology: Advances In Database Technology*, Nantes, France, March 25-30.

Effati, S., Jafarzade, M., 2007. “Nonlinear Neural Networks for Solving the Shortest Path Problem.” In *Journal of Applied Mathematics and Computation*, 189(1), 567-574.

Frank, H., 2009. “Shortest Paths in Probabilistic Graphs.” In *Operations Research*, 17(4), pp. 583-599.

Gendreau, M., Guertin, F., Potvin, J.Y., Seguin, R., 2006. “Neighborhood Search Heuristics for Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries.” In *Transportation Research Part C*, *Emerging Technology*, 14(3), pp. 157–174.

Gonzales, H., Han, J., Li, X., Myslinska, M., Sondag, J.P., 2007. “Adaptive Fastest Path Computation on a Road Network: A Traffic Mining Approach.” In *Proceedings of the 33rd International Conference on Very Large Data Bases*. pp. 794-805, Vienna, Austria.

Rodrigue, J., Comtois, C., Slack, B., 2009. "*The Geography of Transport Systems*." Routledge, New York, NY.

Ji, X., 2005. "Models and Algorithm for Stochastic Shortest Path Problem." In *Applied Mathematics and Computation*, 170(1), pp. 503-514.

Kanoulas, E., Du, Y., Xia, T., Zhang, D., 2006. "Finding Fastest Paths on A Road Network with Speed Patterns." In *Proceedings of the 22nd International Conference on Data Engineering*, Atlanta, GA.

Lees, B., 2008. "Below the Line." In *ITS International*, May-June 2008. Online publication accessed on October 2011 at http://www.itsinternational.com/Features/article.cfm?recordID=3216.

Loui, R.P., 1983. "Optimal Paths in Graphs with Stochastic or Multidimensional Weights." In *Communications of the ACM*, 26(9), pp. 670-676.

Margiotta, R.A., Spiller, N., 2009. "*Recurring Traffic Bottlenecks: A Primer Focus on Low-Cost Operational Improvements*." Federal Highway Administration, US Department of Transportation, Washington, DC.

Martin, H., 2001. "*No Idle Boast: L.A. Traffic Worst Again.*" Los Angeles Times, Los Angeles, CA. Online publication accessed on October 2011 at http://articles.latimes.com/2002/jun/21/local/me-traffic21.

Mimbela L.E.Y., Klein, L.A., Kent, P., 2000. "*A Summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Systems*." Federal Highway Administration, US Department of Transportation, Washington, DC.

Norman, J.M., 1975. "*Elementary Dynamic Programming.*" Crane, Russak & Company, Inc. New York, NY.

Orda, A., Rom, R., 1990. "Shortest-Path and Minimum Delay Algorithms in Networks with Time Dependent Edge Length." In *Journal of ACM,* 37(3), pp. 607-625.

Orda, A., Rom, R., Sidi, M., 1993. "Minimum Delay Routing in Stochastic Networks." In *IEEE/ACM Transactions on Networking*, 1(2), pp.187-198.

Peeta, S., Sharma, S., Hsu, Y., 2011. "*Dynamic Real-Time Routing for Evacuation Response Planning and Execution.*" Transportation Research Program, Purdue University, West Lafayette, IN.

Potvin J.Y., Xu Y., Benyahia I., 2006. "Vehicle Routing and Scheduling with Dynamic Travel Times." In *Computers & Operations Research,* 33(4), 1129-1137.

Rinker, R., 2002. "*Measuring the Efficiency of the Algorithms*." University of Idaho, Moscow, ID.

Rubin, T.A., Cox, W., 2001. "*The Road Ahead: Innovations for Better Transportation in Texas.*" Texas Public Policy Foundation, Austin, TX.

Smyth, G.K., 1998. "In *Optimization and Nonlinear Equations*." John Wiley & Sons, Ltd, Chichester, UK.

Taniguchi E., Shimamoto H., 2004. "Intelligent Transportation System Based Dynamic Vehicle Routing and Scheduling with Variable Travel Times." In *Transportation Research Part C*, *Emerging* Technology, 12(3-4), pp. 235-250.

Turner, S.M., Eisele, W.L., Benz, R.J., Holder, D.J., 1998. "*Travel Time Data Collection Handbook.*" Federal Highway Administration, US Department of Transportation, Washington, DC.

Ohio Department of Transportation, 2009. "*Ohio's 21st Century Transportation Priorities Task Force.*" Final Report, Ohio Department of Transportation, Columbus, OH.

U.S. Census Bureau, 2001. "*Profile of General Demographic Characteristics: 2000 Census of Population and Housing*" Technical Documentation. US Department of Commerce, Washington DC.

Wasserman, J., 2002. "*2020 Traffic Report: Growth Means More Time Behind the Wheel for Everyone.*" Associated Press. Online publication accessed on October 2011 at http://www.fairus.org/site/PageServer?pagename=iic_immigrationissuecenters64c1.

Yen, J.Y., 1975. "*Shortest Path Network Problems.*" Verlag Anton Hain, Konigstein Germany.

APPENDIX

MATLAB Source Code

```
% Classification method and Route finding Algorithm, Amin Azimian
% Classification method:

plot([200 400 500 700 900 1100 1300 1500 1700 1900], [2100 2100 2100
2100 2100 2100 2100 2100 2100 2100],'k-s',...
[200 300 400 500 700 900 1100 1300 1500 1600 1700 1900 2100],[1900 1900
1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900],'k-s',...
[900 1100 1300], [1800 1800 1800], 'k-s',...
[200 400 600 800 900 1100 1300 1500], [1700 1700 1700 1700 1700 1700
1700 1700], 'k-s',...
[1900 2100], [1700 1700], 'k-s', ...
[1600 1800 1900], [1600 1600 1600],'k-s', ...
[200 400], [1500 1500],'k-s', ...
[600 700 900 1000 1100 1300 1500 1700 1900], [1500 1500 1500 1500 1500
1500 1500 1500 1500], 'k-s',...
[400 600 700 900 1100],[1300 1300 1300 1300 1300],'k',...
[1500 1700 1900 2000 2100], [1300 1300 1300 1300 1300],'k',...
[ 200 400 500 600 900 1100 1300 1500 1700 1900 2100], [1100 1100 1100
1100 1100 1100 1100 1100 1100 1100 1100], 'k-s',...
[500 600], [900 900],'k-s',...
[900 1100], [900 900], 'k-s',...
[1300 1500 1700], [900 900 900],'k-s',...
[1900 2000 2100], [900 900 900], 'k-s',...
[200 400 500 600 900 1100 1300 1500 1700 1900 2000 2100], [700 700 700
700 700 700 700 700 700 700 700 700], 'k-s',...
[1500 1700], [600 600], 'k-s',...
[200 400 500 600 700 900 1100 1300 1500], [500 500 500 500 500 500 500
500 500], 'k-s',...
[1700 1900 2000 2100], [500 500 500 500], 'k-s',...
[200 400 500 600 700 1100 1300 1500 1700 1900 2000 2100], [300 300 300
300 300 300 300 300 300 300 300 300], 'k-s',...
[700 1100 1300 1500 1700], [200 200 200 200 200], 'k-s',...
[400 500 600 700], [100 100 100 100], 'k-s',...
[200 200],[2100 1900],'k-s',...
[200 200],[1700 1500], 'k-s',...
[200 200 200],[1100 900 700],'k-s',...
[200 200],[500 300], 'k-s',...
[400 400],[2100 1900],'k-s',...
[300 400 400 400 400],[1900 1700 1500 1300 1100],'k-s',...
[400 400 400 400],[700 500 300 100],'k-s',...
[500 500],[2100 1900],'k-s',...
[500 500 500 500],[1100 900 700 500],'k-s',...
[500 500],[300 100],'k-s',...
[700 700 600 600 600],[2100 1900 1700 1500 1300],'k-s',...
```

```
[600 600 600 600],[1100 900 700 500],'k-s',...
[600 600],[300 100],'k-s',...
[700 700],[1500 1300],'k-s',...
[700 700 700 700],[500 300 200 100],'k-s',...
[700 800 900 1000],[1900 1700 1600 1500],'k-s',...
[900 900 900 900],[2100 1900 1800 1700],'k-s',...
[900 900 900],[1600 1500 1300], 'k-s',...
[900 900 900 900],[1100 900 700 500],'k-s',...
[1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100], [2100
1900 1800 1700 1500 1300 1100 900 700 500 300 200],'k-s',...
[1100 1300], [900 700],'k-s',...
[1300 1300 1300 1300 1300],[2100 1900 1800 1700 1500],'k-s',...
[1300 1300 1300 1300 1300 1300], [1100 900 700 500 300 200],'k-s',...
[1500  1600 1700 1800 1900 2000 2100],[2100 1900 1800 1600 1500 1300
1100],'k-s',...
[1500 1500 1500 1500 1500 1500 1500 1500 1500 1500 1500], [1900 1700
1500  1300 1100 900 700 600 500 300 200],'k-s',...
 [1500 1600], [1700 1600],'k-s',...
 [1500 1600], [1500 1600],'k-s',...
 [1700 1700 1700],[2100 1900 1800],'k-s',...
 [1700 1700 1700 1700 1700 1700 1700 1700 1700], [1500 1300 1100 900
700 600 500 300 200],'k-s',...
 [1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900],[2100 1900
1700 1600 1500 1300 1100 900 700 500 300],'k-s',...
 [2100 2100 2100 2100 2100 2100],[1900 1700 1300 1100 900 700],'k-
s',...
 [2100 2100],[500 300],'k-s', 'LineWidth',2,'MarkerEdgeColor','k',
'MarkerFaceColor','k', 'MarkerSize',4); xlim([0 2300]); ylim([0 2300])

i=input('What is your starting point?:');
j=input('What is your destination?:');
coord=[200 400 500 700 900 1100 1300 1500 1700 1900 ...
    200 300 400 500 700 900 1100 1300 1500 1600 1700 1900 2100 ...
    900 1100 1300 1700 ...
    200 400 600 800 900 1100 1300 1500 1900 2100 ...
    900 1600 1800 1900 ...
    200 400 600 700 900 1000 1100 1300 1500 1700 1900 ...
    400 600 700 900 1100 1500 1700 1900 2000 2100 ...
    200 400 500 600 900 1100 1300 1500 1700 1900 2100 ...
    200 500 600 900 1100 1300 1500 1700 1900 2000 2100 ...
    200 400 500 600 900 1100 1300 1500 1700 1900 2000 2100 ...
    1500 1700 ...
    200 400 500 600 700 900 1100 1300 1500 1700 1900 2000 2100 ...
    200 400 500 600 700 1100 1300 1500 1700 1900 2000 2100 ...
    700 1100 1300 1500 1700 ...
    400 500 600 700 ; 2100 2100 2100 2100 2100 2100 2100 2100 2100 2100
...
    1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900
...
    1800 1800 1800 1800 ...
    1700 1700 1700 1700 1700 1700 1700 1700 1700 1700 ...
    1600 1600 1600 1600 ...
    1500 1500 1500 1500 1500 1500 1500 1500 1500 1500 1500 ...
    1300 1300 1300 1300 1300 1300 1300 1300 1300 1300 ...
    1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 ...
    900 900 900 900 900 900 900 900 900 900 900 ...
    700 700 700 700 700 700 700 700 700 700 700 700 ...
    600 600 ...
    500 500 500 500 500 500 500 500 500 500 500 500 500 ...
    300 300 300 300 300 300 300 300 300 300 300 300 ...
    200 200 200 200 200 ...
    100 100 100 100];
m=(coord(2,i)-coord(2,j))/(coord(1,i)-coord(1,j));
pm=-1/m;
xm=(coord(1,i)+coord(1,j))/2;
ym=(coord(2,i)+coord(2,j))/2;
nh=1;
ng=1;

for k=1:size(coord,2)
    f=pm*(coord(1,k)-xm)+ym;
    if
((coord(1,i)<=coord(1,k))&&(coord(1,k)<=coord(1,j)))&&((coord(2,j)<=coo
rd(2,k))&&(coord(2,k)<=coord(2,i)))
           if coord(2,k)>f
              h(1,nh)=k;
              nh=nh+1;
           else
              g(1,ng)=k;
              ng=ng+1;
           end
    elseif
((coord(1,i)<=coord(1,k))&&(coord(1,k)<=coord(1,j)))&&((coord(2,i)<=coo
rd(2,k))&&(coord(2,k)<=coord(2,j)))
            if coord(2,k)<f
                h(1,nh)=k;
                nh=nh+1;
            else
                 g(1,ng)=k;
                 ng=ng+1;
            end

    elseif
((coord(1,j)<=coord(1,k))&&(coord(1,k)<=coord(1,i)))&&((coord(2,i)<=coo
rd(2,k))&&(coord(2,k)<=coord(2,j)))

           if coord(2,k)<f
              h(1,nh)=k;
              nh=nh+1;
           else
               g(1,ng)=k;
              ng=ng+1;
           end
    elseif
((coord(1,j)<=coord(1,k))&&(coord(1,k)<=coord(1,i)))&&((coord(2,j)<=coo
rd(2,k))&&(coord(2,k)<=coord(2,i)))
           if coord(2,k)>f
              h(1,nh)=k;
              nh=nh+1;
           else
              g(1,ng)=k;
              ng=ng+1;
           end
    end
end
mtime=0
fpath=[]
h3=h
h2=h;
g2=g;
rnodes=[h g]
while exist('spath')==0


for q=1:size(rnodes,2);
    disth=sqrt(((coord(1,i)-coord(1,rnodes(1,1)))^2)+((coord(2,i)-...
        coord(2,rnodes(1,1)))^2));
    for k=1:size(rnodes,2);
           if sqrt(((coord(1,i)-coord(1,rnodes(1,k)))^2)+((coord(2,i)-
...
                       coord(2,rnodes(1,k)))^2))<=disth
              eh=k;
              disth=sqrt(((coord(1,i)-
coord(1,rnodes(1,k)))^2)+((coord(2,i)-...
                  coord(2,rnodes(1,k)))^2));
              th=rnodes(1,k);
          end
    end
    fh(1,q)=th;
    rnodes(:,eh)=[];


end
fh;
H=fh
N=[fh]

   % Route finding algorithm
w=1;
u=0;
p=0;
z=1;
sum=0;
Tsum=inf;
T=inf(132,132);

T(1,2)=2 ; T(1,11)=3;
T(2,1)=3; T(2,13)=3; T(2,3)=4;
T(3,2)=3; T(3,14)=4; T(2,4)=5;
T(4,3)=6; T(4,15)=1; T(4,5)=3;
T(5,4)=4; T(5,16)=7; T(5,6)=8;
T(6,5)=9; T(6,17)=3; T(6,7)=4;
T(7,6)=2; T(7,18)=4; T(7,8)=5;
T(8,7)=6; T(8,20)=7; T(8,9)=6;
T(9,8)=4; T(9,21)=8; T(9,10)=8;
T(10,9)=6; T(10,22)=3;
T(11,1)=4; T(11,12)=inf;
```

```
T(12,11)=2; T(12,29)=3; T(12,13)=2;
T(13,12)=2; T(13,14)=4;T(13,2)= 5;
T(14,13)=6; T(14,3)=5; T(14,15)=9;
T(15,14)=11; T(15,16)=9; T(15,4)=8; T(15,30)=9; T(15,31)=7;
T(16,15)=6; T(16,5)=8;  T(16,17)=9; T(16,24)=8;
T(17,16)=7; T(17,6)=7; T(17,18)=9; T(17,25)=8;
T(18,17)=9; T(18,19)=6; T(18,7)=6; T(18,26)=10;
T(19,18)=9 ; T(19,20)=8 ; T(19,35)=4;
T(20,8)=7; T(20,19)=12; T(20,21)=3; T(20,27)=3;
T(21,20)=7; T(21,9)=6; T(21,22)=5; T(21,27)=3;
T(22,21)=12; T(22,10)=7; T(22,23)=7; T(22,36)=5;
T(23,22)=10; T(23,37)=7;
T(24,16)=12; T(24,32)=7;T(24,25)=12;
T(25,17)=6; T(25,24)=7;T(25,33)=9; T(25,26)=7;
T(26,25)=11; T(26,18)=7;T(26,34)=6;
T(27,20)=5; T(27,21)=5;
T(28,42)=6; T(28,29)=3;
T(29,12)=7; T(29,28)=4; T(29,30)=11; T(29,43)=7;
T(30,29)=12; T(30,15)=5; T(30,31)=3; T(30,44)=8;
T(31,30)=7; T(31,15)=6; T(31,32)=3; T(31,38)=4;
T(32,31)=4; T(32,24)=2; T(32,33)=12;
T(33,32)=5; T(33,25)=5; T(33,34)=12; T(33,48)=4;
T(34,33)=3; T(34,26)=2; T(34,35)=11; T(34,49)=4;
T(35,34)=8; T(35,19)=5; T(35,39)=12; T(35,50)=4;
T(36,22)=7; T(36,37)=6 ;T(36,41)=12;
T(37,23)=3; T(37,36)=6;T(37,62)=12;
T(38,31)=3; T(38,46)=1;T(38,47)=10;
T(39,35)=2; T(39,50)=3; T(39,40)=9;
T(40,39)=2; T(40,27)=11;T(40,52)=5;T(40,41)=5;
T(41,13)=6; T(41,3)=5; T(41,15)=7;
T(42,28)=5; T(42,43)=6;
T(43,42)=6; T(43,29)=4;T(43,53)=9;
T(44,30)=2; T(44,45)=3; T(44,54)=2;
T(45,44)=2; T(45,55)=4;T(45,46)= 5;
T(46,45)=6; T(46,38)=5; T(46,47)=3;T(46,56)=4;
T(47,46)=11; T(47,38)=9; T(47,48)=3;
T(48,47)=4; T(48,33)=8;  T(48,49)=9; T(48,57)=7;
T(49,48)=6; T(49,34)=8;  T(49,50)=9;
T(50,49)=7; T(50,35)=7; T(50,51)=9; T(50,58)=8; T(50,39)=8;
T(51,50)=6; T(51,52)=8;  T(51,59)=9;
T(52,51)=7; T(52,41)=3; T(52,60)=9; T(52,40)=6;T(52,61)=8;
T(53,43)=9; T(53,64)=6; T(53,54)=6;
T(54,44)=9 ; T(54,53)=8 ; T(54,55)=4;
T(55,54)=3; T(55,45)=9; T(55,56)=8;
T(56,55)=7; T(56,46)=6; T(56,57)=5;
T(57,56)=7; T(57,48)=7; T(57,68)=7;
T(58,50)=10; T(58,70)=7; T(58,59)=7;
T(59,58)=10; T(59,51)=7;T(59,71)=10;T(59,60)=5;
T(60,59)=6; T(60,52)=7;T(60,61)=9; T(60,72)=7;
T(61,60)=11; T(61,52)=7;T(61,62)=6;T(61,73)=7;
T(62,61)=5; T(62,37)=5; T(62,73)=5;
T(63,64)=6; T(63,74)=3;

T(64,63)=11; T(64,53)=9; T(64,65)=8;
T(65,64)=9; T(65,75)=8;  T(65,66)=9;
T(66,65)=6; T(66,76)=8;  T(66,67)=9;
T(67,66)=7; T(67,68)=7; T(67,77)=9;
```

```
T(68,67)=9; T(68,69)=6; T(68,57)=6; T(68,78)=10;
T(69,68)=9 ; T(69,70)=8 ; T(69,79)=4;
T(70,69)=7; T(70,71)=12; T(70,58)=3; T(70,80)=3;
T(71,70)=7; T(71,72)=6; T(71,59)=5; T(71,81)=3;
T(72,71)=12; T(72,73)=7; T(72,60)=7; T(72,82)=5;
T(73,72)=10; T(73,61)=7;T(73,62)=10; T(73,84)=7;
T(74,63)=12; T(74,85)=7;
T(75,65)=6; T(75,87)=7;T(75,76)=9;
T(76,75)=11; T(76,66)=7;T(72,88)=6;
T(77,67)=5; T(77,78)=5;T(77,89)=5;
T(78,77)=6; T(78,68)=3;T(78,91)=6; T(78,90)=3;
T(79,69)=7; T(79,80)=4; T(79,91)=11;
T(80,79)=12; T(80,70)=5; T(80,81)=3; T(80,92)=8;
T(81,80)=7; T(81,71)=6; T(81,93)=3;
T(82,72)=4; T(82,94)=2; T(82,83)=12;
T(83,82)=5; T(83,84)=5; T(83,95)=12;
T(84,83)=3; T(84,73)=2; T(84,96)=11;
T(85,74)=8; T(85,86)=5;
T(86,85)=7; T(86,87)=6 ;T(86,100)=12;
T(87,86)=3; T(87,75)=6;T(87,88)=12;T(87,101)=6;
T(88,87)=3; T(88,76)=1;T(88,89)=6; T(88,102)=3;
T(89,88)=2; T(89,77)=3; T(89,90)=9;T(89,104)=7;
T(90,89)=2; T(90,78)=11;T(90,91)=5;T(90,105)=5;
T(91,90)=6; T(91,78)=5;T(91,79)=7;T(91,92)=6; T(91,106)=3;
T(92,91)=5; T(92,80)=6; T(92,93)=6; T(92,97)=3;
T(93,92)=6; T(93,81)=4;T(93,94)=9;T(93,98)=6;
T(94,93)=2; T(94,82)=3; T(94,95)=7; T(94,109)=3;
T(95,94)=2; T(95,83)=4;T(95,96)= 5;T(95,110)=6;
T(96,95)=6; T(96,84)=5;
T(97,92)=11; T(97,98)=9; T(97,107)=3;
T(98,97)=4; T(98,93)=8;  T(98,108)=9;
T(99,100)=6; T(99,112)=3;
T(100,99)=7; T(100,86)=7; T(100,101)=9; T(100,113)=8;
T(101,100)=6; T(101,87)=8;  T(101,102)=9;
T(102,101)=7; T(102,88)=3; T(102,103)=9;
T(103,102)=9; T(103,104)=6; T(103,116)=6;
T(104,103)=9; T(104,89)=8 ; T(104,105)=4;
T(105,104)=3; T(105,90)=9; T(105,106)=8;T(105,117)=6;
T(106,105)=7; T(106,91)=6; T(106,107)=5;T(106,118)=6;
T(107,106)=7; T(107,97)=7; T(107,119)=7;
T(108,98)=10; T(108,109)=7; T(108,120)=7;
T(109,108)=10; T(109,94)=7;T(109,110)=10;T(109,121)=5;
T(110,109)=6; T(110,95)=7;T(110,111)=9; T(110,122)=7;
T(111,110)=11; T(111,123)=7;
T(112,99)=5; T(112,113)=5;
T(113,112)=6; T(113,100)=3;T(113,114)=6; T(113,129)=3;
T(114,113)=3; T(114,115)=2; T(114,130)=11;
T(115,114)=8; T(115,116)=5; T(115,131)=12;
T(116,115)=7; T(116,103)=6 ;T(116,117)=12;T(116,124)=12;
T(117,116)=3; T(117,105)=6;T(117,118)=12;T(117,125)=12;
T(118,117)=9; T(118,106)=1;T(118,119)=10;T(118,126)=12;
T(119,118)=2; T(119,107)=3; T(119,120)=9;T(119,127)=12;
T(120,119)=2; T(120,108)=11;T(120,121)=5;T(120,128)=5;
T(121,120)=6; T(121,109)=5; T(121,122)=7;
T(122,121)=5; T(122,110)=6;T(122,123)=9;
T(123,122)=6; T(123,111)=4;
T(124,116)=2; T(124,125)=3; T(124,132)=2;
```

```
T(125,124)=2; T(125,117)=4;T(125,126)= 5;
T(126,125)=6; T(126,118)=5; T(126,127)=3;
T(127,126)=11; T(127,119)=9; T(127,128)=3;
T(128,127)=4; T(128,120)=8;
T(129,113)=6; T(129,130)=8;
T(130,129)=7; T(130,114)=7; T(130,131)=9;
T(131,130)=6; T(131,115)=8;  T(131,132)=9;
T(132,131)=7; T(132,124)=3;


v=w;
for h=1:2000

    if T(N(1,w),N(1,v+u+1))~=inf
            p=p+1;
            c(1,p)=w;
            c(2,p)=v+u+1;
            sum=sum+T(N(1,w),N(1,v+u+1));
            path(1,z)=N(1,w);
            path(1,z+1)=N(1,v+u+1)
            w=v+u+1;
            z=z+1;
            if (v+u+1==size(N,2))
                    if sum<Tsum
                       Tsum=sum
                       spath=path
                    end
                    if p~=1
                        sum=sum-T(N(1,c(1,p)),N(1,c(2,p)))-T(N(1,c(1,p-
1)),N(1,c(2,p-1)));
                        path(1,z)=0;
                        path(1,z-1)=0;
                       w=c(1,p-1);
                       v=w;
                       u=c(1,p)-c(1,p-1);
                       p=p-2;
                       z=z-2;
                    else
                       break
                    end
            else
                u=0;
                v=w;
            end

    else
        if v+u+1==size(N,2)
            if p~=0
            w=c(1,p);
            v=w;
            u=c(2,p)-c(1,p);
            sum=sum-T(N(1,c(1,p)),N(1,c(2,p)));
            npath=path;
             path(1,z)=0;
            p=p-1;
            z=z-1;
            else
                break

            end
        else
            u=u+1;
        end

    end
end
time=0
if exist('spath')==0
    sizenpath=size(npath,2)
                     for k=1:sizenpath
                       if npath(1,k)>0
                          s_npath(1,k)=npath(1,k);
                       else
                         break
                       end

                     end
     sizenpath=size(s_npath,2)
    i=npath(1,sizenpath)
    h=h3
    for k=1:(sizenpath-1)
        time=time+T(s_npath(1,k),s_npath(1,k+1))
    end
    mtime
    mtime=mtime+time
    fpath=[fpath s_npath]
end
end
Tsum=Tsum+mtime
spath=[spath fpath]
for k=1:size(spath,2)
        if spath(1,k)>0
            s_path(1,k)=spath(1,k);
        else
            break
        end

end

disp(['The shortest path includes the following nodes: ',
num2str(s_path)]);
disp(['The estimated time is (Minute): ', num2str(Tsum)]);

for k=1:size(spath,2)
        if spath(1,k)>0
           vec1(1,k)=coord(1,spath(1,k));
           vec2(1,k)=coord(2,s_path(1,k));
        else
            break
        end
```

```
end


plot([200 400 500 700 900 1100 1300 1500 1700 1900], [2100 2100 2100
2100 2100 2100 2100 2100 2100 2100],'k-s',...
[200 300 400 500 700 900 1100 1300 1500 1600 1700 1900 2100],[1900 1900
1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900],'k-s',...
[900 1100 1300], [1800 1800 1800], 'k-s',...
[200 400 600 800 900 1100 1300 1500], [1700 1700 1700 1700 1700 1700
1700 1700], 'k-s',...
[1900 2100], [1700 1700], 'k-s', ...
[1600 1800 1900], [1600 1600 1600],'k-s', ...
[200 400], [1500 1500],'k-s', ...
[600 700 900 1000 1100 1300 1500 1700 1900], [1500 1500 1500 1500 1500
1500 1500 1500 1500], 'k-s',...
[400 600 700 900 1100],[1300 1300 1300 1300 1300],'k',...
[1500 1700 1900 2000 2100], [1300 1300 1300 1300 1300],'k',...
[ 200 400 500 600 900 1100 1300 1500 1700 1900 2100], [1100 1100 1100
1100 1100 1100 1100 1100 1100 1100 1100], 'k-s',...
[500 600], [900 900],'k-s',...
[900 1100], [900 900], 'k-s',...
[1300 1500 1700], [900 900 900],'k-s',...
[1900 2000 2100], [900 900 900], 'k-s',...
[200 400 500 600 900 1100 1300 1500 1700 1900 2000 2100], [700 700 700
700 700 700 700 700 700 700 700 700], 'k-s',...
[1500 1700], [600 600], 'k-s',...
[200 400 500 600 700 900 1100 1300 1500], [500 500 500 500 500 500 500
500 500], 'k-s',...
[1700 1900 2000 2100], [500 500 500 500], 'k-s',...
[200 400 500 600 700 1100 1300 1500 1700 1900 2000 2100], [300 300 300
300 300 300 300 300 300 300 300 300], 'k-s',...
[700 1100 1300 1500 1700], [200 200 200 200 200], 'k-s',...
[400 500 600 700], [100 100 100 100], 'k-s',...
[200 200],[2100 1900],'k-s',...
[200 200],[1700 1500], 'k-s',...
[200 200 200],[1100 900 700],'k-s',...
[200 200],[500 300], 'k-s',...
[400 400],[2100 1900],'k-s',...
[300 400 400 400 400],[1900 1700 1500 1300 1100],'k-s',...
[400 400 400 400],[700 500 300 100],'k-s',...
[500 500],[2100 1900],'k-s',...
[500 500 500 500],[1100 900 700 500],'k-s',...
[500 500],[300 100],'k-s',...
[700 700 600 600 600],[2100 1900 1700 1500 1300],'k-s',...
[600 600 600 600],[1100 900 700 500],'k-s',...
[600 600],[300 100],'k-s',...
[700 700],[1500 1300],'k-s',...
[700 700 700 700],[500 300 200 100],'k-s',...
[700 800 900 1000],[1900 1700 1600 1500],'k-s',...
[900 900 900 900],[2100 1900 1800 1700],'k-s',...
[900 900 900],[1600 1500 1300], 'k-s',...
[900 900 900 900],[1100 900 700 500],'k-s',...
[1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100 1100], [2100
1900 1800 1700 1500 1300 1100 900 700 500 300 200],'k-s',...
[1100 1300], [900 700],'k-s',...
[1300 1300 1300 1300 1300],[2100 1900 1800 1700 1500],'k-s',...
[1300 1300 1300 1300 1300 1300], [1100 900 700 500 300 200],'k-s',...
[1500  1600 1700 1800 1900 2000 2100],[2100 1900 1800 1600 1500 1300
1100],'k-s',...
[1500 1500 1500 1500 1500 1500 1500 1500 1500 1500 1500], [1900 1700
1500  1300 1100 900 700 600 500 300 200],'k-s',...
 [1500 1600], [1700 1600],'k-s',...
 [1500 1600], [1500 1600],'k-s',...
 [1700 1700 1700],[2100 1900 1800],'k-s',...
 [1700 1700 1700 1700 1700 1700 1700 1700 1700], [1500 1300 1100 900
700 600 500 300 200],'k-s',...
 [1900 1900 1900 1900 1900 1900 1900 1900 1900 1900 1900],[2100 1900
1700 1600 1500 1300 1100 900 700 500 300],'k-s',...
 [2100 2100 2100 2100 2100 2100],[1900 1700 1300 1100 900 700],'k-
s',...
 [2100 2100],[500 300],'k-s',vec1, vec2, '-rs',
'LineWidth',2,'MarkerEdgeColor','k', 'MarkerFaceColor','k',
'MarkerSize',4); xlim([0 2300]); ylim([0 2300])
```