ABSTRACT

Title of Thesis:	Route Planning with Statistical Models
	Yufei Huang, Master of Science, 2018
Thesis Directed By:	Associate Professor, Dr. Ilya O. Ryzhov, Department of Decision, Operations and Information Technologies Robert H. Smith School of Business

One difficulty to find the fastest route in route planning is how to determine the precise travel time on each road. In the real world, the travel time of each road varies with time, weather condition and many other factors. The thesis aims at studying route planning algorithms that use statistical models to predict the changes of travel time for each road and calculate the fastest route. Using the historical data of main roads in Washington D.C. area, the thesis studied major factors that would affect the travel time. Different statistical models are presented and compared to fit the travel time of each road. Then the LASSO regression model is chosen, and different predictive route planning algorithms are introduced to fulfill our goal. Finally, deterministic approximate dynamic programming is recommended to solve our problem.

ROUTE PLANNING WITH STATISTICAL MODELS

by

Yufei Huang

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2018

Advisory Committee:
Professor Ilya O. Ryzhov, Chair
Professor S. Raghu Raghavan
Professor Jeffrey W. Herrmann

© Copyright by
Yufei Huang
2018

Dedication

I would like to dedicate the thesis to my grandfather, Mr. Xuexi Huang, who passed away on March 9th, 2018. I want to thank him for his endless love to me, and I am extremely mournful that I could not go back home and join his funeral. His spirit and his kind heart shall live on with me.

Acknowledgments

First of all, I want to express my sincere gratitude to my parents, who gave me life and allow me to experience the beauty of the world. They never stop supporting me, and they provide me the opportunity to study abroad to learn new knowledge. I would never have been able to succeed without them.

This thesis would not have been possible without the guidance of my advisor, Dr. Ryzhov. I had so little knowledge about statistics, and it was my advisor who had been very patient all the time, teaching me and correcting my silly mistakes. Dr. Ryzhov kept inspiring me throughout the thesis study and sharing me with his profound knowledge.

Also, I would like to thank Dr. Herrmann and Dr. Raghavan for being my thesis committee members and providing me with valuable suggestions. I would also thank them for giving me interesting lectures on Decision Making and Linear Programming. Dr. MacCarthy, my academic advisor, is always helpful and concerning about the difficulties I met. I really appreciate the assistance he offered.

Last but not least, I would like to thank all my friends at the University of Maryland, who accompany me through my two years' graduate student life. They leave me warm memories that I shall never forget.

Thank you all.

Yufei Huang

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RITIS	Regional Integrated Transportation Information System
CATT Lab	The Center for Advanced Transportation Technology Laboratory
TMC	Traffic Message Channel
LM	Linear Regression Model
LASSO	Least Absolute Shrinkage and Selection Operator
GLM	Generalized Linear Model
SIS	Sure Independence Screening
Q-Q Plot	Quantile-Quantile Plot
MSE	Mean Squared Error
DP	Dynamic Programming
ADP	Approximate Dynamic Programming
CV	Cross-Validation

Chapter 1: Introduction

Motivation

Traffic congestion has become a matter of concern to all of us, and it is one of the leading factors that restrict the normal operation of our society and economy (Arnott, 2001). Not all roads in a road system are crowded during rush hours. Exhaustively allocating road resource, such as guiding drivers to use roads with less traffic, would help to relieve the traffic pressure. Traffic condition would be different in different periods of a day. For example, I-495, which is a part of Washington D.C. beltway, is extremely busy during rush hours. Moreover, traffic conditions of a road on different sides would also vary from each other. For example, traffic into Washington, D.C. on the road I-395 in Virginia is heavy in the morning, while the traffic out of Washington, D.C. on I-395 is massive in the evening. In this case, highways cannot be merely given higher priority by automotive navigation systems to ordinary roads at any time of a day. Sometimes, traveling through certain ordinary roads is fast than taking highways. Also, drivers should be informed to avoid taking the roads that are crowded during their travel.

Furthermore, the traffic condition and weather condition of different roads would keep changing during our journey. For a long-distance trip that might take several hours, the weather conditions of various road pieces could significantly vary from each other. The traffic condition of different road pieces might also change from the time point when the route is planned to the moment when the car is traveling to them. Therefore, it is of significance if we can predict the traffic condition of the roads we are going to travel to and avoid taking the routes that would be busy in the future.

This thesis aims at studying the predictive route planning problem, which considers the shortest travel time between our starting point and the destination. We require a predictive route planning method that can take real-time traffic and weather conditions into consideration, predict their changes, and evaluate their effect on our travel time at the very beginning.

Background

In the late 1980s, people introduced electronic technology, communication technology, and computer technology into the transportation system, which can improve traffic congestion, traffic safety, and road accident rescue. Such system is named as the intelligent transportation system. Route Planning is one of the most significant parts of the intelligent transportation system (Wootton, García-Ortiz, & S.M., 1995). It is responsible for finding a minimum-cost path for the users, which can either be the shortest distance or the least travel time from the origin to the destination. With the increasingly complex traffic environment nowadays, a well-planned route is essential because it can not only help personal users save their time during their journeys but also can create economic value for companies.

We already have multiple efficient point-to-point shortest path algorithms to help us fulfill our demand to find the desired path. Dijkstra's algorithm and A* Search are two widely used methods for detecting the optimum route. The difficulty of applying these route planning methods to solve our real-world route planning problem is how to determine the precise cost values of each road, which would be the travel time on each road in this thesis. Performing data analysis and obtaining the statistical models of the travel time on different roads can help us make judgments and take optimal actions to choose the best path. In 2012, a group of engineers from IBM tackled traffic congestion problems in Boston by merging multiple data streams to predict the traffic conditions of different roads in a day (IBM Corporation, 2012). They used the data captured from citizen's cell phones, street sensors, and surveillance cameras to analyze the traffic data. By using the acquired results, they successfully helped people in Boston plan their traveling routes to avoid congested roads and make the city smarter.

The success story gives us an example that by analyzing historical traffic data, we can find the major factors that would affect the travel time of different roads. By performing statistical analysis, we would get the quantitative relationships between these factors and obtain the predictive models for the travel time.

Scope of Work

In our thesis, we are going to do some surveys on different statistical models. Also, we will modify and implement some existing route planning methods. We want to combine these techniques to solve the predictive route planning problem.

The main scope of the work can be divided into two parts. The first part is to select, gather, and analyze the traffic data and weather data. We are going to select some highways and main streets in Washington D.C. area. After we have retrieved all the data we need, we will implement different statistical models, including linear regression, LASSO regression, generalized linear model, and sure independence screening, which will be introduced in the later chapters. We will then use cross-validation to compare the accuracy of each model and choose a best fit traffic model for each road.

The second part of this thesis is to use the traffic model we obtained to implement the predictive route planning. We will first use traditional route planning algorithms like Dijkstra's Algorithm to solve our problem. Then we are going to explore deterministic approximate dynamic programming.

Structure of the Thesis

The first chapter states the research problem and some background of the research question. It also gives us an outline of the order of information in the thesis.

In Chapter 2, we focus on the data analysis of the traffic and weather information of some main roads in Washington D.C. area. We first present how we selected the roads and dates that we are going to study. We also show the relevant data we retrieved in section 1. In section 2, we talk about the 4 different kinds of statistical models. We discuss how to use LASSO regression and sure independence screening to select the most significant factors to the travel time. In the last section, we show how to use cross-validation to compare these models and make the conclusion which model is the best fit for our problem.

Predictive route planning algorithms are presented in Chapter 3. We first introduce some basic concepts from graph theory. Then we discuss some common techniques that are widely used in solving shortest path problems, including breadth-first search, depth-first search, and Dijkstra's algorithm. In the 2nd section, we show how we used these traditional methods to find the fastest travel path in our road network. In the 3rd section, we introduced dynamic programming and approximate dynamic programming. We also show how to apply approximate dynamic programming to our question in the last section.

In Chapter 4, we perform the analysis of algorithms which are mentioned in the previous chapter. We discuss the space complexity of the algorithms and introduce a method to reduce the memory space required. Also, we calculate the computational time of each of the predictive route planning algorithms. We compare the algorithms and see how they perform in different road networks with different numbers of nodes and arcs. We also test these algorithms to run multiple times and see their long-term performance. Finally, we present how our program fulfills the aim of the thesis, which is to give us different optimal routes with the same origin and destination to avoid some crowded roads during different time periods of a day.

We conclude some finding to our predictive route planning methods in our last chapter, which is the Chapter 5. Also, we put forward some ideas on possible future work in our last chapter.

Chapter 2: Model Analysis

In this chapter, we introduce different statistical models for the roads in Washington D.C. area. We will talk about how we select the roads and analyze the most significant factors that would affect the travel time on these roads. The chapter is divided into three sub-sections. In section 1, we first take a brief look at how other researchers studied the influencing factors of traffic conditions. Then we collect traffic data and weather data based on their conclusions. In the 2nd section, we introduced 3 different types of regression models, which are Linear Regression, LASSO Regression, and Generalized Linear Model. Also, we apply Sure Independence Screening to screen out the less significant factors in our models. We used R to run these methods on our data and built four different kinds of traffic models to calculate the travel time on different roads. In our last section, we compared these four kinds of models by using cross-validation and selected the most suitable model for our road network. Through building and comparing these models, we obtained the predictive functions that can help us calculate the travel time of different roads, which will be used to support the predictive route planning methods to find the most time-saving route in the next chapter.

Data Collection

Roads and Factors Selection

In this part, we are going to look through the functional classification of roads, which is to group the roads with the same features together, and the analysis of the factors that would affect traffic conditions. We will use their conclusions to determine which roads we are going to study and what kinds of factors we need to collect for building our models.

Table 2.1 Road Classification by the U.S. administrative system

Primary	Interstate Route
	US Route
	US Business Route
	State Route in dense urban areas
Secondary	State Route
	State Business Route
	Major Urban Street
Tertiary	County Route
	Forest Route
	Indian Route
	National Scenic Byway
Unclassified	Minor Road

We can categorize the roads according to their functions and capacities to achieve management aims to ensure traffic flows freely and safely. In the United States and Canada, the basic hierarchy comprises freeways, arterials, collectors, and local road (U.S. Department of Transportation, Federal Highway Administration, 1974). According to the definitions of these four types of roads, freeways are typically designed for high-speed vehicular traffic. The primary function of an arterial road is to deliver traffic from collector roads to freeways. Collector roads are designed to provide access to residential properties, and the local roads are some streets with the lowest speed limit and carry low volumes of traffic. The U.S.

administrative system further defined the four levels that the roads can be classified into, which are presented in table 2.1.

Usually, the first two types of roads are responsible for a large amount of traffic travel every day. Traffic jams happening on these roads might decrease the efficiency of daily economic activities and deteriorate the urban transportation system (Bureau of Transportation Statistics, 2017).

In our thesis, we will focus on the roads in D.C. area that belong to the first two categories. To be more specific, we will choose some US routes and state routes. Based on the classifications mentioned above, we are going to choose 3 roads from Maryland, which are (1) I-495, (2) MD-50, and (3) MD-201. We will take 3 roads from Virginia, which are (1) I-66, (2) I-495, and (3) I-395. Also, five roads will be included in our study from Washington, D.C., which are (1) DC-295, (2) DCI-295, (3) DCI-395, (4) George Washington Memorial Parkway, and (5) US-50. Figure 2.1 shows us a brief view of the road network we are going to study.

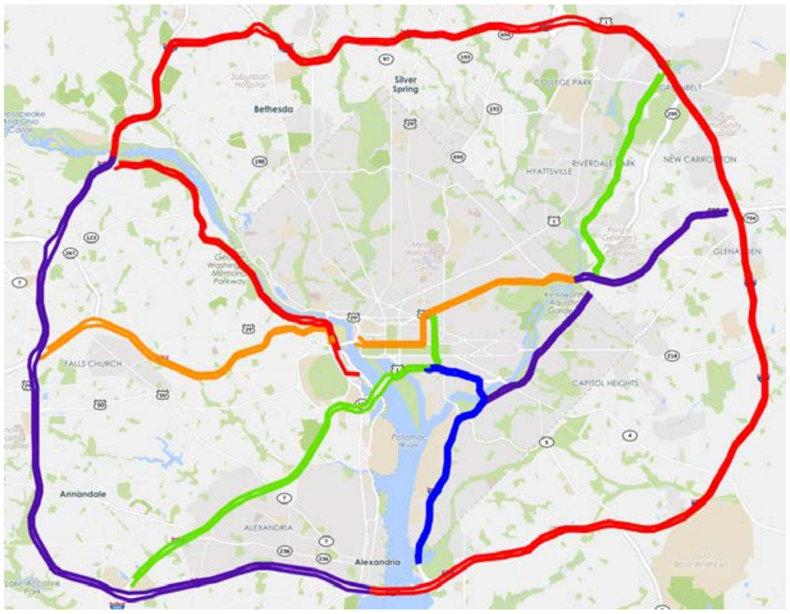


Figure 2.1 Selected Road Network (Retrieved from Ritis.org)

Lay shows us the basic concepts of the road transportation system. He studied the interaction of the roads, people, and the environment of the road system and summarizes the main factors that would affect the traffic condition, which are: (1) condition of the road, which is the geometry of the road, (2) performance of the vehicle, (3) traffic condition, (4) environment and (5) weather (Lay, 2009).

More recently, Li used the number of passing vehicles per meter of a road in an hour to represent the capacity of roads. By analyzing the capacity of roads, Li studied different factors that would affect the traffic conditions on the road in quantity (Li, 2012). Furthermore, he explained the most significant subfactors of these 5 main categories, which is shown in table 2.2.

Table 2.2 Main factors that would affect traffic condition

The condition of the road	Lane width, the number of lanes of a road
	Lateral space of the lane
	The condition of the road surface
	Length of stadia
	The slope of the road
	Length of the ramp
Performance of the vehicle	Ability of acceleration
	Ability of deceleration
	Climbing ability
Traffic condition	Types of cars running on the road
	Distribution of lanes
	Changes in traffic volume
	The frequency of overtaking and transferring lanes
Environment	Landforms
	Bus stops
Weather	Temperature
	Visibility
	Weather conditions. Rain, snow, etc.

Human factors are also important in some complex transportation systems (Hawas, 2013). However, we will not study this aspect in our study because they are usually subjective and varies in different situations. Therefore, we will mainly focus on the factors to traffic condition mentioned above. Our next step is to collect the data corresponding to these factors.

Traffic Data Collection

RITIS is the Regional Integrated Transportation Information System. It is a data-driven platform for transportation analysis, monitoring, and data visualization built and maintained by CATT Lab at the University of Maryland (CATT Lab, 2018). RITIS gathers traffic volume, speed, and road events from roadway sensors, radar, and video. It reorganizes all the data and stores them in the database as is shown in Figure 2.2.

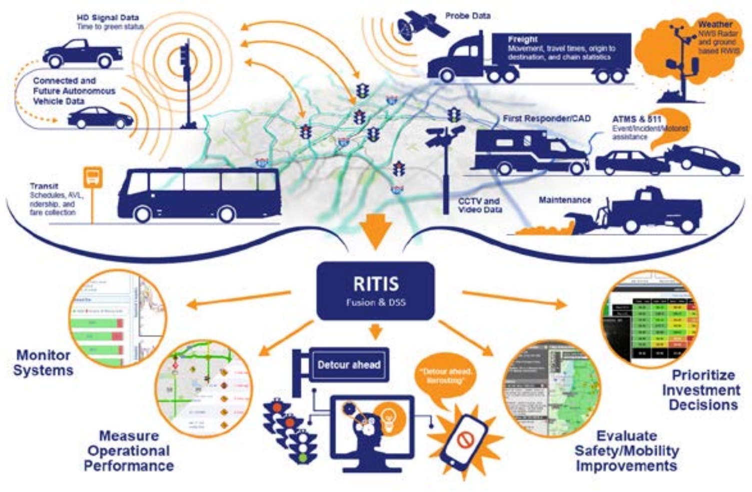

HD Signal Data
Time to green status
Connected and Future Autonomous Vehicle Data
Probe Data
Freight
Movement, travel times, origin to destination, and chain statistics
Weather
NWS Radar and ground based RWIS
First Responder/CAD
ATMS & 511
Event/Incident/Motorist assistance
Transit
Schedules, AVL, ridership, and fare collection
CCTV and Video Data
Maintenance
RITIS
Fusion & DSS
Monitor Systems
Detour ahead
"Detour ahead, Rerouting"
Prioritize Investment Decisions
Measure Operational Performance
Evaluate Safety/Mobility Improvements

Figure 2.2 Operating principle of RITIS (Retrieved from Ritis.org)

Traffic Message Channel (TMC) is a technology for delivering traffic and travel information to motor vehicle drivers. RITIS breaks a road into many different road pieces and uses name each road piece by the TMC it uses. Figure 2.3 illustrates how RITIS names the ID to different road pieces and distinguishes the differences between internal and external TMCs.

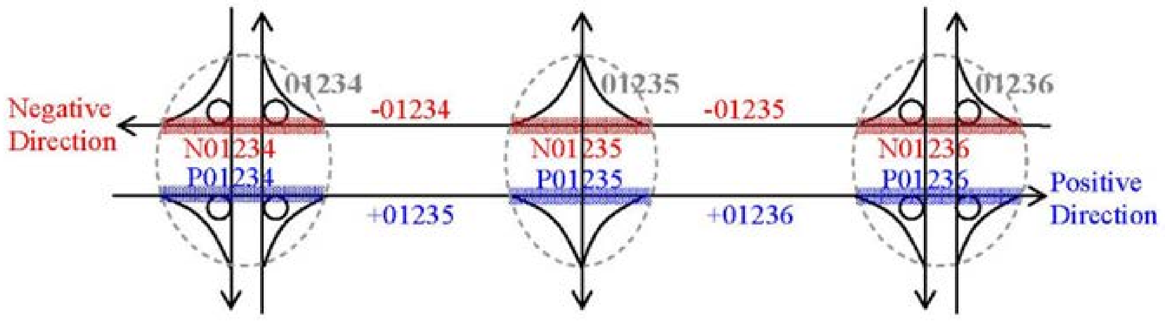


Figure 2.3 TMC Identifications

Meaning of symbols:

"P": Northbound or Eastbound, internal paths

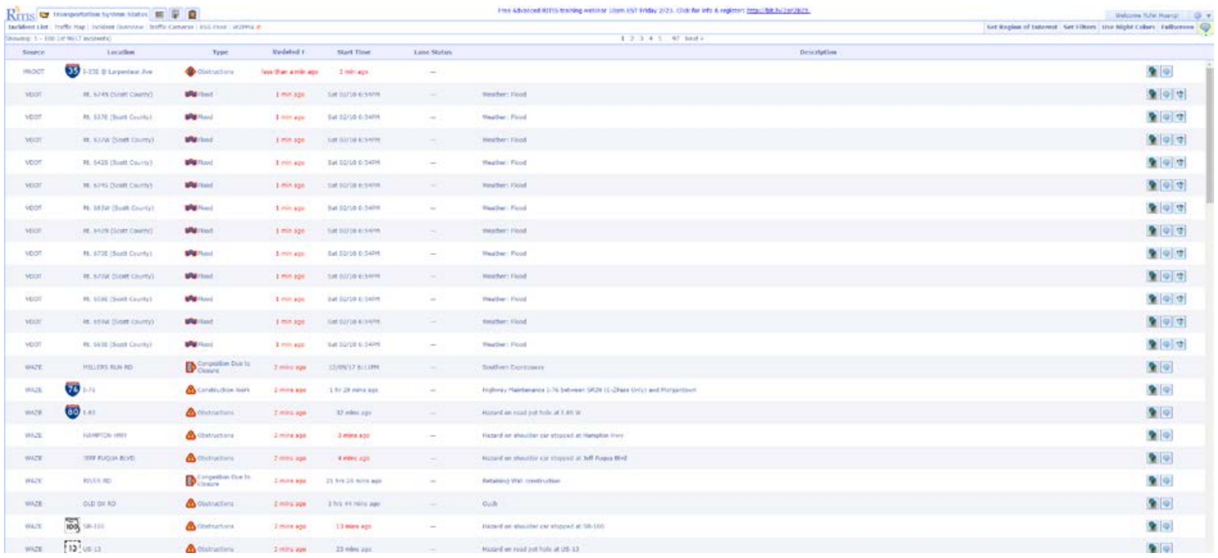
"N": Southbound or Westbound, internal paths

"+": Northbound or Eastbound, external paths

"-": Southbound or Westbound, external paths

TMC location coding describes two types of paths for every location code: “internal path” and “external path.” The internal path refers to the area just past the decision point or intersection at which the TMC code was placed (for example, a freeway off-ramp), while the external path refers to the section of the road leading up the decision point (Rajbhandari, 2012).

Using RITIS, we can either check real-time traffic data or look up historical information from the data archive as is shown in Figure 2.4. In our thesis, we use the traffic information provided by RITIS to perform data analysis. Based on Lay and Li’s conclusion, we are going to collect the data for the road condition from RITIS. Also, we will consider the average speed of cars on the adjoining road pieces. We will use the data of I-395, which is a part of the Capital Beltway in Virginia, as an example to talk about how we did the data analysis in this chapter.



RITIS Transportation System Status

Incident List | Traffic Map | Incident Overview | Traffic Cameras | RSS Feed | [illegible]

Free Advanced RITIS training webinar 10am EST Friday 2/23. Click for info & register: [illegible]

Welcome Yufei Huang!

Set Region of Interest | Set Filters | Use Night Colors | Fullscreen

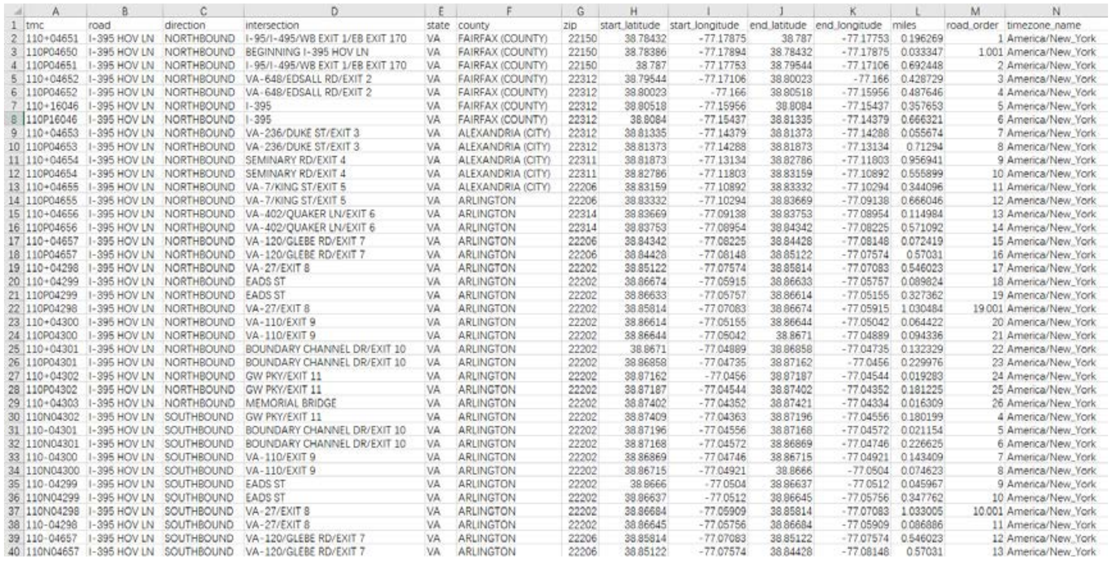
Showing: 1 - 100 (of 9617 incidents)

1 2 3 4 5 ... 97 Next »

Source	Location	Type	Updated ↓	Start Time	Lane Status	Description
MnDOT	I-35E @ Larpenteur Ave	Obstructions	less than a min ago	1 min ago	—	
VDOT	Rt. 674N (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 637E (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 672W (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 642S (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 674S (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 683W (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 642N (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 673E (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 670W (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 658E (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 659W (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
VDOT	Rt. 663E (Scott County)	Flood	1 min ago	Sat 02/10 6:54PM	—	Weather: Flood
WAZE	MILLERS RUN RD	Congestion Due to Closure	2 mins ago	12/09/17 8:11PM	—	Southern Expressway
WAZE	I-76	Construction Work	2 mins ago	1 hr 28 mins ago	—	Highway Maintenance I-76 between SR28 (1-2lane Only) and Morgantown
WAZE	I-80	Obstructions	2 mins ago	32 mins ago	—	Hazard on road pot hole at I-80 W
WAZE	HAMPTON HWY	Obstructions	2 mins ago	3 mins ago	—	Hazard on shoulder car stopped at Hampton Hwy
WAZE	JEFF FUQUA BLVD	Obstructions	2 mins ago	4 mins ago	—	Hazard on shoulder car stopped at Jeff Fuqua Blvd
WAZE	RIVER RD	Congestion Due to Closure	2 mins ago	21 hrs 25 mins ago	—	Retaining Wall construction
WAZE	OLD OX RD	Obstructions	2 mins ago	3 hrs 44 mins ago	—	Ouch
WAZE	SR-100	Obstructions	2 mins ago	13 mins ago	—	Hazard on shoulder car stopped at SR-100
WAZE	US-13	Obstructions	2 mins ago	23 mins ago	—	Hazard on road pot hole at US-13

Figure 2.4 User Interface of RITIS

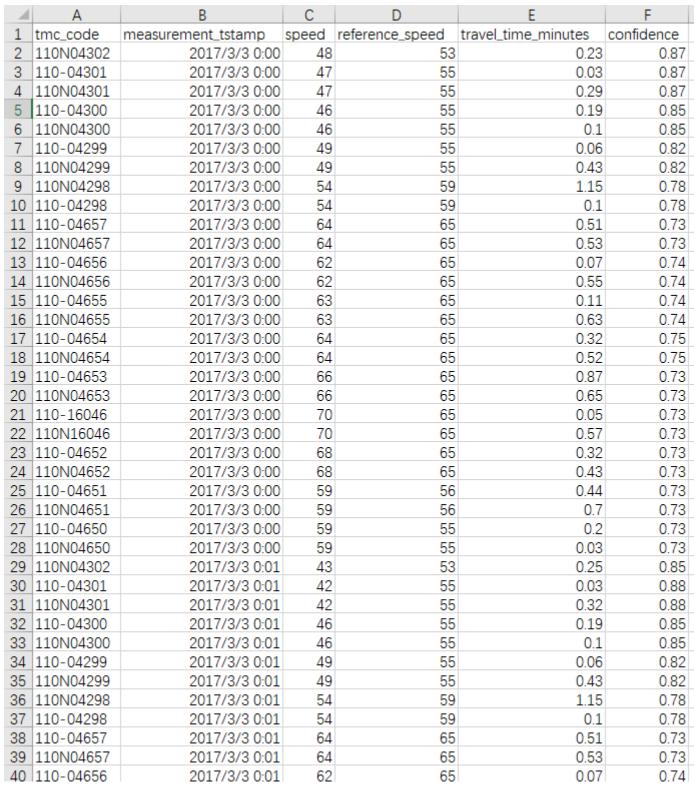
Figure 2.5 shows a part of the identification list for all TMCs retrieved from RITIS. Each road piece is given an ID and accompanied with the location, the length, and many other road information.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	tmc	road	direction	intersection	state	county	zip	start_latitude	start_longitude	end_latitude	end_longitude	miles	road_order	timezone_name
2	110+04651	I-395 HOV LN	NORTHBOUND	I-95/I-495/WB EXIT 1/EB EXIT 170	VA	FAIRFAX (COUNTY)	22150	38.78432	-77.17875	38.787	-77.17753	0.196269	1	America/New_York
3	110P04650	I-395 HOV LN	NORTHBOUND	BEGINNING I-395 HOV LN	VA	FAIRFAX (COUNTY)	22150	38.78386	-77.17894	38.78432	-77.17875	0.033347	1.001	America/New_York
4	110P04651	I-395 HOV LN	NORTHBOUND	I-95/I-495/WB EXIT 1/EB EXIT 170	VA	FAIRFAX (COUNTY)	22150	38.787	-77.17753	38.79544	-77.17106	0.692448	2	America/New_York
5	110+04652	I-395 HOV LN	NORTHBOUND	VA-648/EDSALL RD/EXIT 2	VA	FAIRFAX (COUNTY)	22312	38.79544	-77.17106	38.80023	-77.166	0.428729	3	America/New_York
6	110P04652	I-395 HOV LN	NORTHBOUND	VA-648/EDSALL RD/EXIT 2	VA	FAIRFAX (COUNTY)	22312	38.80023	-77.166	38.80518	-77.15956	0.487646	4	America/New_York
7	110+16046	I-395 HOV LN	NORTHBOUND	I-395	VA	FAIRFAX (COUNTY)	22312	38.80518	-77.15956	38.8084	-77.15437	0.357653	5	America/New_York
8	110P16046	I-395 HOV LN	NORTHBOUND	I-395	VA	FAIRFAX (COUNTY)	22312	38.8084	-77.15437	38.81335	-77.14379	0.666321	6	America/New_York
9	110+04653	I-395 HOV LN	NORTHBOUND	VA-236/DUKE ST/EXIT 3	VA	ALEXANDRIA (CITY)	22312	38.81335	-77.14379	38.81373	-77.14288	0.055674	7	America/New_York
10	110P04653	I-395 HOV LN	NORTHBOUND	VA-236/DUKE ST/EXIT 3	VA	ALEXANDRIA (CITY)	22312	38.81373	-77.14288	38.81873	-77.13134	0.71294	8	America/New_York
11	110+04654	I-395 HOV LN	NORTHBOUND	SEMINARY RD/EXIT 4	VA	ALEXANDRIA (CITY)	22311	38.81873	-77.13134	38.82786	-77.11803	0.956941	9	America/New_York
12	110P04654	I-395 HOV LN	NORTHBOUND	SEMINARY RD/EXIT 4	VA	ALEXANDRIA (CITY)	22311	38.82786	-77.11803	38.83159	-77.10892	0.555899	10	America/New_York
13	110+04655	I-395 HOV LN	NORTHBOUND	VA-7/KING ST/EXIT 5	VA	ALEXANDRIA (CITY)	22206	38.83159	-77.10892	38.83332	-77.10294	0.344096	11	America/New_York
14	110P04655	I-395 HOV LN	NORTHBOUND	VA-7/KING ST/EXIT 5	VA	ARLINGTON	22206	38.83332	-77.10294	38.83669	-77.09138	0.666046	12	America/New_York
15	110+04656	I-395 HOV LN	NORTHBOUND	VA-402/QUAKER LN/EXIT 6	VA	ARLINGTON	22314	38.83669	-77.09138	38.83753	-77.08954	0.114984	13	America/New_York
16	110P04656	I-395 HOV LN	NORTHBOUND	VA-402/QUAKER LN/EXIT 6	VA	ARLINGTON	22314	38.83753	-77.08954	38.84342	-77.08225	0.571092	14	America/New_York
17	110+04657	I-395 HOV LN	NORTHBOUND	VA-120/GLEBE RD/EXIT 7	VA	ARLINGTON	22206	38.84342	-77.08225	38.84428	-77.08148	0.072419	15	America/New_York
18	110P04657	I-395 HOV LN	NORTHBOUND	VA-120/GLEBE RD/EXIT 7	VA	ARLINGTON	22206	38.84428	-77.08148	38.85122	-77.07574	0.57031	16	America/New_York
19	110+04298	I-395 HOV LN	NORTHBOUND	VA-27/EXIT 8	VA	ARLINGTON	22202	38.85122	-77.07574	38.85814	-77.07083	0.546023	17	America/New_York
20	110+04299	I-395 HOV LN	NORTHBOUND	EADS ST	VA	ARLINGTON	22202	38.86674	-77.05915	38.86633	-77.05757	0.089824	18	America/New_York
21	110P04299	I-395 HOV LN	NORTHBOUND	EADS ST	VA	ARLINGTON	22202	38.86633	-77.05757	38.86614	-77.05155	0.327362	19	America/New_York
22	110P04298	I-395 HOV LN	NORTHBOUND	VA-27/EXIT 8	VA	ARLINGTON	22202	38.85814	-77.07083	38.86674	-77.05915	1.030484	19.001	America/New_York
23	110+04300	I-395 HOV LN	NORTHBOUND	VA-110/EXIT 9	VA	ARLINGTON	22202	38.86614	-77.05155	38.86644	-77.05042	0.064422	20	America/New_York
24	110P04300	I-395 HOV LN	NORTHBOUND	VA-110/EXIT 9	VA	ARLINGTON	22202	38.86644	-77.05042	38.8671	-77.04889	0.094336	21	America/New_York
25	110+04301	I-395 HOV LN	NORTHBOUND	BOUNDARY CHANNEL DR/EXIT 10	VA	ARLINGTON	22202	38.8671	-77.04889	38.86858	-77.04735	0.132329	22	America/New_York
26	110P04301	I-395 HOV LN	NORTHBOUND	BOUNDARY CHANNEL DR/EXIT 10	VA	ARLINGTON	22202	38.86858	-77.04735	38.87162	-77.0456	0.229976	23	America/New_York
27	110+04302	I-395 HOV LN	NORTHBOUND	GW PKY/EXIT 11	VA	ARLINGTON	22202	38.87162	-77.0456	38.87187	-77.04544	0.019283	24	America/New_York
28	110P04302	I-395 HOV LN	NORTHBOUND	GW PKY/EXIT 11	VA	ARLINGTON	22202	38.87187	-77.04544	38.87402	-77.04352	0.181225	25	America/New_York
29	110+04303	I-395 HOV LN	NORTHBOUND	MEMORIAL BRIDGE	VA	ARLINGTON	22202	38.87402	-77.04352	38.87421	-77.04334	0.016309	26	America/New_York
30	110N04302	I-395 HOV LN	SOUTHBOUND	GW PKY/EXIT 11	VA	ARLINGTON	22202	38.87409	-77.04363	38.87196	-77.04556	0.180199	4	America/New_York
31	110-04301	I-395 HOV LN	SOUTHBOUND	BOUNDARY CHANNEL DR/EXIT 10	VA	ARLINGTON	22202	38.87196	-77.04556	38.87168	-77.04572	0.021154	5	America/New_York
32	110N04301	I-395 HOV LN	SOUTHBOUND	BOUNDARY CHANNEL DR/EXIT 10	VA	ARLINGTON	22202	38.87168	-77.04572	38.86869	-77.04746	0.226625	6	America/New_York
33	110-04300	I-395 HOV LN	SOUTHBOUND	VA-110/EXIT 9	VA	ARLINGTON	22202	38.86869	-77.04746	38.86715	-77.04921	0.143409	7	America/New_York
34	110N04300	I-395 HOV LN	SOUTHBOUND	VA-110/EXIT 9	VA	ARLINGTON	22202	38.86715	-77.04921	38.8666	-77.0504	0.074623	8	America/New_York
35	110-04299	I-395 HOV LN	SOUTHBOUND	EADS ST	VA	ARLINGTON	22202	38.8666	-77.0504	38.86637	-77.0512	0.045967	9	America/New_York
36	110N04299	I-395 HOV LN	SOUTHBOUND	EADS ST	VA	ARLINGTON	22202	38.86637	-77.0512	38.86645	-77.05756	0.347762	10	America/New_York
37	110N04298	I-395 HOV LN	SOUTHBOUND	VA-27/EXIT 8	VA	ARLINGTON	22202	38.86684	-77.05909	38.85814	-77.07083	1.033005	10.001	America/New_York
38	110-04298	I-395 HOV LN	SOUTHBOUND	VA-27/EXIT 8	VA	ARLINGTON	22202	38.86645	-77.05756	38.86684	-77.05909	0.086886	11	America/New_York
39	110-04657	I-395 HOV LN	SOUTHBOUND	VA-120/GLEBE RD/EXIT 7	VA	ARLINGTON	22206	38.85814	-77.07083	38.85122	-77.07574	0.546023	12	America/New_York
40	110N04657	I-395 HOV LN	SOUTHBOUND	VA-120/GLEBE RD/EXIT 7	VA	ARLINGTON	22206	38.85122	-77.07574	38.84428	-77.08148	0.57031	13	America/New_York

Figure 2.5 Identification list for all TMCs

Figure 2.6 shows the traffic data we retrieved from RITIS. The reference speed represents the so-called “free-flow speed”, which is the speed we would have with very few vehicles on the road. It corresponds to the speed typically measured around midnight or 1 AM. As for the confidence, note that these speeds are estimated by blending (1) information from vehicles equipped with GPS devices and (2) historical data. When the number of probe vehicles is low, more weight would be put on the historical data and report smaller confidence level.



	A	B	C	D	E	F
1	tmc_code	measurement_tstamp	speed	reference_speed	travel_time_minutes	confidence
2	110N04302	2017/3/3 0:00	48	53	0.23	0.87
3	110-04301	2017/3/3 0:00	47	55	0.03	0.87
4	110N04301	2017/3/3 0:00	47	55	0.29	0.87
5	110-04300	2017/3/3 0:00	46	55	0.19	0.85
6	110N04300	2017/3/3 0:00	46	55	0.1	0.85
7	110-04299	2017/3/3 0:00	49	55	0.06	0.82
8	110N04299	2017/3/3 0:00	49	55	0.43	0.82
9	110N04298	2017/3/3 0:00	54	59	1.15	0.78
10	110-04298	2017/3/3 0:00	54	59	0.1	0.78
11	110-04657	2017/3/3 0:00	64	65	0.51	0.73
12	110N04657	2017/3/3 0:00	64	65	0.53	0.73
13	110-04656	2017/3/3 0:00	62	65	0.07	0.74
14	110N04656	2017/3/3 0:00	62	65	0.55	0.74
15	110-04655	2017/3/3 0:00	63	65	0.11	0.74
16	110N04655	2017/3/3 0:00	63	65	0.63	0.74
17	110-04654	2017/3/3 0:00	64	65	0.32	0.75
18	110N04654	2017/3/3 0:00	64	65	0.52	0.75
19	110-04653	2017/3/3 0:00	66	65	0.87	0.73
20	110N04653	2017/3/3 0:00	66	65	0.65	0.73
21	110-16046	2017/3/3 0:00	70	65	0.05	0.73
22	110N16046	2017/3/3 0:00	70	65	0.57	0.73
23	110-04652	2017/3/3 0:00	68	65	0.32	0.73
24	110N04652	2017/3/3 0:00	68	65	0.43	0.73
25	110-04651	2017/3/3 0:00	59	56	0.44	0.73
26	110N04651	2017/3/3 0:00	59	56	0.7	0.73
27	110-04650	2017/3/3 0:00	59	55	0.2	0.73
28	110N04650	2017/3/3 0:00	59	55	0.03	0.73
29	110N04302	2017/3/3 0:01	43	53	0.25	0.85
30	110-04301	2017/3/3 0:01	42	55	0.03	0.88
31	110N04301	2017/3/3 0:01	42	55	0.32	0.88
32	110-04300	2017/3/3 0:01	46	55	0.19	0.85
33	110N04300	2017/3/3 0:01	46	55	0.1	0.85
34	110-04299	2017/3/3 0:01	49	55	0.06	0.82
35	110N04299	2017/3/3 0:01	49	55	0.43	0.82
36	110N04298	2017/3/3 0:01	54	59	1.15	0.78
37	110-04298	2017/3/3 0:01	54	59	0.1	0.78
38	110-04657	2017/3/3 0:01	64	65	0.51	0.73
39	110N04657	2017/3/3 0:01	64	65	0.53	0.73
40	110-04656	2017/3/3 0:01	62	65	0.07	0.74

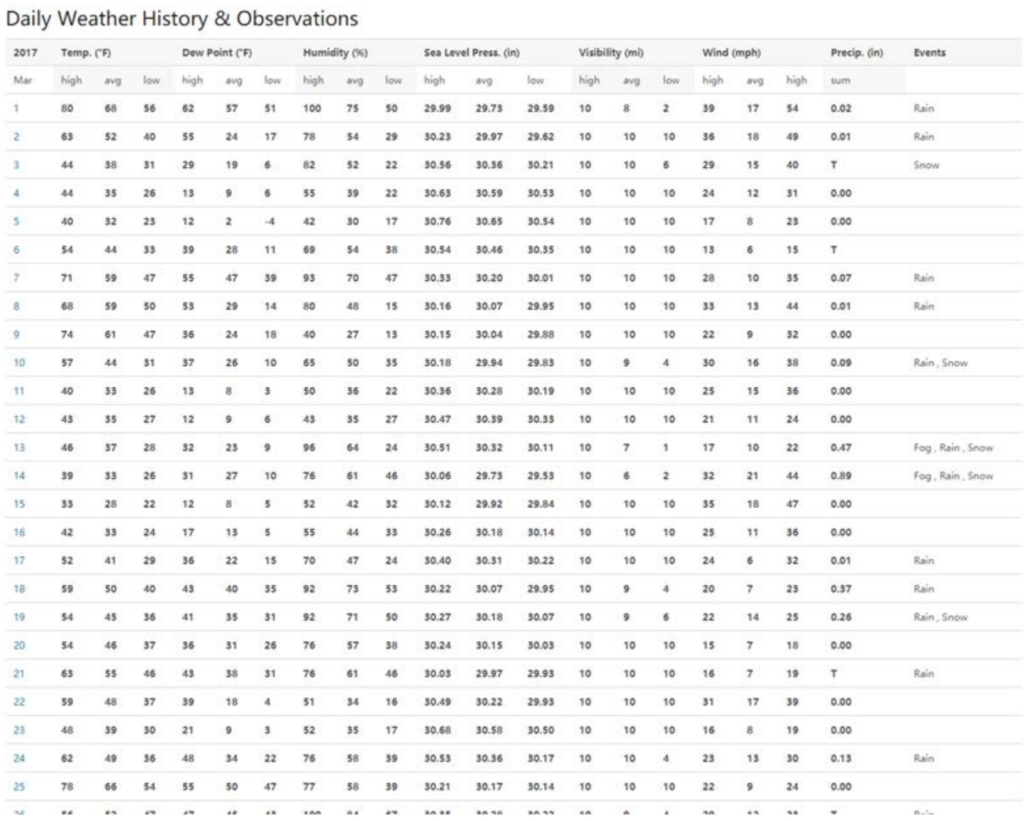
Figure 2.6 TMC traffic conditions

Weather Data Collection

Weather conditions are also main factors that would affect the transportation on the road (Li, 2012). We will collect weather conditions like temperature, visibility, precipitation, and weather events on each day during different time periods. Also, we will combine the weather data with the traffic data to analyze their effects on the travel time.

Historical weather data are collected from Weather Underground, which is a website where weather data are stored and shared. As is shown in Figure 2.7, we would use the weather data in March 2017 as an example for our analysis. In which the temperature has a broad range between 23°F and 80.1°F. There are 16 kinds of weather events in total, which are (1) Light Rain, (2) Overcast, (3) Mostly Cloudy, (4) Heavy Rain, (5) Scattered Clouds, (6) Partly Cloudy, (7) Unknown, (8) Light Snow, (9) Clear, (10) Light Freezing Rain, (11) Ice Pellets, (12) Light Ice Pellets, (13) Heavy Ice Pellets, (14) Rain, (15) Light Drizzle, and (16) Mist.

We used weather information from 3 different weather observation points in DMV area for different roads. Roads in Virginia will use the weather information retrieved from Annandale, roads in Washington, D.C. will use the weather data observed in Washington D.C., and the roads in Maryland will use the weather data obtained in College Park.



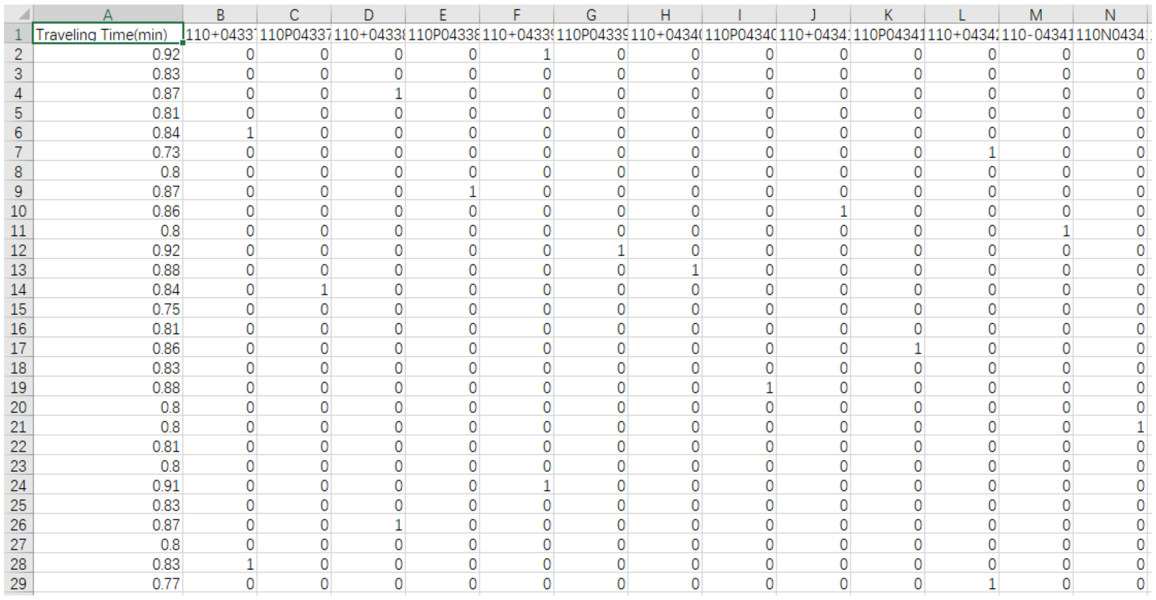
Daily Weather History & Observations

2017	Temp. (°F)			Dew Point (°F)			Humidity (%)			Sea Level Press. (in)			Visibility (mi)			Wind (mph)			Precip. (in)	Events
Mar	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	low	high	avg	high	sum	
1	80	68	56	62	57	51	100	75	50	29.99	29.73	29.59	10	8	2	39	17	54	0.02	Rain
2	63	52	40	55	24	17	78	54	29	30.23	29.97	29.62	10	10	10	36	18	49	0.01	Rain
3	44	38	31	29	19	6	82	52	22	30.56	30.36	30.21	10	10	6	29	15	40	T	Snow
4	44	35	26	13	9	6	55	39	22	30.63	30.59	30.53	10	10	10	24	12	31	0.00	
5	40	32	23	12	2	-4	42	30	17	30.76	30.65	30.54	10	10	10	17	8	23	0.00	
6	54	44	33	39	28	11	69	54	38	30.54	30.46	30.35	10	10	10	13	6	15	T	
7	71	59	47	55	47	39	93	70	47	30.33	30.20	30.01	10	10	10	28	10	35	0.07	Rain
8	68	59	50	53	29	14	80	48	15	30.16	30.07	29.95	10	10	10	33	13	44	0.01	Rain
9	74	61	47	36	24	18	40	27	13	30.15	30.04	29.88	10	10	10	22	9	32	0.00	
10	57	44	31	37	26	10	65	50	35	30.18	29.94	29.83	10	9	4	30	16	38	0.09	Rain , Snow
11	40	33	26	13	8	3	50	36	22	30.36	30.28	30.19	10	10	10	25	15	36	0.00	
12	43	35	27	12	9	6	43	35	27	30.47	30.39	30.33	10	10	10	21	11	24	0.00	
13	46	37	28	32	23	9	96	64	24	30.51	30.32	30.11	10	7	1	17	10	22	0.47	Fog , Rain , Snow
14	39	33	26	31	27	10	76	61	46	30.06	29.73	29.53	10	6	2	32	21	44	0.89	Fog , Rain , Snow
15	33	28	22	12	8	5	52	42	32	30.12	29.92	29.84	10	10	10	35	18	47	0.00	
16	42	33	24	17	13	5	55	44	33	30.26	30.18	30.14	10	10	10	25	11	36	0.00	
17	52	41	29	36	22	15	70	47	24	30.40	30.31	30.22	10	10	10	24	6	32	0.01	Rain
18	59	50	40	43	40	35	92	73	53	30.22	30.07	29.95	10	9	4	20	7	23	0.37	Rain
19	54	45	36	41	35	31	92	71	50	30.27	30.18	30.07	10	9	6	22	14	25	0.26	Rain , Snow
20	54	46	37	36	31	26	76	57	38	30.24	30.15	30.03	10	10	10	15	7	18	0.00	
21	63	55	46	43	38	31	76	61	46	30.03	29.97	29.93	10	10	10	16	7	19	T	Rain
22	59	48	37	39	18	4	51	34	16	30.49	30.22	29.93	10	10	10	31	17	39	0.00	
23	48	39	30	21	9	3	52	35	17	30.68	30.58	30.50	10	10	10	16	8	19	0.00	
24	62	49	36	48	34	22	76	58	39	30.53	30.36	30.17	10	10	4	23	13	30	0.13	Rain
25	78	66	54	55	50	47	77	58	39	30.21	30.17	30.14	10	10	10	22	9	24	0.00	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 2.7 Illustration of historical weather data

Data Selection, Solving, and Combination

We introduce dummy variables to represent some discrete variables like the TMC sets, the day of the week, and the weather events. Figure 2.8 shows how we use dummy variables to represent TMCs. When the value of a variable is 0, the coefficient of that variable would have no impact on the predicted values. On the other hand, its coefficient acts to alter the dependent variable, which is the travel time in our thesis, when it takes the value of 1. We also discretized the time in a day. We divided a day into 24 dummy variables, and each variable stood for one hour in a day.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Traveling Time(min)	110+04337	110P04337	110+04338	110P04338	110+04339	110P04339	110+04340	110P04340	110+04341	110P04341	110+04342	110-04341	110N0434
2	0.92	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0.83	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0.87	0	0	1	0	0	0	0	0	0	0	0	0	0
5	0.81	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0.84	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0.73	0	0	0	0	0	0	0	0	0	0	1	0	0
8	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0.87	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0.86	0	0	0	0	0	0	0	0	1	0	0	0	0
11	0.8	0	0	0	0	0	0	0	0	0	0	0	1	0
12	0.92	0	0	0	0	0	1	0	0	0	0	0	0	0
13	0.88	0	0	0	0	0	0	1	0	0	0	0	0	0
14	0.84	0	1	0	0	0	0	0	0	0	0	0	0	0
15	0.75	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0.81	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0.86	0	0	0	0	0	0	0	0	0	1	0	0	0
18	0.83	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0.88	0	0	0	0	0	0	0	1	0	0	0	0	0
20	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0.8	0	0	0	0	0	0	0	0	0	0	0	0	1
22	0.81	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0.91	0	0	0	0	1	0	0	0	0	0	0	0	0
25	0.83	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0.87	0	0	1	0	0	0	0	0	0	0	0	0	0
27	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0.83	1	0	0	0	0	0	0	0	0	0	0	0	0
29	0.77	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 2.8 Illustration of the dummy variables of the TMC sets

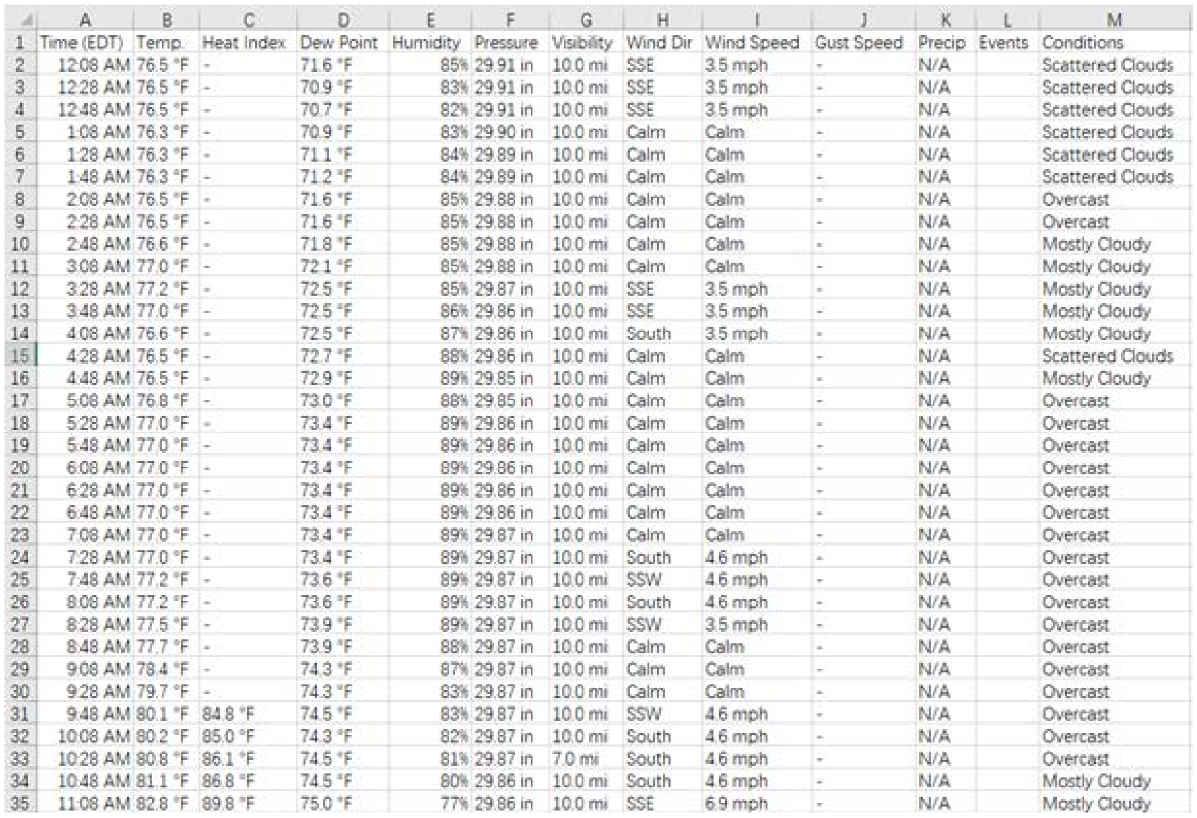
Figure 2.9 illustrates the weather data we retrieved from Weather Underground. We will delete the data to remove rows with missing records, such as the Heat Index.

Also, we introduce an intermediate variable called Timer, which is calculated by:

$$Timer = 60 \times Hour + Minute$$

The time interval for the weather conditions is 20 minutes. We will combine traffic conditions and weather information by adding the weather information at the end of each row of the traffic condition list if the timer of traffic dataset falls in the time interval of the weather information.

Additionally, we will take the average speed of the previous road piece and the average speed for the road piece after current road piece into consideration. These two factors are used to represent the effect of traffic flows on the road to the travel time.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Time (EDT)	Temp.	Heat Index	Dew Point	Humidity	Pressure	Visibility	Wind Dir	Wind Speed	Gust Speed	Precip	Events	Conditions
2	12:08 AM	76.5 °F	-	71.6 °F	85%	29.91 in	10.0 mi	SSE	3.5 mph	-	N/A		Scattered Clouds
3	12:28 AM	76.5 °F	-	70.9 °F	83%	29.91 in	10.0 mi	SSE	3.5 mph	-	N/A		Scattered Clouds
4	12:48 AM	76.5 °F	-	70.7 °F	82%	29.91 in	10.0 mi	SSE	3.5 mph	-	N/A		Scattered Clouds
5	1:08 AM	76.3 °F	-	70.9 °F	83%	29.90 in	10.0 mi	Calm	Calm	-	N/A		Scattered Clouds
6	1:28 AM	76.3 °F	-	71.1 °F	84%	29.89 in	10.0 mi	Calm	Calm	-	N/A		Scattered Clouds
7	1:48 AM	76.3 °F	-	71.2 °F	84%	29.89 in	10.0 mi	Calm	Calm	-	N/A		Scattered Clouds
8	2:08 AM	76.5 °F	-	71.6 °F	85%	29.88 in	10.0 mi	Calm	Calm	-	N/A		Overcast
9	2:28 AM	76.5 °F	-	71.6 °F	85%	29.88 in	10.0 mi	Calm	Calm	-	N/A		Overcast
10	2:48 AM	76.6 °F	-	71.8 °F	85%	29.88 in	10.0 mi	Calm	Calm	-	N/A		Mostly Cloudy
11	3:08 AM	77.0 °F	-	72.1 °F	85%	29.88 in	10.0 mi	Calm	Calm	-	N/A		Mostly Cloudy
12	3:28 AM	77.2 °F	-	72.5 °F	85%	29.87 in	10.0 mi	SSE	3.5 mph	-	N/A		Mostly Cloudy
13	3:48 AM	77.0 °F	-	72.5 °F	86%	29.86 in	10.0 mi	SSE	3.5 mph	-	N/A		Mostly Cloudy
14	4:08 AM	76.6 °F	-	72.5 °F	87%	29.86 in	10.0 mi	South	3.5 mph	-	N/A		Mostly Cloudy
15	4:28 AM	76.5 °F	-	72.7 °F	88%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Scattered Clouds
16	4:48 AM	76.5 °F	-	72.9 °F	89%	29.85 in	10.0 mi	Calm	Calm	-	N/A		Mostly Cloudy
17	5:08 AM	76.8 °F	-	73.0 °F	88%	29.85 in	10.0 mi	Calm	Calm	-	N/A		Overcast
18	5:28 AM	77.0 °F	-	73.4 °F	89%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Overcast
19	5:48 AM	77.0 °F	-	73.4 °F	89%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Overcast
20	6:08 AM	77.0 °F	-	73.4 °F	89%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Overcast
21	6:28 AM	77.0 °F	-	73.4 °F	89%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Overcast
22	6:48 AM	77.0 °F	-	73.4 °F	89%	29.86 in	10.0 mi	Calm	Calm	-	N/A		Overcast
23	7:08 AM	77.0 °F	-	73.4 °F	89%	29.87 in	10.0 mi	Calm	Calm	-	N/A		Overcast
24	7:28 AM	77.0 °F	-	73.4 °F	89%	29.87 in	10.0 mi	South	4.6 mph	-	N/A		Overcast
25	7:48 AM	77.2 °F	-	73.6 °F	89%	29.87 in	10.0 mi	SSW	4.6 mph	-	N/A		Overcast
26	8:08 AM	77.2 °F	-	73.6 °F	89%	29.87 in	10.0 mi	South	4.6 mph	-	N/A		Overcast
27	8:28 AM	77.5 °F	-	73.9 °F	89%	29.87 in	10.0 mi	SSW	3.5 mph	-	N/A		Overcast
28	8:48 AM	77.7 °F	-	73.9 °F	88%	29.87 in	10.0 mi	Calm	Calm	-	N/A		Overcast
29	9:08 AM	78.4 °F	-	74.3 °F	87%	29.87 in	10.0 mi	Calm	Calm	-	N/A		Overcast
30	9:28 AM	79.7 °F	-	74.3 °F	83%	29.87 in	10.0 mi	Calm	Calm	-	N/A		Overcast
31	9:48 AM	80.1 °F	84.8 °F	74.5 °F	83%	29.87 in	10.0 mi	SSW	4.6 mph	-	N/A		Overcast
32	10:08 AM	80.2 °F	85.0 °F	74.3 °F	82%	29.87 in	10.0 mi	South	4.6 mph	-	N/A		Overcast
33	10:28 AM	80.8 °F	86.1 °F	74.5 °F	81%	29.87 in	7.0 mi	South	4.6 mph	-	N/A		Overcast
34	10:48 AM	81.1 °F	86.8 °F	74.5 °F	80%	29.86 in	10.0 mi	South	4.6 mph	-	N/A		Mostly Cloudy
35	11:08 AM	82.8 °F	89.8 °F	75.0 °F	77%	29.86 in	10.0 mi	SSE	6.9 mph	-	N/A		Mostly Cloudy

Figure 2.9 Illustration of the weather information

Statistical Models

Linear Regression Model

The data, which we obtained for the road I-395 in March 2017, have multiple independent variables, including 55 dummy variables for the road pieces, 7 dummy variables for the day of the week, 24 dummy variables for the time of the day, 16 dummy variables for different weather events, and 10 continuous variables for 10 weather parameters such as temperature and wind speed. We are going to use these factors to find their influence on the dependent variable, which is the travel time.

Multiple linear regression is our first model introduced to show the relationship between multiple explanatory variables and a response variable. We are going to fit the observed data to a linear equation, which can be represented as:

$$\mu_y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

In the equation, μ_y is the mean of the observed response values, which are represented as y_i. $x_i = (x_1, x_2, x_3, \cdots, x_n)$ are n independent explanatory variables, and they are the 102 independent variables for the road I-395. The values $\beta_i = (\beta_1, \beta_2, \beta_3, \cdots, \beta_n)$ are the corresponding estimated coefficients to the independent variables and β_0 is the intercept. The residual δ_i is defined as the deviation between the observed travel time and their calculated mean:

$$\delta_i = y_i - \mu_y$$

We will use the ordinary least squares method to estimate the coefficients which can minimize the sum of squared residuals:

$$\hat{\beta} = (\beta_0, \beta_i) = argmin_{\beta_0, \beta_1} \sum_{i=1}^{N} {\delta_i}^2 = argmin_{\beta_0, \beta_1} \sum_{i=1}^{N} (y_i - \mu_y)^2$$

We will use the lm() function in R to calculate the linear regression model for our model. The Linear Regression Model we got for the travel time of I-395 is attached in Appendix A.

Generalized Linear Model

In linear regression, the dependent variable is a linear combination of the input components. Which means ordinary linear regression assumes that the response variables have an error distribution that is normally distributed. For those models that do not have continuous real-valued response variables, ordinary linear regression might not fit the model very well.

Generalized linear model is a more flexible way of fitting the model. It allows the response variables whose distribution belongs to some exponential family, e.g., it can be Exponential, Gamma, and Binomial (Nelder & Wedderburn, 2012). The main idea is that generalized linear model introduces a link function g to the linear regression:

$$\mu_y = g^{-1}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)$$

The coefficients in the generalized linear model, which are β_i, are usually estimated by maximum likelihood, maximum quasi-likelihood, or Bayesian techniques (Das & Dey, 2007).

Table 2.3 shows us some common distributions and their link functions in the generalized linear model.

Table 2.3 Common distributions and canonical link functions

Distribution	Definition Domain	Link name	Link Function $X\beta = g(\mu)$	Mean Function
Normal	Real: $(-\infty, +\infty)$	Identity	$X\beta = \mu$	$\mu = X\beta$
Exponential	Real: $(0, +\infty)$	Inverse	$X\beta = \mu^{-1}$	$\mu = (X\beta)^{-1}$
Gamma				
Inverse Gaussian	Real: $(0, +\infty)$	Inverse Squared	$X\beta = \mu^{-2}$	$\mu = (X\beta)^{-1/2}$
Poisson	Integer: 0, 1, 2, …	Log	$X\beta = \ln(\mu)$	$\mu = \exp(X\beta)$

Figure 2.10 shows us the histogram of the dependent variable, which is the travel time in our thesis, from the historical data set. It gives us a brief view of the distribution of the data. We can easily tell that the data are not normally distributed.

The Q-Q plot shown in Figure 2.16 also supports the evidence that the data set is right skewed, which we will discuss later.

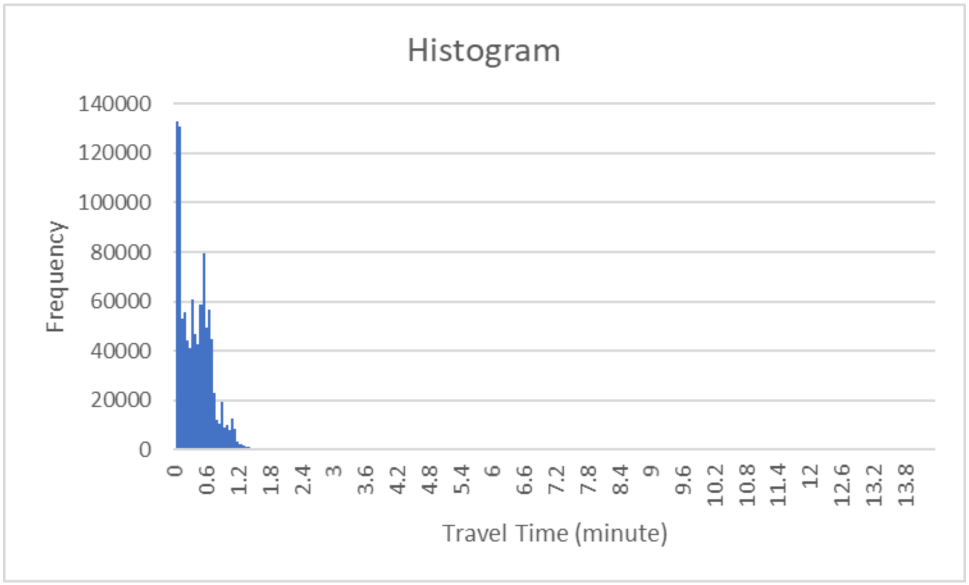


Figure 2.10 Histogram of the dependent variable

Gamma distribution with the “log” link function, Gamma distribution with the “inverse” link function, and the Inverse Gaussian distribution with the “inverse squared” link function are typically used to solve skewed data (Ngo, 2016). Poisson distribution is used for cases that have discrete variables, and it is not suitable for our project because the travel time is a set of continuous numbers. Therefore, we are going to compare the first three types of generalized linear models mentioned in this paragraph to see which fits our model better.

We use the glm() function in R to calculate generalized linear models for our thesis. The model for the travel time of I-395 is attached in Appendix C.

LASSO Regression Model

Although the least square method yields an unbiased estimator, it cannot perform very well under multicollinearity in the independent variables. In the situation of collinearity, the multiple regression may change erratically in response to small changes in the model or the data. It will also affect the calculations regarding individual predictors. Therefore, we want to find a way to rule out some factors that are less significant to the prediction of the response variable.

Least absolute shrinkage and selection operator (LASSO) is a widely used regression analysis method (Tibshirani, 1996). The main feature of LASSO regression is that it adds an L-1 norm into the ordinary linear regression model. The estimator of LASSO can be represented as:

$$\hat{\beta}^{lasso} = argmin\left\{\sum_{i=1}^{N}(y_i - \mu_{y_i})^2 + \lambda\sum_{j=1}^{p}\left|\beta_j\right|\right\}$$

The characteristic of LASSO regression is that it can perform the variable selection and adjust the complexity of the model while fitting the generalized linear regression. The basic idea of LASSO is to force the sum of the absolute value of the regression coefficients to be smaller. Some certain coefficients would be set to zero, which means those factors would be picked out of the model. Take bivariate regression model as an example, as is shown in Figure 2.11, the shadow area is formed by $\lambda\sum_{j=1}^{p}\left|\beta_j\right|$ and we can adjust its size by changing the value of λ.

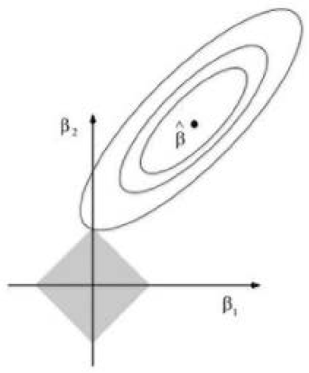


Figure 2.11 LASSO Regression

In the case presented in Figure 2.11, β_1 will be set to 0 first while β_2 remains in the regression model. Using this method, we can effectively choose a simpler model that only include the most significant coefficients.

We are going to use the glmnet() package in R to perform the LASSO regression on our data. The LASSO Regression Model we calculated for the travel time of I-395 is attached in Appendix B.

Sure Independence Screening

LASSO regression is a widely used approach for variable and model selection. However, it has some limitations that it may not perform well in high dimensional settings, which means when we have a massive number of independent variables to select from, due to the simultaneous challenges of computational expediency, statistical accuracy and algorithmic stability (Fan & Song, 2010).

Sure independence screening (SIS) is an independent screening framework by ranking the marginal correlations (Fan & Song, 2017). We assume that the p_n independent variables in our true regression model is:

$$\mathrm{M} = \{1 \le j \le p_n : \beta_j \ne 0\}, \text{where } \beta_j = (\beta_1, \beta_2, \cdots, \beta_{p_n})$$

Then the maximum marginal likelihood estimator would be defined as:

$$\hat{\beta}_j^M = \left(\hat{\beta}_{j,0}^M, \hat{\beta}_{j,1}^M\right) = argmax_{\beta_0,\beta_1} L(\beta_{j,0} + \beta_{j,1}x_j, y_i)$$

In the equation, $L(\beta_{j,0} + \beta_{j,1}X_j, Y)$ is called the likelihood function. It represents the likelihood that $\beta_{j,0} + \beta_{j,1}x_j$ equals to y_i when the parameters β_0 and β_1 are given. Therefore, the maximum marginal likelihood estimator is trying to find β_0 and β_1 that can make the possibility that $\beta_{j,0} + \beta_{j,1}x_j$ equals to y_i the biggest.

Thus, we can select the most significant variables by changing the threshold γ_n, which is a predefined value to screen out some factors whose coefficients are smaller than it:

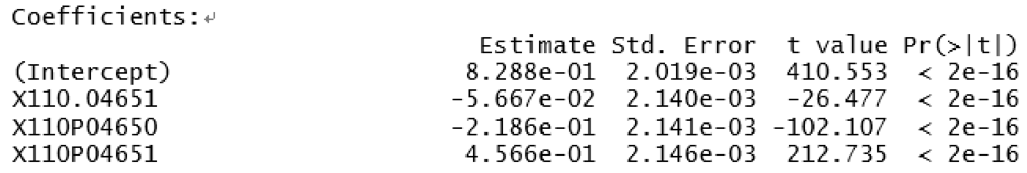
$$\widehat{\mathrm{M}}_{\gamma_n} = \left\{1 \le j \le p_n : \left|\hat{\beta}_j^M\right| \ge \gamma_n\right\}$$

By changing the threshold value and rule out the less significant factors by the ranking, we can compare the influence of different variables to the model and decide which variables would be remained in our model. We will use cross-validation to compare the influence of different threshold values and choose the best result that has the least mean square error between the observed travel time and the result we get from the regression model. Details will be covered in section 2.3

We used R to write the code for sure independent screening. For each factor in the regression model, we calculated its generalized linear model to fit the travel time. Then, we calculated $\left|\hat{\beta}_j^M\right|$ for each of them and used cross-validation to compare the mean square errors between the observed value and calculated value when using different threshold values. We will explain the process in detail later in the model comparison section. Results of mean square errors for different threshold values can be checked in Appendix D.

__Model Explanation__

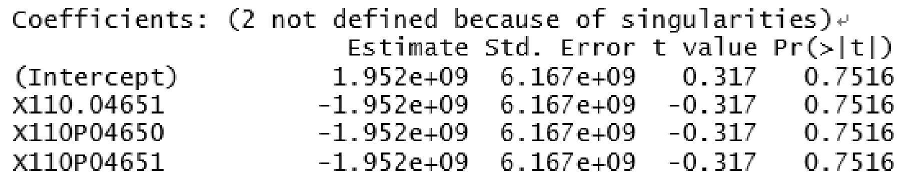
Among all the models we calculated, all TMCs are considered significant to the prediction of the travel time. Figure 2.12 gives us a part of the result from the LASSO regression model. In the figure, two of the TMCs are given negative estimated coefficients, and one of them is given a positive value. Note that we are estimating the effects of the independent variables on the travel time. Therefore, if the estimated coefficient has a positive value, that means the factor would increase our travel time. On the contrary, the factor would reduce the travel time if it is negative.



```
Coefficients:↵
                          Estimate Std. Error  t value Pr(>|t|)
(Intercept)              8.288e-01  2.019e-03  410.553  < 2e-16
X110.04651              -5.667e-02  2.140e-03  -26.477  < 2e-16
X110P04650              -2.186e-01  2.141e-03 -102.107  < 2e-16
X110P04651               4.566e-01  2.146e-03  212.735  < 2e-16
```

Figure 2.12 Partial result of the LASSO regression for TMCs

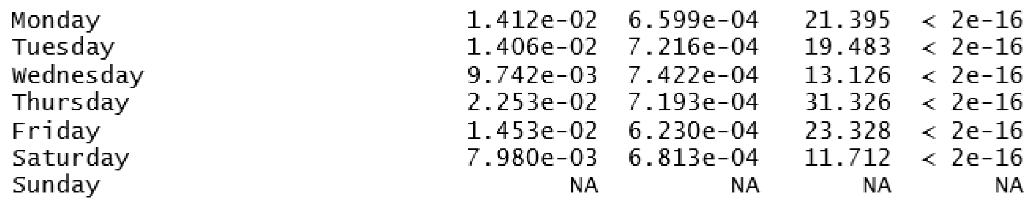
These models give us different estimated coefficients of each TMC, but both linear model and LASSO regression believe that the TMCs are the most significant factors of all. The orders of magnitude for these coefficients are usually -1 or -2. While the coefficients for other factors, such as time and weather, are usually 10 times smaller. However, generalized linear model and sure independence screening believe the TMCs are not significant as is shown in Figure 2.13.



```
Coefficients: (2 not defined because of singularities)↵
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     1.952e+09  6.167e+09   0.317   0.7516
X110.04651     -1.952e+09  6.167e+09  -0.317   0.7516
X110P04650     -1.952e+09  6.167e+09  -0.317   0.7516
X110P04651     -1.952e+09  6.167e+09  -0.317   0.7516
```

Figure 2.13 Partial result of the sure independence screening for TMCs

Take Figure 2.14 as an example, as we can see from the estimated coefficients, the coefficient for the factor "Sunday" is marked as "NA". These are because of singularity. When we define dummy variables for each day, we give it a value 1 if it is that day. Otherwise, the value would be 0. It causes the seven factors to be linearly dependent. If the factors from "Monday" to "Saturday" are all 0s, the value of "Sunday" will certainly be 1. In this case, R would not define the coefficient for "Sunday". The coefficient of "Sunday" will instead be set as the average value and the coefficients for other six factors are estimated compared to "Sunday". For example, when the factors from "Monday" to "Saturday" are 0s, the regression model will calculate the travel time for "Sunday". When the factor "Monday" takes the value as 1, the model will calculate the travel time for Mondays by finding how much time it would spend more than on Sundays.



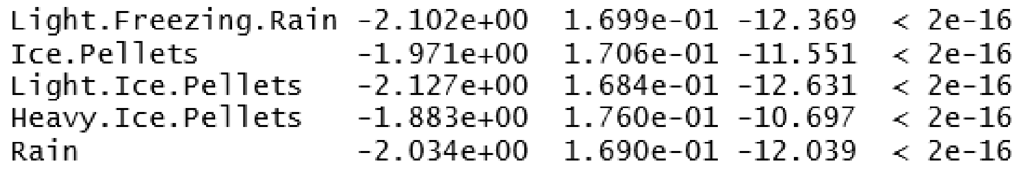
```
Monday                          1.412e-02  6.599e-04   21.395  < 2e-16
Tuesday                         1.406e-02  7.216e-04   19.483  < 2e-16
Wednesday                       9.742e-03  7.422e-04   13.126  < 2e-16
Thursday                        2.253e-02  7.193e-04   31.326  < 2e-16
Friday                          1.453e-02  6.230e-04   23.328  < 2e-16
Saturday                        7.980e-03  6.813e-04   11.712  < 2e-16
Sunday                                 NA         NA       NA       NA
```

Figure 2.14 Singularity in the linear model

The linear regression model and LASSO regression consider the traveling time of I-395 on Thursdays are longer than other days in a week. The traveling time of I-395 on Sundays is shorter than other days. We can suggest that people would use the road more often on Thursdays, and thus the traveling time would become longer due to the heavy traffic. On the contrary, people would use I-395 less on Sundays compared to other days. However, the coefficients for other models are hard to explain. Generalized linear model and sure independence screening believe the travel time on Sundays is the longest, which is contrary to reality.

Sure independence screening thinks the average speed on the adjacent TMCs is less significant. However, the LASSO model tells us the average speed on the adjacent TMCs would affect the travel time. If the average speed on adjacent TMCs is fast, the travel time on the TMC will become shorter.

As we can see from the remaining coefficients for factors like weather and time, LASSO regression gives us the best explanation for their effect on the travel time. Among the 24 hours of a day, there is less traffic on the road between 2:00 AM and 3:00 AM. On the contrary, the traffic is really heavy between 5:00 AM and 10:00 AM, as well as 6:00 PM and 7:00 PM. Extreme weather conditions would also affect the travel time. For example, the travel time would become longer if it rains heavily. However, some other mild weather conditions would not that much affect the travel time. For example, if temperature and wind speed remains within a reasonable range, the effect is considerable because their coefficients have an order of magnitude as ten to the negative three.



```
Light.Freezing.Rain -2.102e+00  1.699e-01 -12.369  < 2e-16
Ice.Pellets         -1.971e+00  1.706e-01 -11.551  < 2e-16
Light.Ice.Pellets   -2.127e+00  1.684e-01 -12.631  < 2e-16
Heavy.Ice.Pellets   -1.883e+00  1.760e-01 -10.697  < 2e-16
Rain                -2.034e+00  1.690e-01 -12.039  < 2e-16
```

Figure 2.15 Part of estimated coefficients for weather conditions by SIS

Figure 2.15 shows us the partial result of estimated coefficients by the sure independence screening. The model thinks these weather conditions will reduce the travel time, which doesn’t make much sense.

As a conclusion, the LASSO regression gives us the model that is easiest to explain and is close to reality.

Model Comparison

Q-Q Plots

Q-Q (quantile-quantile) plot is a statistical method to compare two different probability distributions with each other. To draw a Q-Q plot, we need first to choose the set of intervals for the quantiles. A point (x, y) on the plot corresponds to one of the quantiles of the second distribution (y-coordinate) plotted against the same quantile of the first distribution (x-coordinate). If the two distributions being compared are identical, the Q–Q plot follows the 45°-line y = x. If the two distributions are the same after linearly transforming the values in one of the distributions, then the Q–Q plot would follow some line, but not necessarily the line y = x.

In our linear regression model, we assume our data was normally distributed. Under this hypothesis, we are drawing a Q-Q plot to compare the distribution of the data for the travel time with the normal distribution. As is shown in Figure 2.16, the data points are almost a straight line on the left side of origin point on the x-coordinate, but its tail goes up at the right side. The Q-Q plot shows us that the data we have is heavily right-skewed. Therefore, it is unlikely that they are normally distributed.

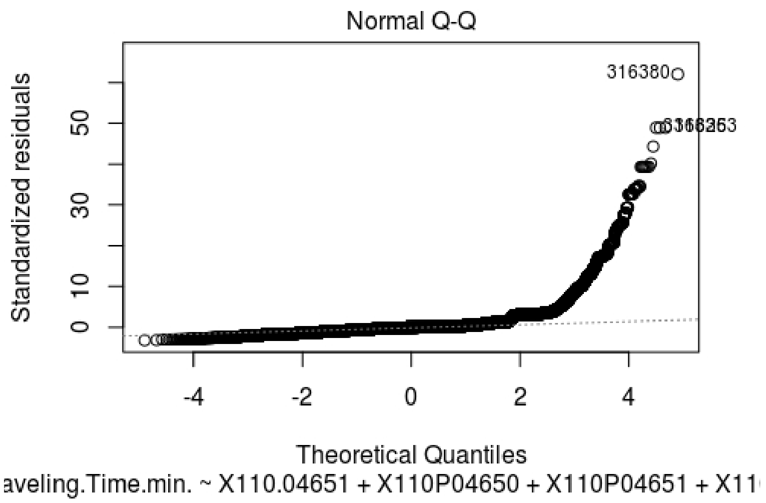


Figure 2.16 Q-Q Plot for Normal Distribution

Since we have found that the data of travel time is right-skewed, we would like to try if we add a log function to adjust our independent variables, which is usually used for adjusting right-skewed data (Feng, et al., 2014), would improve our result. As is shown in Figure 2.17, adding the log function does improve the model because it reduced the range of the standardized residuals from (-3,60) to (-4,6).

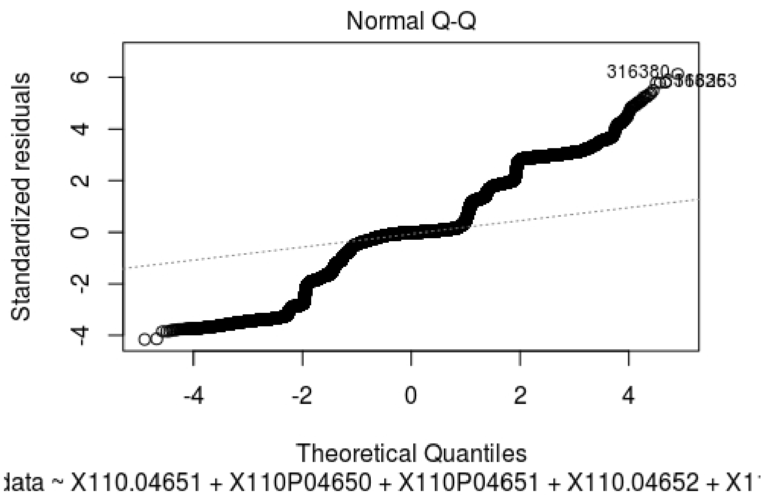


Figure 2.17 Q-Q Plot for Normal Distribution after the log-transformation

We now test the Gamma distribution with the “log” link function. As is shown in Figure 2.18, the standardized residual lies between (-10, 20).

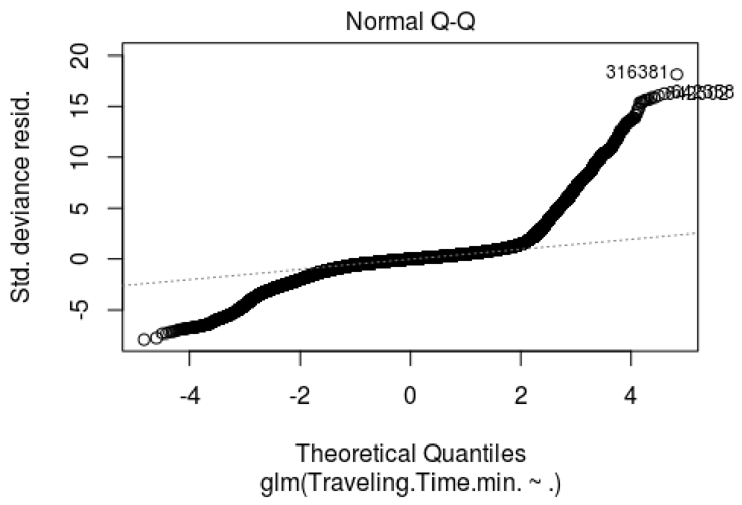


Figure 2.18 Q-Q plot for GLM with Gamma distribution and log link function

Figure 2.19 shows us the Q-Q plot for Gamma distribution with the "inverse" link function. The standardized residual lies between (-10,10).

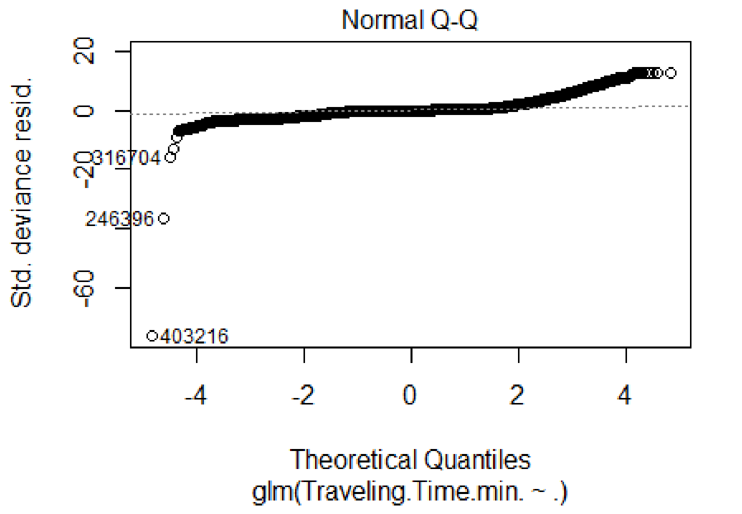


Figure 2.19 Q-Q plot for GLM with Gamma distribution and inverse link function

We also considered the inverse Gaussian distribution with "Inverse Square" link function, but the generalized linear model shows that it considered all the factors in the regression model insignificant. The result can be checked in Appendix E. This indicates that the model is not suitable for our data and we are not going to do further evaluations for this model.

From the Q-Q plots, we can conclude that our data is right-skewed and linear regression model might not be suitable for our traffic model. Also, we can rule out the generalized linear model that has inverse Gaussian distribution with “inverse square” link function. In the next step, we are going to use cross-validation method to fit the data and choose the best model for the remaining candidates.

Cross-Validation

Cross-Validation is a data-driven way to evaluate and compare different regression models we created. The basic idea of cross-validation is to divide the data we have into two parts. One is used for training the regression model, and the other is used as a testing set to evaluate how our model fits the data (Refaeilzadeh, Tang, & Liu, 2009). In our thesis, we mainly used two different types of cross-validation, which are (1) holdout cross-validation and (2) k-fold cross-validation.

The first type of cross-validation is called holdout cross-validation. When we have a large quantity of data, holdout cross-validation is the easiest and a time-saving method to assess our model. We can directly divide the original set into two parts, which are the training set and the test set as is shown in Figure 2.20. For example, we can suggest one of them takes 80% of the original data, and the other would take the remaining 20%. We are going to train our regression model using the data from the training set. After we have got the regression model, we would use it to fit the data in the test set. By evaluating the model with the test set, we can see how well our model performs. Although the holdout cross-validation is a relatively easy and fast way to test the model, it has some issues. For example, not all of our data set are used to train the model, which may cause our model less representative on the data in the test set.

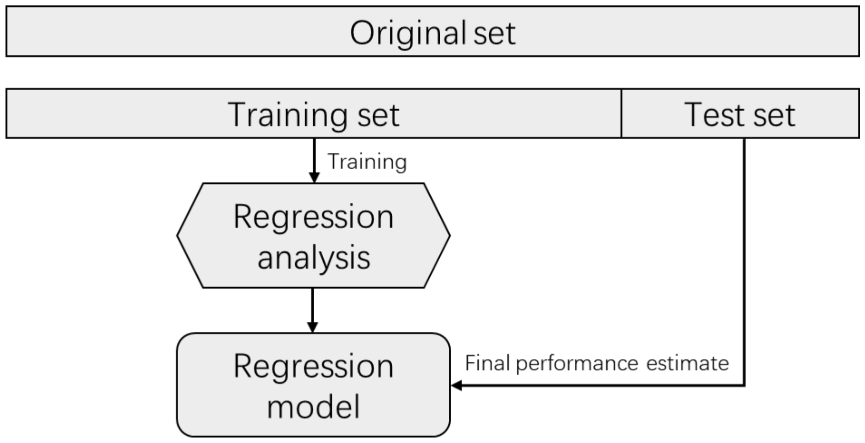


Figure 2.20 Holdout cross-validation

The second type of cross-validation is called k-fold cross-validation. In this method, we divide the whole original set into K roughly equal parts. For each $k = 1, 2, 3, \cdots,$ K. Each time, we choose the kth fold as the test fold to evaluate the model trained by the other K-1 folds. After we have completed all the iterations, we would use the mean value of all the mean square errors to represent the deviation of the whole model. In K-fold cross validation, all data are used both as training data and testing data, which successfully avoid the problem that holdout cross-validation has. But the process of K-fold cross-validation would take a long time compared to holdout cross-validation. Because in K-fold cross validation, each fold of data is trained $k - 1$ times and tested once. Especially in our thesis, we are going to test each λ value in LASSO regression and different γ_n values in sure independence screening, the whole process would spend us a lot of time. Figure 2.21 gives us an example that how the 5-fold cross-validation works.

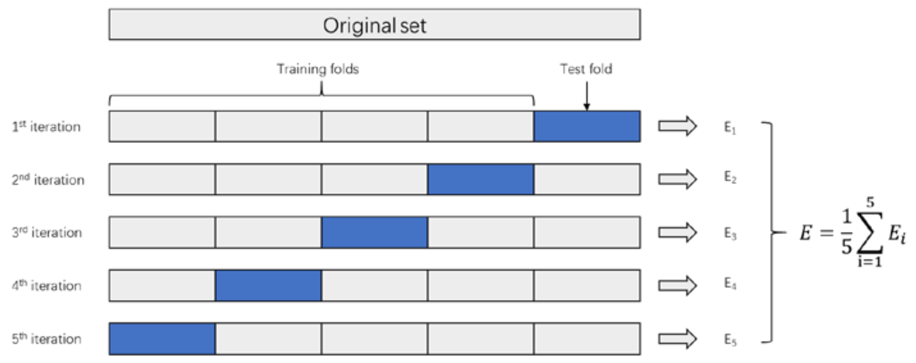


Figure 2.21 5-fold cross-validation

We will use both these methods to evaluate the accuracy of different models. We will use 5-fold cross-validation to evaluate the linear regression model because computing the mean squared errors for the model is relatively fast. However, it would take us a lot of time to perform 5-fold cross-validation on LASSO regression and sure independence screening. In order to save time, during the evaluation of LASSO regression models and sure independent screening, we will first use holdout cross-validation to rule out some λ values and γ_n values that give us a large mean square error. For example, when selecting the threshold value for sure independence screening, we first perform the holdout cross-validation for calculating mean squared errors of different threshold values is sure independence screening. From Figure 2.22 we can easily rule out the first two points that have high mean squared errors. For those top-ranking threshold values, we will perform another 5-fold cross-validation to figure out which one can give us the best fit result.

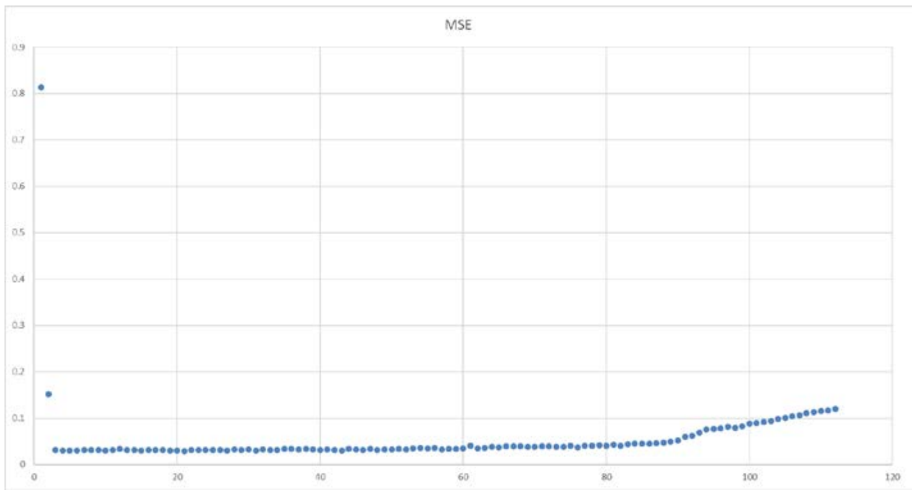


Figure 2.22 Holdout cross-validation results

Result

The mean squared error (MSE) of travel time in the linear regression model for I-395 is 0.02269. The MSE of travel time in LASSO regression is 0.02274. The MSE of travel time in SIS is 0.031. However, the MSE for the GLM model (Gamma distribution and “inverse” link function) is 0.814, which is higher than other three models. We can tell that the MSE for LM and LASSO are very close and small. Also, SIS successfully screens out some irrelevant features and improve the GLM model.

Since LASSO regression reduces the size of LM and its coefficients are easier to explain, we conclude that LASSO regression helps us get the best fit model among all these methods. We will use the model calculated by LASSO regression and move to the next step of our thesis, which is to build a route planning method that can predict the traffic conditions of different roads in the future and avoid the roads that might be crowded.

Chapter 3: Predictive Route Planning Algorithms

This chapter discusses the predictive route planning methods to find the fastest travel path using the regression model obtained in the previous chapter. In the first section, we will give an initial idea, which is called the random path method, to solve the route planning problem. Next, we are going to introduce graph theory and talk about some common graph traversal methods that are used to search for the path between the origin and the destination. We will also apply the graph traversal method to our route planning problem to find the fastest path. In section 3, we are going to introduce Approximate Dynamic Programming, which is also known as reinforcement learning in the field of artificial intelligence. We will use deterministic Approximate Dynamic Programming to solve our predictive route planning problem. Finally, we are going to use an example to illustrate how these algorithms give us different optimal paths at different time periods.

Random Path Method

Introduction

When talking about route planning problems to find the fastest path, a natural way is to find all possible paths from our origin to the destination and record the travel time for each of them. Then, we are going to compare the travel time and choose the fastest path.

Implementation of the Random Path Method

The main idea is to take multiple trials in our road-network randomly and set a temporary variable to store the fastest trial through iterations.

We are going to use t to represent the time of the day and D means a day of the week. We will take Tr as the traffic condition at the TMC we are on and use W as the weather condition. Both of these two variables are changing with t. $\mathrm{T(TMC, t, Tr, W)}$ represents the travel time for the TMC we are at. We are going to use the LASSO regression model mentioned in Chapter 2 to calculate the travel time for each road piece. $\mathrm{LM(D, t, Tr(t), W(t))}$ represents the predicted value of LASSO model.

$$\mathrm{T(TMC, t, Tr, W) = LM(D, t, Tr(t), W(t))}$$

The Pseudocode for the random path method is:

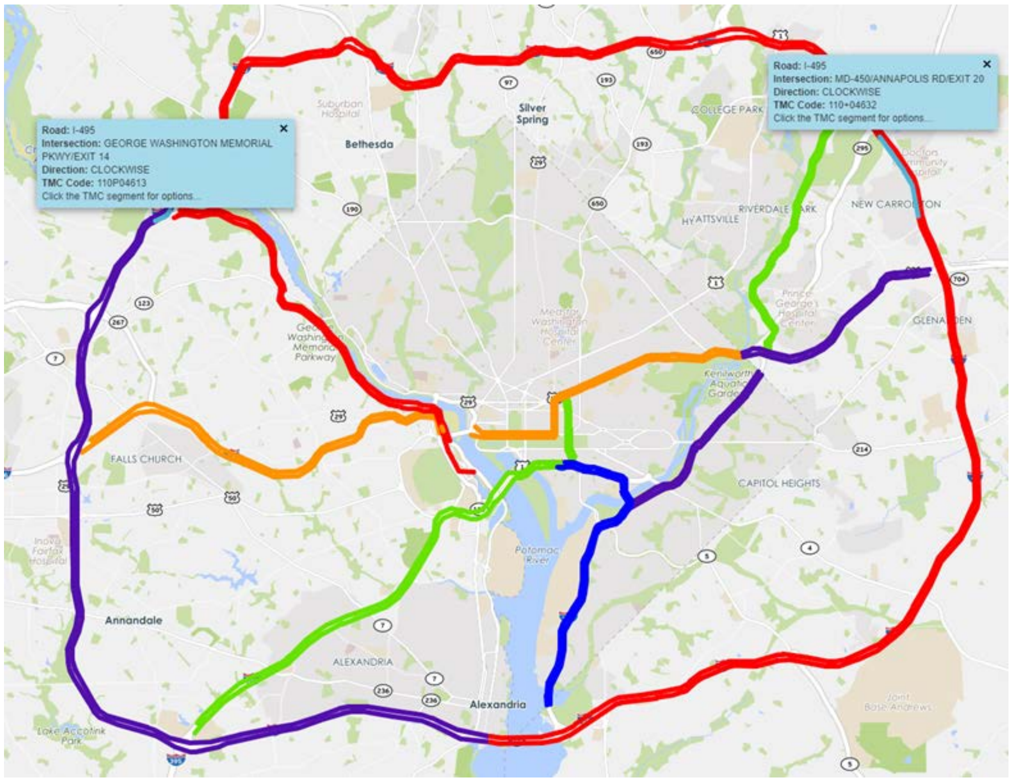
```
Step 0. Initialization

    Step 0a.    Set the starting point as our initial state $S_{STR}$, which is the
                starting TMC
                Set the destination as our goal state $S_{ENR}$, which is the ending
                TMC
    Step 0b.    Initialize the route list $L = [S_{STR}]$    # Used to store the path in
                each trial
                Initialize $T$    # Accumulated travel time from the origin
                Initialize $S_t = S_{STR}$    # Current TMC we are at
                Initialize $L_b = [\ ]$    # List to store the best path
                Initialize $T_b = 9999$    # Travel time on the best path
    Step 0c.    Read in the adjacency matrix $\mathrm{M}_a$ of the road pieces
```

```
        Step 0d.        Set n = 1, which is the number of the trial

        Step 0e.        Read in the starting time, give it to $\mathrm{T}_{S_{STR}}$

        Step 0f.        Set N=100 (for example), which is the total trials we want to
                        evaluate

Step 1. While $n < N$

        Step 1a. While $S_t \neq S_{ENR}$

                Use the regression model $CalTime()$ function to compute the travel
                time of $S_t$

                        $\mathrm{T(TMC, t, Tr, W) = LM(D, t, Tr(t), W(t))}$

                Choose the next TMC in our trial using $\mathrm{M}_a$

                        $\mathrm{S}_{t+1}^{n} = \mathrm{M}_a(\mathrm{S}_t{}^{n})$

                Add next TMC into $\mathrm{L}^n$

                        $L^n = L^n + S_{t+1}^n$

                Update current TMC we are at

                        $S_t^n = S_{t+1}^n$

                Update accumulated travel time from the origin

                        $T = T + C_{\mathrm{S}_t}{}^{n}$

        Step 1b. If $T < T_b$
        $T_b = T$        # Update the best travel time
        $L_b = L$        # Update the best route
```

Step 2. Output the results, T_b and L_b

Result

We are going to use an example to show the result of our algorithm. We choose the TMC, whose ID is "110P04613", as the origin, and we use the TMC, whose ID is "110+04632", as the destination. The locations of these two TMCs are shown in Figure 3.1.

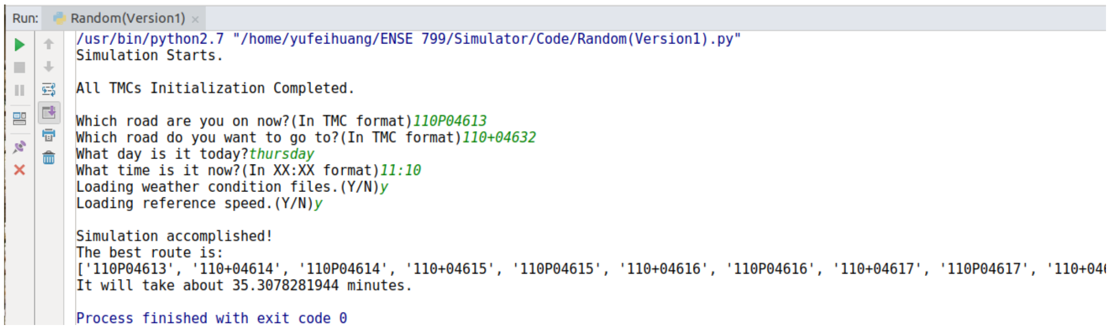


Road: I-495
Intersection: GEORGE WASHINGTON MEMORIAL PKWY/EXIT 14
Direction: CLOCKWISE
TMC Code: 110P04613
Click the TMC segment for options...

Road: I-495
Intersection: MD-450/ANNAPOLIS RD/EXIT 20
Direction: CLOCKWISE
TMC Code: 110+04632
Click the TMC segment for options...

Figure 3.1 Selected start TMC and end TMC in the road network

We are going to set the day of the week to be Thursday and the time of the day to be 11:10 AM. Figure 3.2 shows us the inputs and outputs of the program.



```
/usr/bin/python2.7 "/home/yufeihuang/ENSE 799/Simulator/Code/Random(Version1).py"
Simulation Starts.

All TMCs Initialization Completed.

Which road are you on now?(In TMC format)110P04613
Which road do you want to go to?(In TMC format)110+04632
What day is it today?thursday
What time is it now?(In XX:XX format)11:10
Loading weather condition files.(Y/N)y
Loading reference speed.(Y/N)y

Simulation accomplished!
The best route is:
['110P04613', '110+04614', '110P04614', '110+04615', '110P04615', '110+04616', '110P04616', '110+04617', '110P04617', '110+04
It will take about 35.3078281944 minutes.

Process finished with exit code 0
```

Figure 3.2 Inputs and outputs of the random path method

The random path is an easy method to accomplish, but it is obvious that it has many problems such as low efficiency. Next, we are going to explore some more intelligent algorithms to solve our route planning problem.

Graph Traversal Method

Introduction to Graph Theory

Graph theory has been applied to many fields nowadays. We would encounter problems like navigation, network analysis, and program flows which all would use graphs to simplify the systems and build up models for the systems.

A graph consists of a nonempty set $\mathrm{V} = (v_1, v_2, \cdots, v_n)$, a possibly empty set $\mathrm{E} = (e_1, e_2, \cdots, e_n)$ and an incidence mapping φ associated with the graph. The elements of V and E are called the vertices (nodes) and edges (arcs, links) of the graph (Smith, 2003).

$$\mathrm{G} = (\mathrm{V}, \mathrm{E})$$

A number c_{ij} may be associated with an arc (x_i, x_j), which refers to a directed arc from x_i to x_j. These numbers are called weights. A graph with weights is then called an arc-weighted graph. Weight v_i may sometimes be associated with a vertex

x_i and the resulting graph is then called vertex-weighted. Consider a path μ represented by the sequence of arcs $(a_1, a_2, \cdots, a_n)$, the length (cost) of the path $\ell(\mu)$ is taken to be the sum of the arc weights on the arcs in μ.

$$\ell(\mu) = \sum_{(x_i, x_j)\ in\ \mu} c_{ij}$$

The shortest path problem is one of the fundamental questions in graph theory. Our project can be classified into the shortest path problem with determined start point and destination. We are going to use the vertex-weighted graph model to simplify our road network. The nodes represent all the TMCs in our road network. The weight of each node is the travel time of that road piece, which changes with the time that we traveled to that road piece.

After we have defined our graph model of the road network, we need to develop a method that can give us the fastest travel route between the given origin and destination. Dijkstra algorithm is a frequently used method to solve shortest path problems with determined start and end points. Next, we are going to talk about how this method works and how we can apply it to our route planning problem.

Dijkstra's Algorithm

In an undirected vertex-weighted graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$, suppose that each node $\mathrm{V[i]}$ has the cost of w[j], and we want to calculate the shortest path between two nodes. Dijkstra's algorithm is a suitable method to solve such a problem by looking for the shortest path from a source node to all other vertices. It picks the unvisited vertex with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if the new distance is smaller (Dijkstra, 1959). The principle of Dijkstra’s algorithm can be represented as the flowchart in Figure 3.3:

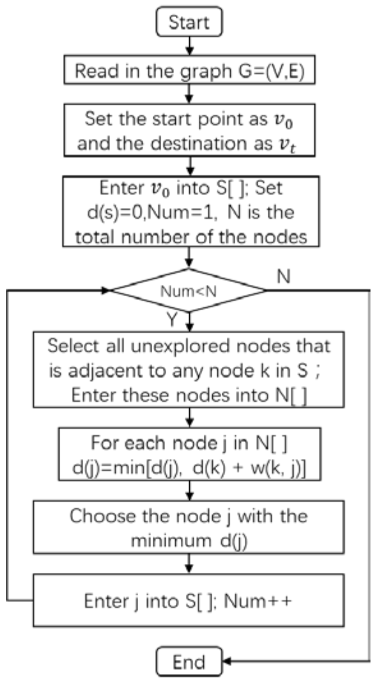

Start
Read in the graph G=(V,E)
Set the start point as v_0 and the destination as v_t
Enter v_0 into S[ ]; Set d(s)=0,Num=1, N is the total number of the nodes
Num<N
N
Y
Select all unexplored nodes that is adjacent to any node k in S ; Enter these nodes into N[ ]
For each node j in N[ ] d(j)=min[d(j), d(k) + w(k, j)]
Choose the node j with the minimum d(j)
Enter j into S[ ]; Num++
End

Figure 3.3 The Principle of Dijkstra’s Algorithm

By introducing the data structure “stack” into the algorithm (Knuth, 1997), we can write the Pseudocode for the algorithm as:

```
Step 1. Enter the adjacent nodes of the source node into the stack $S$;
Step 2. Adjust the stack by sorting the costs of the nodes, put the node which has smaller cost on the top
Step 3. Select the top node in $S$, mark it as $u$ and pop it out of $S$;
Step 4. While $n < N$ do
            Look for the nodes that are adjacent to $u$
            If the adjacent node is in $S$
            Update its cost and adjust its position in $S$
        Else
            Push the adjacent node into $S$
            Adjust $S$
```

Dijkstra's algorithm can provide us the shortest path from the origin to any other node in the graph, which is unnecessary for our problem. We only need to calculate the shortest path between the origin and the destination. Therefore, we need to modify the algorithm and add rules that how it searches for the destination. Next, we are going to discuss some graph traversal methods and combine them with the Dijkstra's algorithm.

Graph Traversal Methods

Graph traversal is also known as graph search. It is a process that we start from a certain vertex in a graph and visit each other vertices until we reach our destination. We can only visit each vertex in the graph once and only once.

Classified by the order in which the vertices are visited, graph traversal can be categorized into two different algorithms, which are depth-first search and breadth-first search. Graph traversal is the fundamental method solving graph connectivity problems, topological sorting, and finding critical paths.

(1) Depth-first search method

Similar to pre-order traversal in a tree, the main idea of depth-first search is to select a certain vertex v_i in a graph, and we begin searching from v_i by visiting an adjacent vertex v_j that hasn't been travelled to. After we have reached the end of that path, we need to go back and start from v_j to search for the next vertex by pre-order traversal till all vertices have been visited.

The example shown in Figure 3.4 gives us the order of traveling through the nodes using depth-first search on the graph, which is:

$$V_0 \rightarrow V_1 \rightarrow V_3 \rightarrow V_7 \rightarrow V_4 \rightarrow V_2 \rightarrow V_5 \rightarrow V_6$$

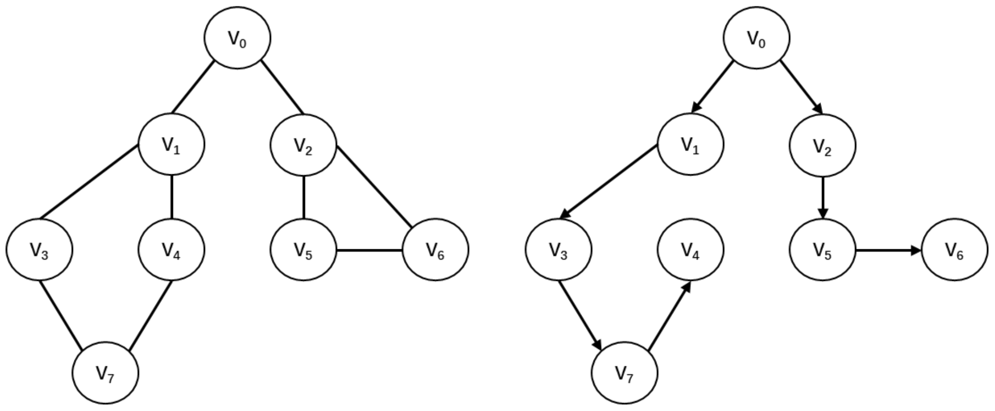


Figure 3.4 Depth-first search in a graph

The Pseudocode for depth-first search is:

```
Step 0. Initialize the stack S;
Step 1. Input the start point $V_0$; Set it as explored; Enter $V_0$ into S;
Step 2. While S is not empty do
            Read the top element in S, set as $v_T$
            If $v_T$ has unexplored adjacent vertex $w$ then
                Set $w$ as explored
                Push $w$ in the front of S
            Else:
                Pop the top element out of S
```

(2) Breadth-first search method

Different from depth-first search, breadth-first search visits the neighbor vertices first before visiting its child vertices. The example shown in Figure 3.5 gives us the order of traveling through the nodes using breadth-first search on the graph, which is:

$$V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_6 \rightarrow V_7$$

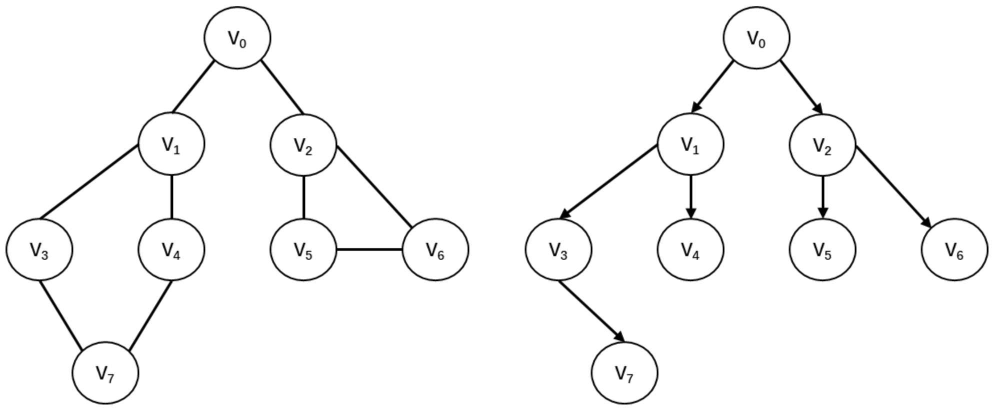


Figure 3.5 Breadth-first search in a graph

Likewise, we can write the Pseudocode for breadth-first search, which is:

```
Step 0. Initialize queue Q; Initialize visited$[V_0, V_1, \dots, V_n] = 0$;
Step 1. Visit start point $V_0$; Set visited$[V_0] = 1$; Enter $V_0$ into Q;
Step 2. While Q is not empty do
            $V_f$ = the first element in Q
            Dequeue $V_f$ from Q
            $W$ = the adjacent vertices of $V_f$
            For all adjacent vertices $w$ in $W$ do
                If w has not been visited, then
                    visited$[w] = 1$; Enter $w$ at the end of Q
                Look for next $w$ in $W$
```

Comparison Between Breadth-first Search and Depth-first Search

The choice between these two methods heavily depends on the structure of our graph and the number and location of our solutions. If the endpoint is far from the start point and each node only has considerable branches, BFS might perform better than DFS. However, it would be a better choice to use DFS when we know the destination is not far away from the start point and there are many branches.

As we can see from Figure 2.1, our road network is a sparse graph, which means there are not many branches for each vertex. Therefore, using BFS can calculate our optimal solution faster. We are going to use BFS for the implementation of predictive route planning.

Implementation of the Graph Traversal Method

Suppose our road network shown in Figure 2.1 is a directed graph represented by $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ with nonnegative arc (road piece) cost c_{ij}. The edges of the graph are the road pieces(TMCs). The vertices of the graph are the nodes between each road piece. The cost of each edge is changing with time and determined by weather and time when the car is traveling on that road piece. We will also use the LASSO regression model mentioned in Chapter 2 to calculate the travel time for each road piece.

Additionally, we need to construct a value dictionary to store the cost value and the cumulated travel time for each TMC. We are going to use S_i to represent the states, which are the TMCs. T_{S_i} is the cost value, which is the travel time at S_i. $C_{S_i}(T_{S_i})$ means the cumulated travel time from the origin to the current state.

$$\mathrm{VD} = \begin{bmatrix} [\mathrm{S}_1 & T_{S_1} & C_{S_1}(T_{S_1})] \\ [\mathrm{S}_2 & T_{S_2} & C_{S_2}(T_{S_2})] \\ & \cdots & \end{bmatrix}$$

$\mathrm{VD}[i][0]$ contains all the TMCs. The cost value, which is the travel time for each TMC, is calculated by the LASSO regression model and stored in $\mathrm{VD}[i][1]$.

We are going to update the value dictionary and store the shortest travel time from our source node to each vertex in $\mathrm{VD}[i][2]$. The update function for the cumulated travel time is given by:

$$\mathrm{VD}[i][2] = \min\{\mathrm{VD}[i][2], \mathrm{VD}[j][1] + \mathrm{VD}[j][2]\}$$

Initially, each cumulated travel time, which is the $\mathrm{VD}[i][2]$, is given the value of 999, which is a number that is large enough. Suppose $\mathrm{VD}[j]$ stores the information of the previous adjacent TMC. We are going to compare the cumulated travel time of current TMC with the cumulated travel time of the previous TMC plus the cost value of the previous TMC. We are going to update and record the minimum value of these two and give it to the current TMC.

Pseudocode of the predictive route planning algorithm based on graph traversal method is:

```
Step 0. Initialization
    Step 0a. Read in the transition matrix
    Step 0b. Initialize the value dictionary for road links (arcs of the graph):
```

$$\mathrm{VD} = \begin{bmatrix} [S_1 & T_{S_1} & C_{S1}(T_{S_1})] \\ [S_2 & T_{S_2} & C_{S1}(T_{S_2})] \\ & \cdots & \end{bmatrix}$$

S_1: It represents a TMC (road piece)

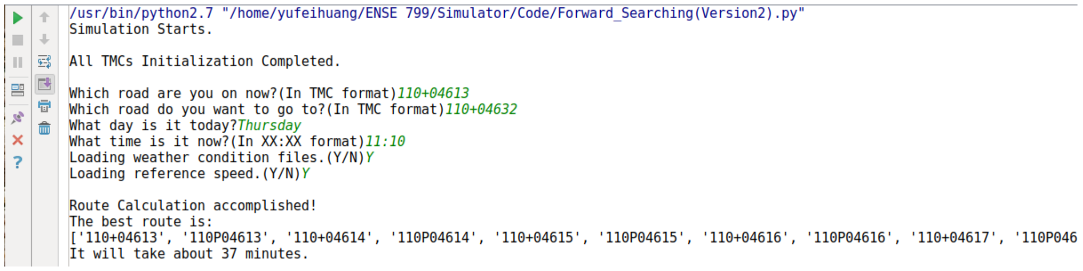
T_{S_1}: Accumulated Travel Time when we are at S_1

```
        $C_{S1}(T_{S_1})$: Immediate Cost of S1, which is the travel time of S1, it
            depends on $T_{S_1}$. Using the regression model, when we have
            $T_{S_1}$, we can find the weather condition and traffic condition at
            that time and calculate the travel time of $S_1$.
    Step 0c. Initialize the list for optimal path OP = []
Step 1. Input starting point, set as S. Input ending point, set as E; Enter current state S
    into the decision set: D = [S]
Step 2. Time calculating loop:
    For all states in decision Set:
        Look for next states in the transition matrix
        Make current states in the decision set leave
        Enter next states into the decision set (except the ending point)
        Calculate the cost values for all next states based on the regression
        model
            $\mathrm{T(TMC, t, Tr, W) = LM(D, t, Tr(t), W(t))}$
        Update the value dictionary for new states in the decision set
          $\mathrm{VD}[i][2] = \min\{\mathrm{VD}[i][2], \mathrm{VD}[j][1] + \mathrm{VD}[j][2]\}$
        Look for in the cumulated travel time for all new states in the time
        recording list
        If the cumulated travel time ($\mathrm{C}_T$) for a state is smaller than the value in
        the list
            update it in its value dictionary
        Else
```

```
                delete the state from decision set
            *When there is no state in the decision set, the time calculation is
            completed
Step 3. Route finding loop:
        Start from the ending point, find the value dictionary of the destination
        Calculate corresponding previous cumulated travel time, $T$ represents the total
        travel time
                        $C_T = T - T_{S_i}$
        Look for the new $C_T$ in the value dictionary
        Add the founded road piece into OP list reversely [..., TMC, End Point]
Step 4. Output the result: OP list and optimal travel time
```

Result

We will use the same origin, destination, day of the week, and time of the day to test our predictive route planning method. Figure 3.6 gives us a brief illustration of the inputs and outputs of our program. It gives us the same optimal route as the result of the random path method. However, there is little difference in the estimated travel time. We would consider the error is caused by decimal precision since we are given the same path.



```
/usr/bin/python2.7 "/home/yufeihuang/ENSE 799/Simulator/Code/Forward_Searching(Version2).py"
Simulation Starts.

All TMCs Initialization Completed.

Which road are you on now?(In TMC format)110+04613
Which road do you want to go to?(In TMC format)110+04632
What day is it today?Thursday
What time is it now?(In XX:XX format)11:10
Loading weather condition files.(Y/N)Y
Loading reference speed.(Y/N)Y

Route Calculation accomplished!
The best route is:
['110+04613', '110P04613', '110+04614', '110P04614', '110+04615', '110P04615', '110+04616', '110P04616', '110+04617', '110P046
It will take about 37 minutes.
```

Figure 3.6 Illustration of the inputs and outputs of the graph traversal method

Approximate Dynamic Programming

Dynamic Programming

Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems (Bellman, Dynamic Programming, 1957). The main idea of Dynamic Programming is to split the problem and define the relationship between each sub-problem. Using dynamic programming, the problem can be resolved in a recursive or a divide-and-conquer way.

The core of dynamic programming is to divide the problem into sub-states, which relies on how we define the states and the state transfer functions that vary from different questions. Let’s first make clear some basic concepts in dynamic programming.

(1) Multi-stage decision problems

Multiple-stage decisions refer to decision tasks that consist of a series of interdependent stages leading towards a final resolution. The decision-maker must decide at each stage what action to take next in order to optimize performance (Johnson & Busemeyer, 2001).

(2) Stage

The division of a sequence of a problem into various subparts is called stages. Stages may be different when we take different processes to solve the problem. Dynamic programming decomposes the problem into a set of n stages of analysis, each stage corresponding to one of the decisions. Each stage of analysis is described by a set of elements decision, input state, output state and returns (Kumar, 2013).

(3) State

The state represents a specific, measurable condition of the problem. It can be defined as the set of parameters that can uniquely identify a certain position or standing in the given problem (Elmaghraby, 1970).

(4) Decision

The decision is a choice or action that evolves the current state to the next state based on current stage.

(5) Policy

A policy is a set of principles to guide decisions and achieve rational outcomes (Bagnell, Kakade, NG, & Schneider, 2004).

(6) State transition equation

At stage k, when the state variable $x(k)$ is given, we can choose to take decision $u(k)$ to get the state variable $x(k+1)$ at stage $k+1$. We can use a function to represent the relation between $x(k)$, $u(k)$, and $x(k+1)$:

$$x(k+1) = T_k(x(k), u(k))$$

The equation is called state transfer equation. It shows the correspondence between $x(k)$, $u(k)$, and $x(k+1)$. Once $x(k)$ and $u(k)$ are given, $x(k+1)$ will also be determined.

(7) Backward recursion approach

Backward recursion approach takes a problem decomposed into a sequence of n stages and analyzes the problem starting with the last stage in the sequence, working backward to the first stage.

Figure 3.7 presents an example how dynamic programming works (Brown, 1979). In this problem, we are trying to find the minimum cost path from node A to node G. The costs are marked on each edge.

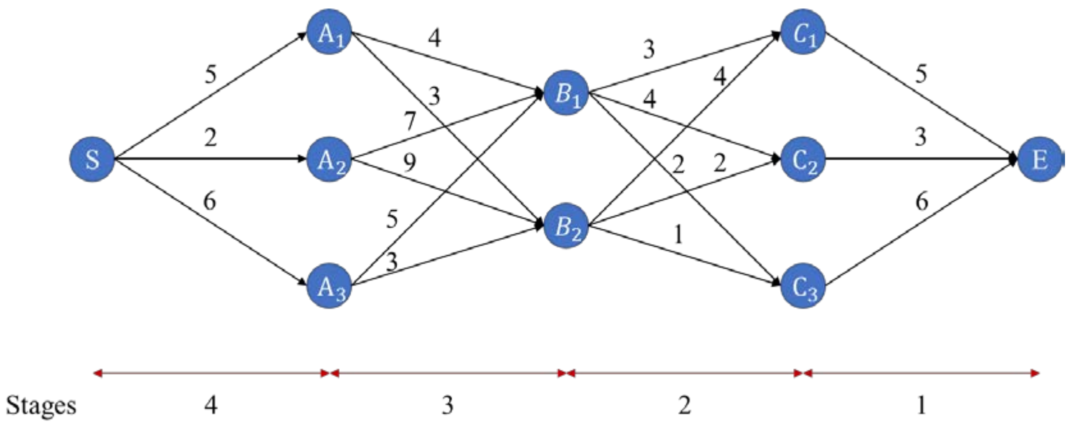


Figure 3.7 Example question of dynamic programming

The question is divided into 4 different stages, and each node represents a different state. Dynamic programming solves the problem backward. Before we start solving this question, we are going first to introduce the Bellman's equation:

$$\begin{cases} D_0(G) = 0 \\ D_k(j) = \min\{\mathrm{D_{k-1}}(i) + \mathrm{c_{ij}} \mid i \in \text{stage k}-1\} \end{cases}$$

In the equation, $D_k(j)$ represents the distance from state j at stage k to our goal point G. The distance of $D_k(j)$ is calculated by the minimum combination of the distance from state i at stage $k-1$ to G and the immediate cost moving from state i to state j. It is obvious that the distance of our goal point to itself is 0.

We are going to calculate the shortest distance from state A to state G using dynamic programming.

(1) Phase I

$k = 1$, we have 3 options for stage s_1, which are C_1, C_2, and C_3.

Objective function:

$f_1(C_1) = \min\{d_1(C_1, E)\} = 5$
$f_1(C_2) = \min\{d_1(C_2, E)\} = 3$
$f_1(C_3) = \min\{d_1(C_3, E)\} = 6$

Table 3.1 Phrase I of Dynamic Programming

	To E	$u_1{}^*$	$f_1{}^*$
C_1	5	C_1 to E	5
C_2	3	C_2 to E	3
C_3	6	C_3 to E	6

(2) Phase II

$k = 2$, we have 2 options for stage s_2, which are B_1 and B_2.

Objective function:

$$f_2(B_1) = min\begin{cases} d_2(B_1, C_1) + f_1(C_1) \\ d_2(B_1, C_2) + f_1(C_2) = 7 \\ d_2(B_1, C_3) + f_1(C_3) \end{cases}$$
$$f_2(B_2) = min\begin{cases} d_2(B_2, C_1) + f_1(C_1) \\ d_2(B_2, C_2) + f_1(C_2) = 5 \\ d_2(B_2, C_3) + f_1(C_3) \end{cases}$$

Table 3.2 Phrase II of Dynamic Programming

	To C_1	To C_2	To C_3	$u_2{}^*$	$f_2{}^*$
B_1	3+5	4+3	2+6	B_1 to C_2	7
B_2	4+5	2+3	1+6	B_2 to C_2	5

(3) Phase III

k = 3, we have 3 options for stage 3, which are A_1, A_2, and A_3.

Objective function:

$$f_3(A_1) = min\begin{cases} d_3(A_1, B_1) + f_2(B_1) \\ d_3(A_1, B_2) + f_2(B_2) \end{cases} = 8$$
$$f_3(A_2) = min\begin{cases} d_3(A_2, B_1) + f_2(B_1) \\ d_3(A_2, B_2) + f_2(B_2) \end{cases} = 14$$
$$f_3(A_3) = min\begin{cases} d_3(A_3, B_1) + f_2(B_1) \\ d_3(A_3, B_2) + f_2(B_2) \end{cases} = 8$$

Table 3.3 Phrase III of Dynamic Programming

	To B_1	To B_2	${u_3}^*$	${f_3}^*$
A_1	4+7	3+5	A_1 to B_2	8
A_2	7+7	9+5	A_2 to B_1 or A_2 to B_2	14
A_2	5+7	3+5	A_3 to B_2	8

(4) Phase IV

k = 4, we only have 1 options for stage 4, which is S.

Objective function:

$$f_4(S) = min\begin{cases} d_4(S, A_1) + f_3(A_1) \\ d_4(S, A_2) + f_3(A_2) \\ d_4(S, A_3) + f_3(A_3) \end{cases} = 13$$

Table 3.4 Phrase IV of Dynamic Programming

	To A_1	To A_2	To A_3	${u_4}^*$	${f_4}^*$
S	5+8	2+14	6+8	S to A_1	13

Finally, we get the optimal path of the question:

$$\text{S} \rightarrow A_1 \rightarrow B_2 \rightarrow C_2 \rightarrow E$$

We can obtain the minimum distance from S to E using dynamic programming, which is 13. What’s more, we have got all optimal paths from any starting node to our destination E.

Approximate Dynamic Programming

Although using dynamic programming can solve optimization problems, sometimes it is hard for us to get the answer to a large and complex question. For example, suggest our system has 10 stages, and each stage has 2 states. In this case, we need to make decisions for $2 \times 2 \times (10 - 1) = 36$ times. When the number of stages increases to 20 and each stage has 3 states, the total number of decisions we need to compare will increase to $3 \times 3 \times (20 - 1) = 171$. It would take a lot of memory to store all the choices. Such phenomenon is called the curse of dimensionality because when the dimensionality increases, the volume of the space increases so fast that it would be hard for us to handle (Bellman, 1961).

ADP is introduced to solve three types of curses of dimensionalities that dynamic programming might faces, which are high dimensional state variables, information variables, and potential actions (Powell, 2011).

Different from dynamic programming, ADP uses approximate value functions for each state instead of the exact cost. The optimal distance from each state to our destination does not have to be precise and correct at first. At each state, ADP chooses a sample path, which is a route from the origin to the destination, in an iteration. Through iterations, ADP learns from each sample path it travels and updates its value function using Bellman’s equation. Thus, it can learn from better policies and make better decisions over time (Powell, 2008).

For our problem, we are using a deterministic road model. Once we determine the start TMC, the end TMC, the time of the day, and the day of the week, we can use our regression model to calculate the travel time for each road piece and the total travel time. Therefore, we are going to modify this generic procedure to fit and solve our own problem.

Implementation of the Deterministic Approximate Dynamic Programming

In our thesis, once the time period and TMC is given, we can calculate the travel time that is a constant value by using the LASSO regression model. Therefore, we will use deterministic approximate dynamic programming to implement the predictive route planning.

Same as the assumptions that we made for the Graph Traversal method in the previous section, we are also going to take the road network shown in Figure 2.1. Likely, we would still use the regression model presented in Chapter 2 to calculate the travel time for each road piece.

We are going to build a lookup table to store the cost value, the cumulated travel time, and the value function for each TMC. We are going to use S_i to represent the states, which are the TMCs. T_{S_i} is the cost value, which is the travel time at S_i. $C_{S_i}(T_{S_i})$ means the cumulated travel time from the origin to the current state. $\bar{V}_t^{\,n}(S_i)$ is the value function for S_i in the nth sample path, which is the distance from S_i to the destination.

$$\mathrm{LT} = \begin{bmatrix} [\mathrm{S}_1 & T_{S_1} & C_{S_1}(T_{S_1}) & \bar{V}_t^{\,1}(S_1)] \\ [\mathrm{S}_2 & T_{S_2} & C_{S_2}(T_{S_2}) & \bar{V}_t^{\,1}(S_2)] \\ & & \cdots & \end{bmatrix}$$

The value functions would all be given the initial value of 999, which is a number that is large enough. We are going to update the minimum value of the value function to the corresponding TMC in the lookup table in each iteration.

One thing that would affect the learning effect is the balance between exploration and exploitation. Exploration is used for gathering more information from the unknown future states and exploitation is to make the best decision from the given current information. The best long-term strategy may involve short-term sacrifices because it used to gather enough information to make the best overall decisions. In order to deal with the exploration and exploitation problem during our trials for different sample paths, we are going to ask the program first to choose the TMCs that haven't been explored when it is making decisions. Additionally, we would give some chance to take the TMC with higher travel time to our destination because we would like to explore further to their subsequent TMCs to see if they would give us better results. While we are trying to choose which next TMC to choose, the program will have 80% chance to choose the TMCs with calculated better travel time. The TMCs that doesn't have a better result will also have 20% chance to be selected.

The Pseudocode of Approximate Dynamic Programming is:

```
Step 0. Initialization
    Step 0a.    Initialize the Lookup Table:
                $[[\mathrm{S}_1, T_{S_1}, C_{S_1}(T_{S_1}), \bar{V}_t^{\,1}(S_1)],$
                $[\mathrm{S}_2, T_{S_2}, C_{S_2}(T_{S_2}), \bar{V}_t^{\,1}(S_2)],$
                $\ldots]$
```

S_1: State variable, it represents a TMC (road piece)

T_{S_1}: Accumulated Travel Time when we are at S_1

$C_{S1}(T_{S_1})$: Immediate Cost of S1, which is the travel time of S1, it depends on T_{S_1}. Using the regression model, when we have T_{S_1}, we can find the weather condition and traffic condition at that time and calculate the travel time of S_1.

$\bar{V}_t^{\,1}(S_1)$: Initial estimated value function. It is the estimated travel time from S_1 to our destination S_n

We are going to initialize the lookup table for all road pieces and set T_{S_i}, $C_{S_i}(T)$, $\bar{V}_t^{\,1}(S_i)$ all 999s.

Step 0b. Set the starting point as our initial state S_{STR}, which is the starting TMC

Set $\bar{V}_t^{\,0}(S_{ENR}) = 0$

Set the destination as our goal state S_{ENR}, which is the ending TMC

Use S_t to represent our current stage, initialize S_t to be S_{STR}

Step 0c. Read in the adjacency matrix M_a of the road pieces

Step 0d. Set n = 1, which is the number of the sample path

Step 0e. Read in the starting time, give it to $\mathrm{T}_{S_{STR}}$

```
    Step 0f.    Set N=100 (for example), which is the total sample paths we
                would like to evaluate

Step 1. Choose the next state in a sample path $\omega^n$

                $S_{t+1}^n = D({S_t}^n, {x_t}^n)$

    # Exploration and Exploitation

    Generate a random variable $R \sim U(0,1)$, R follows uniform distribution
    between 0 and 1

    If $R < 0.2$:    # Use exploration strategy

        Look for the next stages we can take in $M_a$, select one of them
        completely randomly

        If the chosen state already exists in our sample path, use the same
        strategy to choose another state.

        # Avoid circuit in our sample path

        # When all possible next stages of current stage have been explored,
        we still want to give the stages with higher value functions
        opportunities to look at for their next stages

    Else:    # Use Exploitation strategy

        Look for the minimum value function for current state in $M_a$

        If there are some cost function values as 999
        # these states haven't been explored, explore them first
            Choose one of these states
        Else
            Choose the next state with the minimum cost function value
            $\hat{v}_t^n = \min_{x_t}(C_t(\mathrm{S}_t{}^n, x_t) + \bar{V}_{t+1}^{n-1}(\mathrm{S}_{t+1}))$
        If the chosen state already exists in our sample path, use the same
        strategy to choose another state.
        # Avoid circuit in our sample path
Step 2. While $S_{t+1} \neq S_{ENR}$     # Update the lookup table
    Step 2a. Use the regression model $CalTime()$ function to compute immediate
        cost of $S_t$
            $\mathrm{T(TMC, t, Tr, W) = LM(D, t, Tr(t), W(t))}$
    Step 2b. Update the accumulated travel time of $S_{t+1}$
            $\mathrm{T}_{S_{t+1}}{}^n = C_{\mathrm{S}_t}{}^n + \mathrm{T}_{S_t}{}^n$
    Step 3c. Update the value functions
        For all road pieces we have traveled in the sample path $\omega^n$
        # If the value function is still the initial value and hasn't been updated
        If $\hat{v}_t^{n-1} = 0$:
```

$$\hat{v}_t^n = \min_{x_t}(C_t(\mathrm{S}_t{}^n, x_t) + \bar{V}_{t+1}^{n-1}(\mathrm{S}_{t+1}))$$

# If we have found a better value for the value function

If $\hat{v}_t^n < \hat{v}_t^{n-1}$:

$$\hat{v}_t^n = \min_{x_t}(C_t(\mathrm{S}_t{}^n, x_t) + \bar{V}_{t+1}^{n-1}(\mathrm{S}_{t+1}))$$

Step 3d. Update current state

$\mathrm{S}_t^n = \mathrm{S}_{t+1}^n$

Step 3. If current state $\neq S_{ENR}$, go to step 1

Step 4. If $\mathrm{n} < \mathrm{N}$, set the current state $\mathrm{S}_t^n = S_{STR}$. Set $\mathrm{n} = \mathrm{n} + 1$, go back to step 1

Step 5. The minimum total travel time would be $\bar{V}_t{}^n(S_{STR})$

Look for the best path through adding stages with

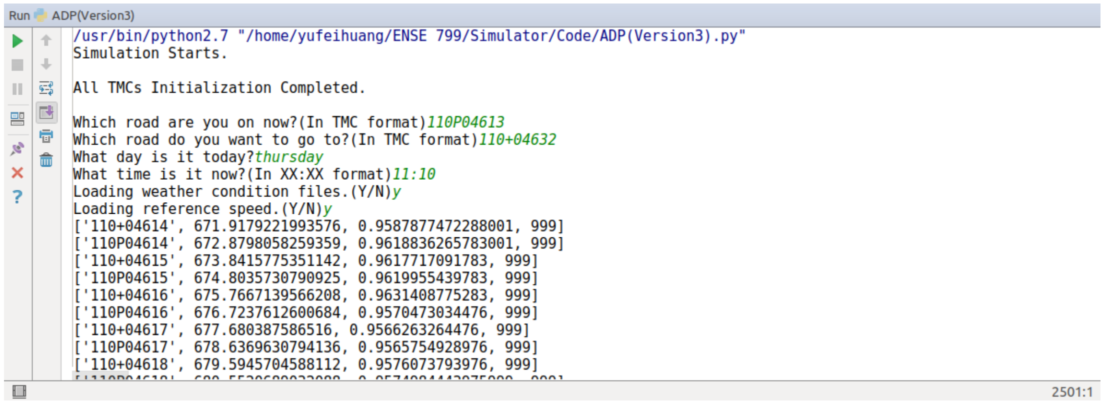
$$\overline{V_{t+1}}^n(S_{t+1}) = \bar{V}_t{}^n(\mathrm{S}_t) - C_{\mathrm{S}_t}{}^n$$

Until we reach S_{ENR}

Result

We will use the same example as we did for the graph traversal method to show the result of our algorithm. We are going to choose the TMC with its ID “110P04613” as the start point. We will use the TMC which is “110+04632” as the destination. The locations of these two TMCs are presented in Figure 3.1.

We are going to give our program the same inputs, as is shown in Figure 3.8. Figure 3.8 also gives us a brief view of the initial lookup table. All the value functions are given the value as 999.

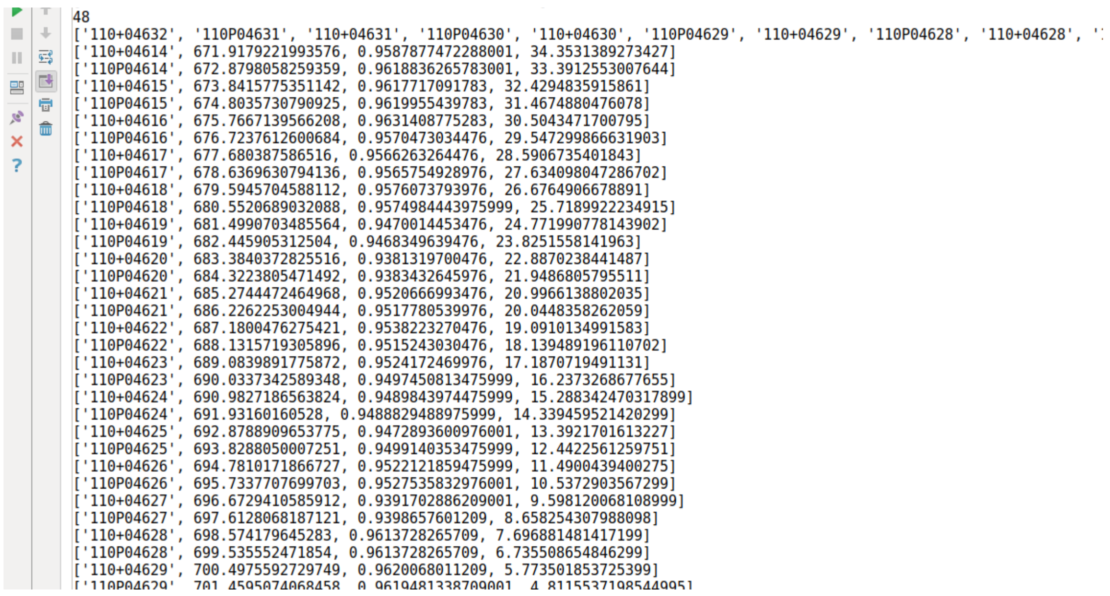

Run ADP(Version3)

```
/usr/bin/python2.7 "/home/yufeihuang/ENSE 799/Simulator/Code/ADP(Version3).py"
Simulation Starts.

All TMCs Initialization Completed.

Which road are you on now?(In TMC format)110P04613
Which road do you want to go to?(In TMC format)110+04632
What day is it today?thursday
What time is it now?(In XX:XX format)11:10
Loading weather condition files.(Y/N)y
Loading reference speed.(Y/N)y
['110+04614', 671.9179221993576, 0.9587877472288001, 999]
['110P04614', 672.8798058259359, 0.9618836265783001, 999]
['110+04615', 673.8415775351142, 0.9617717091783, 999]
['110P04615', 674.8035730790925, 0.9619955439783, 999]
['110+04616', 675.7667139566208, 0.9631408775283, 999]
['110P04616', 676.7237612600684, 0.9570473034476, 999]
['110+04617', 677.680387586516, 0.9566263264476, 999]
['110P04617', 678.6369630794136, 0.9565754928976, 999]
['110+04618', 679.5945704588112, 0.9576073793976, 999]
```

2501:1

Figure 3.8 Inputs of the ADP and the initial Value Functions

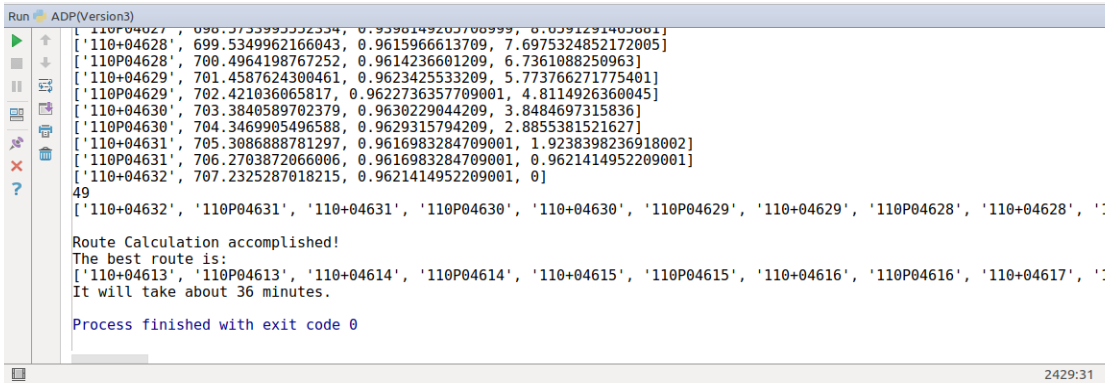
Figure 3.9 shows us the lookup table after 48th iteration. The lookup table after 48th iteration has already given us the optimal result. To be more specific, the program only needs about 30 iterations to find the optimal solution for our road network.



```
48
['110+04632', '110P04631', '110+04631', '110P04630', '110+04630', '110P04629', '110+04629', '110P04628', '110+04628', '
['110+04614', 671.9179221993576, 0.9587877472288001, 34.3531389273427]
['110P04614', 672.8798058259359, 0.9618836265783001, 33.3912553007644]
['110+04615', 673.8415775351142, 0.9617717091783, 32.4294835915861]
['110P04615', 674.8035730790925, 0.9619955439783, 31.4674880476078]
['110+04616', 675.7667139566208, 0.9631408775283, 30.5043471700795]
['110P04616', 676.7237612600684, 0.9570473034476, 29.547299866631903]
['110+04617', 677.680387586516, 0.9566263264476, 28.5906735401843]
['110P04617', 678.6369630794136, 0.9565754928976, 27.634098047286702]
['110+04618', 679.5945704588112, 0.9576073793976, 26.6764906678891]
['110P04618', 680.5520689032088, 0.9574984443975999, 25.7189922234915]
['110+04619', 681.4990703485564, 0.9470014453476, 24.771990778143902]
['110P04619', 682.445905312504, 0.9468349639476, 23.8251558141963]
['110+04620', 683.3840372825516, 0.9381319700476, 22.8870238441487]
['110P04620', 684.3223805471492, 0.9383432645976, 21.9486805795511]
['110+04621', 685.2744472464968, 0.9520666993476, 20.9966138802035]
['110P04621', 686.2262253004944, 0.9517780539976, 20.0448358262059]
['110+04622', 687.1800476275421, 0.9538223270476, 19.0910134991583]
['110P04622', 688.1315719305896, 0.9515243030476, 18.139489196110702]
['110+04623', 689.0839891775872, 0.9524172469976, 17.1870719491131]
['110P04623', 690.0337342589348, 0.9497450813475999, 16.2373268677655]
['110+04624', 690.9827186563824, 0.9489843974475999, 15.288342470317899]
['110P04624', 691.93160160528, 0.9488829488975999, 14.339459521420299]
['110+04625', 692.8788909653775, 0.9472893600976001, 13.3921701613227]
['110P04625', 693.8288050007251, 0.9499140353475999, 12.4422561259751]
['110+04626', 694.7810171866727, 0.9522121859475999, 11.4900439400275]
['110P04626', 695.7337707699703, 0.9527535832976001, 10.5372903567299]
['110+04627', 696.6729410585912, 0.9391702886209001, 9.598120068108999]
['110P04627', 697.6128068187121, 0.9398657601209, 8.658254307988098]
['110+04628', 698.574179645283, 0.9613728265709, 7.696881481417199]
['110P04628', 699.535552471854, 0.9613728265709, 6.735508654846299]
['110+04629', 700.4975592729749, 0.9620068011209, 5.773501853725399]
['110P04629', 701.4595074068458, 0.9619481338709001, 4.8115537198544995]
```

Figure 3.9 The Lookup Table After 48th Iteration

Figure 3.10 gives us the outputs of the ADP. It gives us the same optimal path from the TMC “110P04613” to the TMC “110+04632” as the other two algorithms.



```
['110+04628', 699.5349962166043, 0.9615966613709, 7.6975324852172005]
['110P04628', 700.4964198767252, 0.9614236601209, 6.7361088250963]
['110+04629', 701.4587624300461, 0.9623425533209, 5.773766271775401]
['110P04629', 702.421036065817, 0.9622736357709001, 4.8114926360045]
['110+04630', 703.3840589702379, 0.9630229044209, 3.8484697315836]
['110P04630', 704.3469905496588, 0.9629315794209, 2.8855381521627]
['110+04631', 705.3086888781297, 0.9616983284709001, 1.9238398236918002]
['110P04631', 706.2703872066006, 0.9616983284709001, 0.9621414952209001]
['110+04632', 707.2325287018215, 0.9621414952209001, 0]
49
['110+04632', '110P04631', '110+04631', '110P04630', '110+04630', '110P04629', '110+04629', '110P04628', '110+04628',

Route Calculation accomplished!
The best route is:
['110+04613', '110P04613', '110+04614', '110P04614', '110+04615', '110P04615', '110+04616', '110P04616', '110+04617',
It will take about 36 minutes.

Process finished with exit code 0
```

Figure 3.10 Outputs of the ADP

Optimal Routes in Different Time Periods

All the three algorithms can give us the same optimal paths with the same given inputs. Now, we are going to test our algorithms to see whether they can select the optimal paths during different time periods according to real-time traffic condition, which is the main goal that we would like to accomplish.

Suppose we are now near Tysons Corner Center, which is marked with the green sign in Figure 3.11. We want to visit Lady Bird Johnson Park, which is marked with the red sign in the figure. We have two options to choose from our simulated road network. One way is to go north via I-495 and then use George Washington Memorial Parkway to get to our destination. The other way is to travel south via I-495 and then use I-66 to get to the park. Both of these routes have a length of about 15 miles. We are going to use our predictive route planning algorithms to find the optimal path with minimum travel time during different time periods.

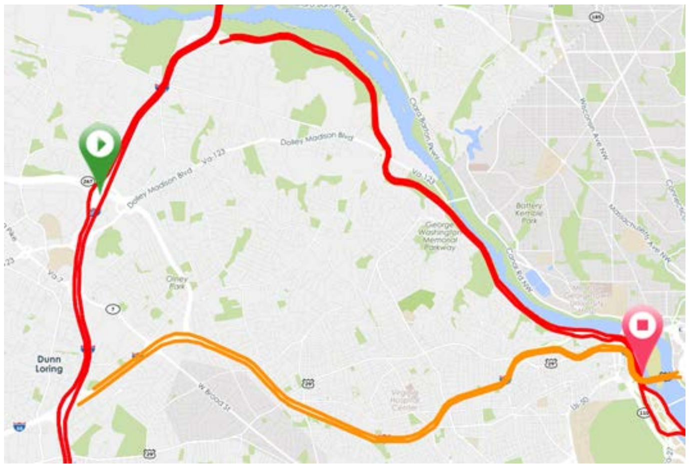


Figure 3.11 Illustration of the origin and destination of the test road

Table 3.5 gives us the test cases that we are going to perform. As is shown in the table, we are going to choose two different days of the week, which are Wednesday and Sunday. Also, we are going to test the routs at different times of the day. We will choose 4 different time periods on Wednesdays in which we will make our trip. We will also test the travel time between 6:00 PM and 7:00 PM on Sundays as the treatment group to test case 4.

Table 3.5 Test cases of optimal routes in different time periods

Test Case	Road Taken	Origin	Destination	Day of the week	Time of the day	Time of Travel (min)	Recommended Route
1	I-495-GW PKY	110P04611	110-04325	Wednesday	9:00 AM - 10:00 AM	33	I-495-I-66
	I-495-I-66	110N04611	110+04325	Wednesday	9:00 AM - 10:00 AM	31	
2	I-495-GW PKY	110P04611	110-04325	Wednesday	10:00 AM - 11:00 AM	24	I-495-GW PKY
	I-495-I-66	110N04611	110+04325	Wednesday	10:00 AM - 11:00 AM	27	
3	I-495-GW PKY	110P04611	110-04325	Wednesday	2:00 PM - 3:00 PM	22	I-495-I-66
	I-495-I-66	110N04611	110+04325	Wednesday	2:00 PM - 3:00 PM	21	
4	I-495-GW PKY	110P04611	110-04325	Wednesday	6:00 PM - 7:00 PM	26	I-495-GW PKY
	I-495-I-66	110N04611	110+04325	Wednesday	6:00 PM - 7:00 PM	28	
5	I-495-GW PKY	110P04611	110-04325	Sunday	6:00 PM - 7:00 PM	24	I-495-I-66
	I-495-I-66	110N04611	110+04325	Sunday	6:00 PM - 7:00 PM	20	

The recommended route calculated by all the three algorithms are the same. As we can see from the table, the travel time on both routes varies in different time periods. Both roads are crowded in the morning between 9:00 AM and 10:00 AM, and it takes us more time to travel through the roads compared to other test cases. Between 6:00 PM and 7:00 PM on Wednesday, using I-66 would cost us more time than using George Washington Memorial Parkway. It is likely that people usually get off work around 6:00 PM on Wednesdays and more people would use I-66 to go home instead of using George Washington Memorial Parkway, and it increases the travel time on I-66. However, as we can see in the test case 5, the travel time for both roads is shorter than the same time period on Wednesdays. I-66 would be a better choice on Sundays between 6:00 PM and 7:00 PM.

Therefore, we can conclude that different paths would cost us different travel time at different times. Our predictive route planning algorithms are able to help us select the optimal path during different time periods, which realizes the main task of our thesis.

Chapter 4: Comparison between the Route Planning Algorithms

In this chapter, we first present a way to reduce the storage space for the adjacent TMCs in our road network (Golden, 1988). Next, we are going to discuss the computational time of the predictive route planning methods we presented in the previous chapter. We will compare the pros and cons of these methods and find out which works better for the route planning problem.

Storage Space Analysis

We are going to use the four TMCs shown in Figure 4.1 to show how we can reduce the storage space for the road links in our program. The IDs for these four TMCs are “110+04269” (it belongs to MD-295 and goes from the down left corner to the middle in the figure), “110-04268” (it belongs to MD-295 and goes from the middle to the down left corner in the figure), “110P04631” (it belongs to I-495 and goes from the top left to the down right in the figure), and “110N04631” (it belongs to I-495 and goes from the down right to the top left in the figure).

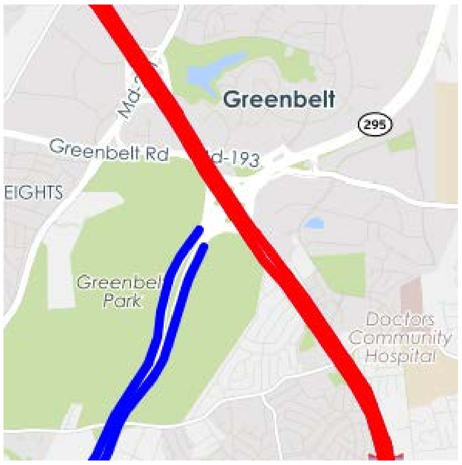


Figure 4.1 Illustration of the picked road pieces

Matrix Representation

Matrix representation is a widely used method to represent a finite graph. The elements in the matrix tell us whether each pair of vertices is adjacent or not in the graph. Figure 4.2 gives a brief view of how we can form the adjacency matrix for the given four TMCs. The direction of each edge for each pair of nodes is defined from rows to columns. If two nodes are connected, it will be given the value as “1” in the matrix. On the contrary, the value would be given as “0” if two nodes are disconnected.

Node Data			End			
Entry Position	Nodes	Start	1	2	3	4
1	110+04269	1	0	0	1	1
2	110-04268	2	0	0	0	0
3	110N04631	3	0	1	0	0
4	110P04631	4	0	1	0	0

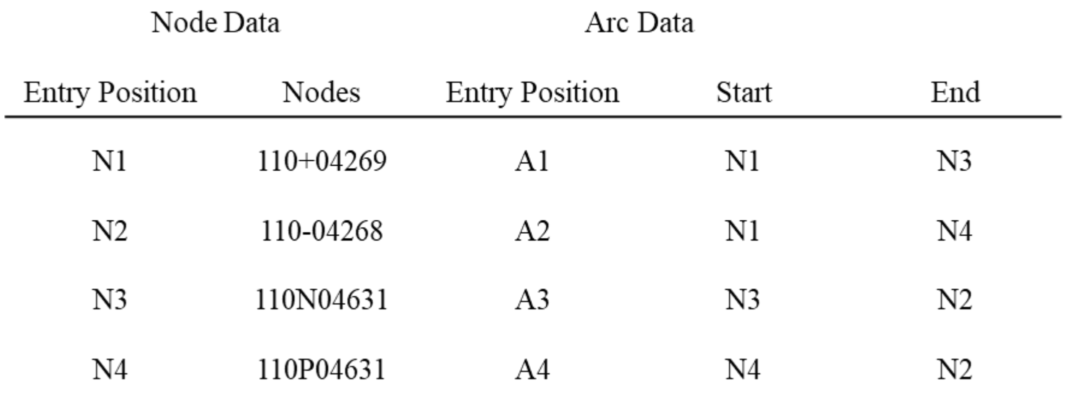
Figure 4.2 Adjacency matrix

If we use n to represent the total number of TMCs in our road network and use m to represent the total number of links between the road pieces, the space complexity for recording the entry position of each node is n and n^2 is the space complexity for storing the adjacency matrix. The total storage space of the computer memory that we need for the matrix representation method can be represented as:

$$S(n) = n + n^2$$

Adjacency Table

Adjacency table is another method to gather the information of all the nodes and arcs from the graph. Different from the matrix representation method, adjacency table directly records each pair of adjacent vertices. As is shown in Figure 4.3, the structure contains two parts. The first part is to give entry positions to all the nodes in our road network. The second part is to store each arc in the graph.



Node Data		Arc Data		
Entry Position	Nodes	Entry Position	Start	End
N1	110+04269	A1	N1	N3
N2	110-04268	A2	N1	N4
N3	110N04631	A3	N3	N2
N4	110P04631	A4	N4	N2

Figure 4.3 Adjacency table

Using this method, the space complexity for node data is n and the space complexity for arc data is $2m$. The total space complexity can be represented as:

$$S(n) = n + 2m$$

Conclusion

Notice that most values in the adjacency matrix are 0s, it would take us much more space using the adjacency matrix than using adjacency table. As a conclusion, we are going to use the adjacency table for storing the information of how each pair of TMCs is connected to reduce the storage space in the memory.

Running Time Analysis

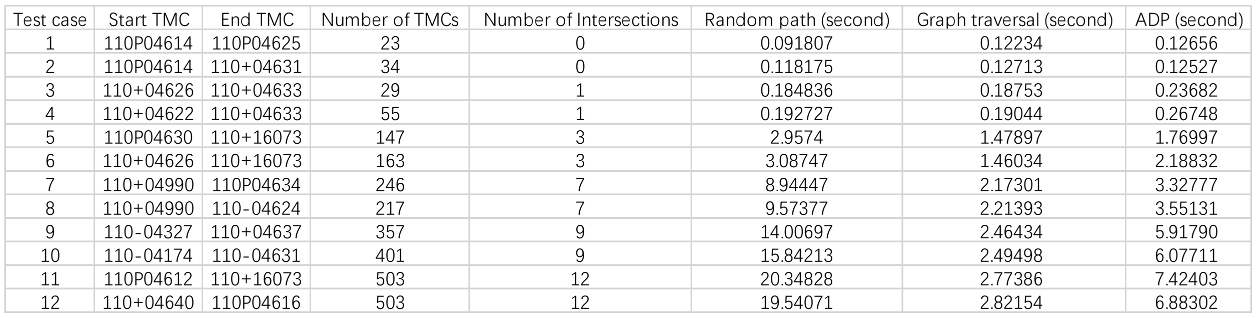
In this section, we will test the running time for the algorithms to seek the optimal path in different situations, including different origins and destinations, different origins and same destination, and the increasing complexity of the road network. We want to find out which algorithm is for our route planning problem.

Test of Algorithms with Different Origins and Destinations

We will first test the running time of the algorithms to find the optimal path with different origins and destinations. We want to see how these algorithms would perform when there are increasing numbers of TMCs and intersections between the start point and the ending point.

Table 4.1 shows us the test cases that we are going to perform. We take some parts out of our whole road network one at a time. We will test the performance of the algorithms by increasing the number of TMCs and the number of intersections in the sub-network. We will control the factors “Day of the week” and “Time of the day” to be the same in all these test cases. By changing the starting and ending TMCs, we can have different length and complexity of the road network.

Table 4.1 Test results for different origins and destinations



Test case	Start TMC	End TMC	Number of TMCs	Number of Intersections	Random path (second)	Graph traversal (second)	ADP (second)
1	110P04614	110P04625	23	0	0.091807	0.12234	0.12656
2	110P04614	110+04631	34	0	0.118175	0.12713	0.12527
3	110+04626	110+04633	29	1	0.184836	0.18753	0.23682
4	110+04622	110+04633	55	1	0.192727	0.19044	0.26748
5	110P04630	110+16073	147	3	2.9574	1.47897	1.76997
6	110+04626	110+16073	163	3	3.08747	1.46034	2.18832
7	110+04990	110P04634	246	7	8.94447	2.17301	3.32777
8	110+04990	110-04624	217	7	9.57377	2.21393	3.55131
9	110-04327	110+04637	357	9	14.00697	2.46434	5.91790
10	110-04174	110-04631	401	9	15.84213	2.49498	6.07711
11	110P04612	110+16073	503	12	20.34828	2.77386	7.42403
12	110+04640	110P04616	503	12	19.54071	2.82154	6.88302

Test case 1 and test case 2 gives us the simplest road network where there is only a list of TMCs and no intersections, which means the path is straightforward from our source node to the destination. Then, we gradually increase the number of TMCs and the number of intersections in the road network. Finally, in test case 11 and test case 12, we would use the full graph to test the running time.

How we define when the program has found the optimal path is clear for the graph traversal method because it calculates the travel time to all TMCs in the graph. It will compare all the possible routes to the destination and directly give us the best choice it can make. However, things would become complicated for the random path method and for approximate dynamic programming because both of these methods are trying different sample paths from the start point to our destination through iterations. The methods might find the optimal path by chance but maybe not aware that it has the best solution. Therefore, we set the rule that if these methods visit through the optimal solution at the second time and cannot make any improvement to it, we would consider they have found the optimal path.

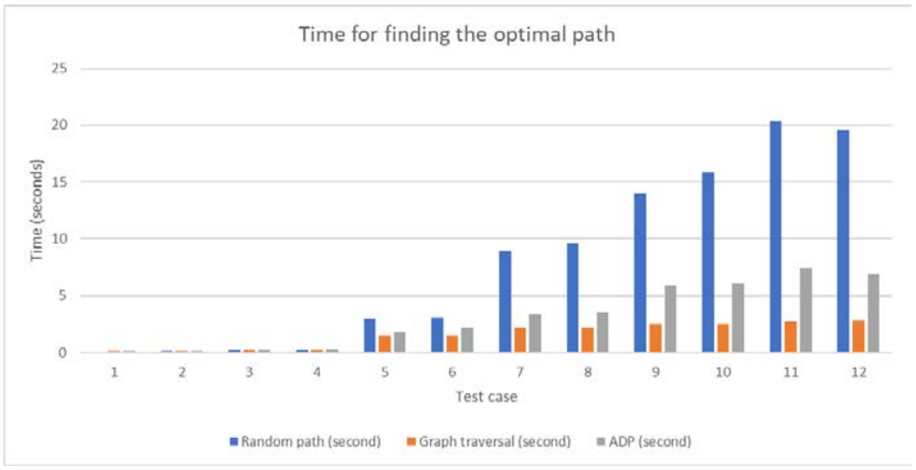


Figure 4.4 Time for finding optimal paths with different origins and destinations

As we can see from table 4.1, the computational time for all three kinds of route planning algorithms grows with the complexity of the graph. Figure 4.4 gives us a more intuitive view of the tendency of the growth of the computational time. Also, it helps us compare the performance of the three different algorithms. In the first two test cases, whose number of TMCs and intersections are small, all these three algorithms can easily find the optimal path and the program runs fast. However, when our graph grows bigger and more complex, the running time of all these methods become longer.

Among all these three methods, the running time of random path method grows the fastest. The reason is that when the graph becomes more complicated, it is harder for the method to find the direct path to our destination. It would waste time on longer paths that go away from the destination. Comparing between the graph traversal method and the ADP, the graph traversal method performs better. The reason is that our graph is a sparse graph and there are not too many choices to examine at the intersection. Therefore, it won't take much time for the method to explore all the paths in the graph. Although ADP also looks for the destination by travelling through different sample paths, it records and updates the value function for the cost of each TMC to the destination through iterations, which is different from the random path methods. Thus, it can reject some long paths that it has examined and stick to better routes with lower values of the cost function.

Test of Algorithms with Denser Road Networks

The previous test shows that the graph traversal method performs better than the other two algorithms in a sparse road network. However, the road network is usually a dense graph, which means there would be lots of intersections connecting different roads, especially in urban areas. We would like to examine the performance of algorithms in a more complex road network.

As is shown in Figure 4.5, we artificially create a road network by randomly picking TMCs in Washington D.C. area. Each TMC will be chosen only once. We first determine a center node, which is “110P04603”, to be our first layer of the road network. Then we add four more TMCs around the center node and connect them to be our second layer. Using the same method, we would get our third layer and so on.

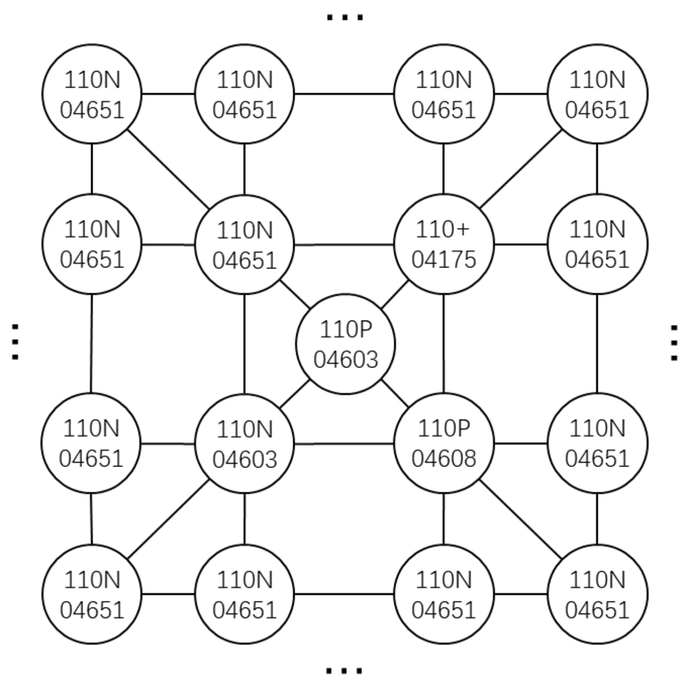


Figure 4.5 Illustration of a denser road networks

The number of TMCs, which is defined as n, on the i^{th} $(n > 2)$ layer can be represented as:

$$n = 4 \times [2 \times (i - 1) - 2] + 4 = 8i - 4$$

We are going to use j to represent the largest number of layers we can have. The total number of TMCs of all layers, which is represented as N, is:

$$N = 1 + 4 + \sum_{3}^{j}(8i - 4)$$

We have 503 different TMCs in total. Thus, N should be smaller or equal to 503. We can calculate that j can take the value as 11 at most. We are going to test the running time of the algorithms when the number of layers increases from 6 to 11. There are not many TMCs in the graph when the number of layers is less than 6 and all the algorithms can solve them easily. Therefore, we are not going to test these cases. In all tests, we set our origin at the top-left of the graph and we want to get to the bottom-right corner. The day of the week will all be set as Thursday and the time of day will all be set as 11:10 AM.

Table 4.2 shows us the results of the test. As we can see from the table, all three algorithms perform worse than the previous test. The result shows that the random path method is totally not applicable. The computational time for the graph traversal method increases greatly compared to the test cases with the similar number of TMCs in the previous section. However, the computational time for ADP doesn't increase as greatly as the other two methods.

Table 4.2 Computational time for different test cases

Test case	Number of Layers	Number of TMCs	Random path (second)	Graph traversal (second)	ADP (second)
1	6	133	4.77	1.68	1.85
2	7	185	10.83	2.03	2.46
3	8	245	13.41	3.70	3.66
4	9	313	22.16	5.38	5.43
5	10	389	37.04	7.55	6.46
6	11	473	63.21	9.27	7.58

Figure 4.6 gives us a clearer view of the results of graph traversal method and ADP. In the first three test cases, the graph traversal method can give us the optimal path faster than ADP. But after test case 4, ADP starts to perform better than the

graph traversal method. It is likely that ADP can perform better in a denser road network because it only considers one sample path at a time and then updates the value function. On the contrary, the graph traversal method takes all possible choices at any intersection into consideration at the same time, which increases the running time of the program.

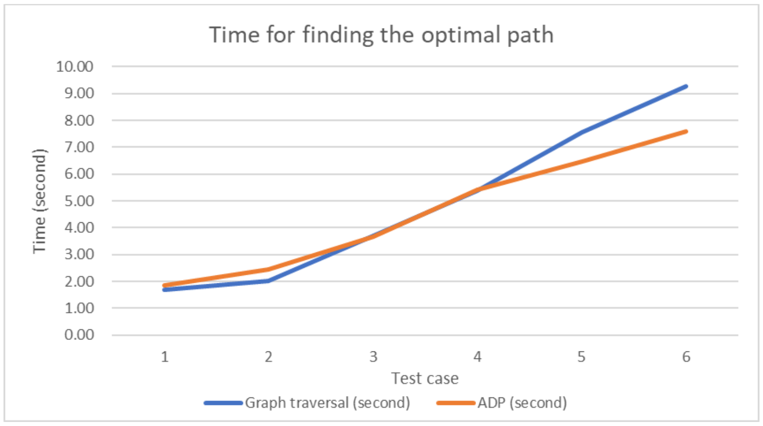


Figure 4.6 Time for finding the optimal path with the increasing complexity of the road network

Test of Algorithms with the Same Destination and Different Origins

One of the advantages of ADP is that it calculates the shortest travel time to the destination for each of the node on the sample path. It means that the method calculates not only the answer to one route planning problem, but also a series of questions that share the same destination. This test is designed to examine how all these three algorithms perform with different origins but the same destination.

Table 4.3 Test cases for Same Destination and Different Origins

Test case	1	2	3	4	5
Origin	110+16073	110P16073	110+04634	110P04634	110+04635
Destination	110P04612	110P04612	110P04612	110P04612	110P04612
Random path (second)	13.414	20.6948	15.3235	16.3987	15.3128
Graph traversal (second)	2.507435	2.617607	2.842682	2.460343	2.54518
ADP (second)	7.424034	3.348512	0.478071	0.059442	0.058716
Test case	6	7	8	9	10
Origin	110P04635	110+04636	110P04636	110+51498	110P51498
Destination	110P04612	110P04612	110P04612	110P04612	110P04612
Random path (second)	20.3025	14.8876	13.7458	20.2119	16.3063
Graph traversal (second)	2.625921	2.762889	2.826987	2.712559	2.979012
ADP (second)	0.410924	0.079567	0.228979	0.431485	0.213027

As is shown in Table 4.3, we set our destination to be the TMC “110+16073”. We are going to test a set of origins between the TMC “110P04612” and our destination. We will also keep the “Day of the week” to be Thursday and “Time of the day” to be 11:10 AM.

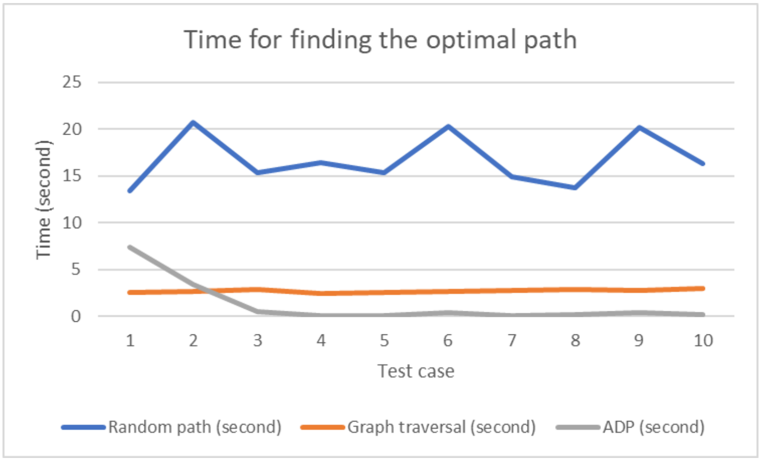


Figure 4.7 Time for finding the optimal path with the same destination and different origins

Figure 4.7 tells us the result of the test. As we can see from the graph, the performance of random path method is very unsteady, and the computational time varies between 8 seconds to 21 seconds. Since the method is randomly picking paths to explore, there is no wonder why it takes very long time to find the optimal path and has a broad range of time in a large graph. The running time for the graph traversal method is very stable and the running time is usually between 2.5 seconds and 3 seconds. It is much better than the random path method.

The running time of ADP is long for the first two trials, which are about 7 seconds and 4 seconds. Then, the time taken for finding the optimal path would become short later, which are less than 1 second. The reason for this phenomenon is that ADP uses the cost value function to store the shortest travel time from each TMC to the destination. And the program can still use the lookup table to get the values for future use. Therefore, the program can avoid some long paths that have been explored in the previous iterations.

Conclusion

Based on all the tests performed above, we can conclude that we would recommend ADP for solving route planning problems. The reason is that it performs better than other two algorithms when we have a more complex road network. Also, the approximate dynamic programming can give us the answers to a series of questions that shares the same destination by solving only one route. However, the random path method is not as good as the other two methods, and it should not be used for completing our goal in the thesis.

Chapter 5: Conclusion and Future Work

In our thesis, we studied the main factors that would affect the travel time of roads in Washington D.C. area and implemented different statistical models to fit them. LASSO regression was chosen as the best fit for the travel time. Also, we introduced some route planning algorithms that can help us find the fastest path using the predicted travel time for each road piece between the origin and the destination. ADP was recommended to be used for planning the route. In our last chapter, we are going to present our findings and some future work that might be illuminating.

Summary of Findings

(1) Main factors that affect traffic conditions.

From the LASSO regression model that we obtained in chapter 2, we can tell that extreme weather conditions have a great impact on everyday traffic. The most significant weather condition is heavy rain, while some mild weather condition will not affect the traffic condition much. For example, temperature and wind speed are less significant to the traffic on the road if they are within reasonable ranges. Busy hours in Washington D.C. area are between 9:00 AM and 10:00 PM, as well as 6:00 PM and 7:00 PM. The traffic on Thursdays is the busiest in a week.

(2) The coefficients of LASSO model are easier to explain.

Compared to other statistical models, the coefficients calculated by LASSO regression is closer to reality and easier to explain. For example, some factors that have negative effects on the travel time have positive coefficients in the regression model, while the factors that can reduce the travel time would have negative coefficients in the model.

(3) LASSO regression reduces the number of features and obtains the same mean squared error.

Compared to the linear regression model, the LASSO model for I-395 removes forty-nine less significant explanatory factors. It reduces the number of features in the model while keeping the same mean squared error calculated by 5-fold cross-validation.

(4) SIS can improve the prediction accuracy of GLM.

SIS is able to screen out some irrelevant features from GLM. It reduces the mean squared error of GLM and improves the model.

(5) ADP is a better method to solve complicated route planning problems.

For a complicated route planning problem, applying approximate dynamic programming can provide us with the desired result faster. One of the advantages of using approximate dynamic programming is that it can learn from the historic trails and the learning results can be inherited and used for solving other questions.

Future Research

(1) Use stochastic models to calculate the predicted travel time.

In our thesis, we used deterministic regression models for the predicted travel time. Once the independent variables in the function are defined, the travel time of the roads would be determined as a fixed value. In reality, the prediction cannot just be a determined number, and it should vary with a distribution. We can further study our predictive route planning method based on the stochastic models and see how they work.

(2) Study a more complex road network including urban streets.

In this thesis, we mainly studied some US routes and State routes, which are usually highways, in Washington D.C. area as is mentioned in Chapter 2. We did not take major urban streets into consideration. Factors that affect travel time on urban streets might be different from those on highways. For example, traffic light plays a significant role in controlling traffic flows. An appropriate traffic light cycle length can help relieve traffic congestion and reduce the travel time (National Association of City Transportation Officials, 2013).

Appendices

Appendix A: Result of Linear Regression

```
Call:
lm(formula = Traveling.Time.min. ~ ., data = test1)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.8300 -0.0332 -0.0025  0.0276 12.6236 

Coefficients: (3 not defined because of singularities)
                                Estimate Std. Error  t value Pr(>|t|)    
(Intercept)                    7.515e-01  2.461e-03  305.307  < 2e-16 ***
X110.04651                     2.390e-01  1.771e-03  134.986  < 2e-16 ***
X110P04650                     7.717e-02  1.771e-03   43.574  < 2e-16 ***
X110P04651                     7.569e-01  1.791e-03  422.522  < 2e-16 ***
X110.04652                     4.704e-01  1.791e-03  262.578  < 2e-16 ***
X110P04652                     5.639e-01  1.785e-03  315.944  < 2e-16 ***
X110.16046                     4.350e-01  1.785e-03  243.761  < 2e-16 ***
X110P16046                     7.160e-01  1.774e-03  403.642  < 2e-16 ***
X110.04653                     1.548e-01  1.774e-03   87.247  < 2e-16 ***
X110P04653                     7.779e-01  1.781e-03  436.815  < 2e-16 ***
X110.04654                     9.905e-01  1.781e-03  556.191  < 2e-16 ***
X110P04654                     6.193e-01  1.779e-03  348.068  < 2e-16 ***
X110.04655                     4.253e-01  1.779e-03  239.014  < 2e-16 ***
X110P04655                     7.297e-01  1.776e-03  410.994  < 2e-16 ***
X110.04656                     2.055e-01  1.775e-03  115.764  < 2e-16 ***
X110P04656                     6.468e-01  1.772e-03  365.052  < 2e-16 ***
X110.04657                     1.507e-01  1.772e-03   85.030  < 2e-16 ***
X110P04657                     6.043e-01  1.752e-03  344.908  < 2e-16 ***
X110.04298                     6.405e-01  1.760e-03  364.007  < 2e-16 ***
X110.04299                     9.732e-02  1.749e-03   55.632  < 2e-16 ***
X110P04299                     5.874e-01  1.819e-03  323.017  < 2e-16 ***
X110P04298                     1.079e+00  1.768e-03  610.065  < 2e-16 ***
X110.04300                     7.477e-02  1.753e-03   42.657  < 2e-16 ***
X110P04300                     6.280e-02  1.783e-03   35.227  < 2e-16 ***
X110.04301                     8.398e-02  1.783e-03   47.110  < 2e-16 ***
X110P04301                     3.145e-01  1.778e-03  176.846  < 2e-16 ***
X110.04302                    -1.646e-01  1.778e-03  -92.576  < 2e-16 ***
X110P04302                     1.334e-01  1.771e-03   75.356  < 2e-16 ***
X110.04303                    -6.506e-02  1.877e-03  -34.658  < 2e-16 ***
X110N04302                     1.105e-01  1.333e-03   82.875  < 2e-16 ***
X110.04301.1                  -8.656e-02  1.333e-03  -64.934  < 2e-16 ***
X110N04301                     2.008e-01  1.365e-03  147.049  < 2e-16 ***
X110.04300.1                   8.117e-02  1.365e-03   59.455  < 2e-16 ***
X110N04300                     4.589e-02  1.328e-03   34.564  < 2e-16 ***
X110.04299.1                   1.535e-02  1.328e-03   11.561  < 2e-16 ***
X110N04299                     3.551e-01  1.331e-03  266.819  < 2e-16 ***
X110N04298                     1.149e+00  1.331e-03  863.421  < 2e-16 ***
X110.04298.1                   9.991e-02  1.376e-03   72.611  < 2e-16 ***
X110.04657.1                   5.343e-01  1.376e-03  388.332  < 2e-16 ***
X110N04657                     6.159e-01  1.338e-03  460.380  < 2e-16 ***
X110.04656.1                   1.281e-01  1.338e-03   95.788  < 2e-16 ***
X110N04656                     6.087e-01  1.332e-03  456.916  < 2e-16 ***
X110.04655.1                   1.629e-01  1.332e-03  122.256  < 2e-16 ***
X110N04655                     7.140e-01  1.345e-03  530.751  < 2e-16 ***
X110.04654.1                   3.840e-01  1.345e-03  285.417  < 2e-16 ***
X110N04654                     6.056e-01  1.340e-03  451.798  < 2e-16 ***
X110.04653.1                   9.957e-01  1.340e-03  742.818  < 2e-16 ***
X110N04653                     7.625e-01  1.342e-03  568.180  < 2e-16 ***
```

```
X110.16046.1                   1.181e-01  1.342e-03   87.994  < 2e-16 ***
X110N16046                     7.267e-01  1.361e-03  533.777  < 2e-16 ***
X110.04652.1                   4.128e-01  1.361e-03  303.244  < 2e-16 ***
X110N04652                     5.155e-01  1.320e-03  390.383  < 2e-16 ***
X110.04651.1                   5.100e-01  1.320e-03  386.283  < 2e-16 ***
X110N04651                     7.246e-01  1.335e-03  542.644  < 2e-16 ***
X110.04650                     2.155e-01  1.335e-03  161.339  < 2e-16 ***
X110N04650                            NA         NA       NA       NA
Monday                         1.412e-02  6.599e-04   21.395  < 2e-16 ***
Tuesday                        1.406e-02  7.216e-04   19.483  < 2e-16 ***
Wednesday                      9.742e-03  7.422e-04   13.126  < 2e-16 ***
Thursday                       2.253e-02  7.193e-04   31.326  < 2e-16 ***
Friday                         1.453e-02  6.230e-04   23.328  < 2e-16 ***
Saturday                       7.980e-03  6.813e-04   11.712  < 2e-16 ***
Sunday                                NA         NA       NA       NA
X0.1am                        -7.626e-03  1.117e-03   -6.830 8.51e-12 ***
X1.2am                        -9.954e-03  1.131e-03   -8.804  < 2e-16 ***
X2.3am                        -1.363e-02  1.119e-03  -12.177  < 2e-16 ***
X3.4am                        -8.609e-03  1.064e-03   -8.089 6.02e-16 ***
X4.5am                         3.477e-03  1.064e-03    3.268 0.001084 **
X5.6am                         2.047e-02  1.084e-03   18.894  < 2e-16 ***
X6.7am                        -1.219e-02  1.088e-03  -11.200  < 2e-16 ***
X7.8am                        -2.942e-02  1.111e-03  -26.472  < 2e-16 ***
X8.9am                        -3.133e-02  1.128e-03  -27.780  < 2e-16 ***
X9.10am                        3.528e-02  1.126e-03   31.330  < 2e-16 ***
X10.11am                       5.973e-03  1.119e-03    5.336 9.50e-08 ***
X11.12am                       3.747e-03  1.300e-03    2.881 0.003960 **
X0.1pm                         4.475e-03  1.288e-03    3.474 0.000513 ***
X1.2pm                         1.805e-02  1.106e-03   16.322  < 2e-16 ***
X2.3pm                         1.315e-02  1.089e-03   12.074  < 2e-16 ***
X3.4pm                         3.249e-03  1.079e-03    3.011 0.002601 **
X4.5pm                         2.924e-03  1.081e-03    2.704 0.006846 **
X5.6pm                         9.555e-04  1.067e-03    0.895 0.370685
X6.7pm                         1.156e-02  1.068e-03   10.827  < 2e-16 ***
X7.8pm                        -9.268e-03  1.051e-03   -8.820  < 2e-16 ***
X8.9pm                        -1.089e-02  1.040e-03  -10.471  < 2e-16 ***
X9.10pm                       -1.019e-02  1.035e-03   -9.847  < 2e-16 ***
X10.11pm                      -5.615e-03  1.035e-03   -5.426 5.76e-08 ***
X11.12pm                              NA         NA       NA       NA
Previous.Average.Speed.mph.   -7.681e-03  2.581e-05 -297.580  < 2e-16 ***
Average.Speed.Afterwards.mph. -5.498e-03  2.568e-05 -214.043  < 2e-16 ***
Temp.F.                        2.031e-03  6.224e-05   32.633  < 2e-16 ***
Dew.Point.F.                  -1.918e-03  6.739e-05  -28.468  < 2e-16 ***
Humidity                       7.915e-02  3.580e-03   22.108  < 2e-16 ***
Pressure.in.                  -1.147e-03  1.015e-03   -1.131 0.258203
Visibility.mile.               2.899e-03  1.786e-04   16.232  < 2e-16 ***
Wind.Speed.mph.               -1.762e-04  3.568e-05   -4.939 7.85e-07 ***
Gust.Speed.mph.               -1.674e-04  2.178e-05   -7.684 1.54e-14 ***
Precip.in.                    -2.648e-02  1.381e-02   -1.917 0.055225 .
Light.Rain                    -7.817e-02  3.139e-02   -2.490 0.012760 *
Overcast                      -6.635e-02  3.140e-02   -2.113 0.034582 *
Mostly.Cloudy                 -7.282e-02  3.138e-02   -2.320 0.020315 *
Heavy.Rain                     7.039e-03  3.153e-02    0.223 0.823380
Scattered.Clouds              -6.807e-02  3.139e-02   -2.168 0.030126 *
Partly.Cloudy                 -7.041e-02  3.146e-02   -2.238 0.025236 *
Unknown                       -5.116e-02  3.157e-02   -1.621 0.105117
Light.Snow                    -6.390e-02  3.147e-02   -2.031 0.042298 *
```

```
Clear                          -7.573e-02  3.152e-02   -2.403 0.016276 *
Light.Freezing.Rain            -1.023e-01  3.162e-02   -3.235 0.001216 **
Ice.Pellets                    -9.541e-02  3.174e-02   -3.006 0.002647 **
Light.Ice.Pellets              -5.912e-02  3.134e-02   -1.886 0.059244 .
Heavy.Ice.Pellets              -7.698e-02  3.253e-02   -2.366 0.017966 *
Rain                           -7.217e-02  3.138e-02   -2.300 0.021444 *
Light.Drizzle                  -8.521e-02  3.151e-02   -2.704 0.006844 **
Mist                           -5.289e-02  3.161e-02   -1.673 0.094241 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1506 on 1026700 degrees of freedom
Multiple R-squared:  0.8161,   Adjusted R-squared:  0.8161
F-statistic: 4.18e+04 on 109 and 1026700 DF,  p-value: < 2.2e-16
```

5-FOLD cv

```
[1] "***Fold_Number***"
[1] 1
[1] "***Training_Fold_MSE***"
[1] 0.0226186
[1] "***Test_Fold_MSE***"
[1] 0.02294357
[1] "***Fold_Number***"
[1] 2
[1] "***Training_Fold_MSE***"
[1] 0.02263285
[1] "***Test_Fold_MSE***"
[1] 0.02288694
[1] "***Fold_Number***"
[1] 3
[1] "***Training_Fold_MSE***"
[1] 0.02286591
[1] "***Test_Fold_MSE***"
[1] 0.0219583
[1] "***Fold_Number***"
[1] 4
[1] "***Training_Fold_MSE***"
[1] 0.02302202
[1] "***Test_Fold_MSE***"
[1] 0.02133232
[1] "***Fold_Number***"
[1] 5
[1] "***Training_Fold_MSE***"
[1] 0.02227251
[1] "***Test_Fold_MSE***"
[1] 0.02433075
> print(ffold_errors)
           [,1]       [,2]       [,3]       [,4]       [,5]
[1,] 0.02261860 0.02263285 0.02286591 0.02302202 0.02227251
[2,] 0.02294357 0.02288694 0.02195830 0.02133232 0.02433075

 [1] "***Average_Testing_Sets_MSE***"
> print(TtSMSE)
[1] 0.02269038
```

Appendix B: Result for LASSO Regression

```
Call:
lm(formula = Traveling.Time.min. ~ X110.04651 + X110P04650 + 
    X110P04651 + X110.04652 + X110P04652 + X110P16046 + X110.04653 + 
    X110P04653 + X110.04654 + X110P04654 + X110P04655 + X110.04656 + 
    X110P04656 + X110.04657 + X110P04657 + X110.04298 + X110.04299 + 
    X110P04299 + X110P04298 + X110.04300 + X110P04300 + X110.04301 + 
    X110P04301 + X110.04302 + X110P04302 + X110.04303 + X110N04302 + 
    X110.04301 + X110N04301 + X110.04300 + X110N04300 + X110.04299 + 
    X110N04299 + X110N04298 + X110.04298 + X110.04657 + X110N04657 + 
    X110.04656 + X110N04656 + X110.04655 + X110N04655 + X110.04654 + 
    X110N04654 + X110.04653 + X110N04653 + X110.16046 + X110N16046 + 
    X110.04652 + X110N04652 + X110.04651 + X110N04651 + X110.04650 + 
    X110N04650 + Thursday + Sunday + X2.3am + X5.6am + X7.8am + 
    X8.9am + X9.10am + X10.11am + X1.2pm + X2.3pm + X6.7pm + 
    Previous.Average.Speed.mph. + Average.Speed.Afterwards.mph. + 
    Temp.F. + Humidity + Wind.Speed.mph. + Overcast + Heavy.Rain + 
    Scattered.Clouds + Light.Drizzle, data = test1)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.7099 -0.0851 -0.0085  0.0282 13.5421 

Coefficients:
                                Estimate Std. Error  t value Pr(>|t|)    
(Intercept)                    8.288e-01  2.019e-03  410.553  < 2e-16 ***
X110.04651                    -5.667e-02  2.140e-03  -26.477  < 2e-16 ***
X110P04650                    -2.186e-01  2.141e-03 -102.107  < 2e-16 ***
X110P04651                     4.566e-01  2.146e-03  212.735  < 2e-16 ***
X110.04652                     1.700e-01  2.146e-03   79.195  < 2e-16 ***
X110P04652                     2.526e-01  2.151e-03  117.459  < 2e-16 ***
X110P16046                     4.047e-01  2.151e-03  188.113  < 2e-16 ***
X110.04653                    -1.565e-01  2.151e-03  -72.765  < 2e-16 ***
X110P04653                     4.705e-01  2.147e-03  219.107  < 2e-16 ***
X110.04654                     6.831e-01  2.147e-03  318.128  < 2e-16 ***
X110P04654                     3.101e-01  2.149e-03  144.320  < 2e-16 ***
X110P04655                     4.215e-01  2.148e-03  196.198  < 2e-16 ***
X110.04656                    -1.027e-01  2.148e-03  -47.810  < 2e-16 ***
X110P04656                     3.420e-01  2.146e-03  159.365  < 2e-16 ***
X110.04657                    -1.541e-01  2.146e-03  -71.827  < 2e-16 ***
X110P04657                     3.113e-01  2.158e-03  144.252  < 2e-16 ***
X110.04298                     3.594e-01  2.234e-03  160.900  < 2e-16 ***
X110.04299                    -1.652e-01  2.179e-03  -75.814  < 2e-16 ***
X110P04299                     3.379e-01  2.166e-03  156.019  < 2e-16 ***
X110P04298                     8.443e-01  2.172e-03  388.795  < 2e-16 ***
X110.04300                    -1.844e-01  2.203e-03  -83.675  < 2e-16 ***
X110P04300                    -1.655e-01  2.171e-03  -76.219  < 2e-16 ***
X110.04301                    -1.443e-01  2.171e-03  -66.460  < 2e-16 ***
X110P04301                     8.321e-02  2.168e-03   38.379  < 2e-16 ***
X110.04302                    -3.958e-01  2.168e-03 -182.569  < 2e-16 ***
X110P04302                    -9.346e-02  2.187e-03  -42.742  < 2e-16 ***
X110.04303                    -3.176e-01  2.212e-03 -143.553  < 2e-16 ***
X110N04302                    -1.359e-01  1.422e-03  -95.631  < 2e-16 ***
X110N04301                    -5.180e-02  1.416e-03  -36.576  < 2e-16 ***
X110N04300                    -2.203e-01  1.403e-03 -157.048  < 2e-16 ***
X110N04299                     9.052e-02  1.402e-03   64.563  < 2e-16 ***
X110N04298                     8.846e-01  1.402e-03  630.895  < 2e-16 ***
X110N04657                     3.254e-01  1.399e-03  232.559  < 2e-16 ***
```

```
X110N04656                     3.201e-01  1.400e-03  228.654  < 2e-16 ***
X110.04655                     1.161e-01  2.149e-03   54.027  < 2e-16 ***
X110N04655                     4.266e-01  1.398e-03  305.248  < 2e-16 ***
X110N04654                     3.139e-01  1.399e-03  224.352  < 2e-16 ***
X110N04653                     4.700e-01  1.400e-03  335.826  < 2e-16 ***
X110.16046                     1.238e-01  2.150e-03   57.564  < 2e-16 ***
X110N16046                     4.295e-01  1.405e-03  305.722  < 2e-16 ***
X110N04652                     2.187e-01  1.428e-03  153.206  < 2e-16 ***
X110N04651                     4.447e-01  1.397e-03  318.398  < 2e-16 ***
X110.04650                    -6.449e-02  1.397e-03  -46.178  < 2e-16 ***
X110N04650                    -2.699e-01  1.450e-03 -186.136  < 2e-16 ***
Thursday                       1.333e-02  6.398e-04   20.832  < 2e-16 ***
Sunday                        -9.386e-03  7.068e-04  -13.279  < 2e-16 ***
X2.3am                        -7.652e-03  1.191e-03   -6.425 1.32e-10 ***
X5.6am                         1.027e-02  1.089e-03    9.431  < 2e-16 ***
X7.8am                         2.279e-03  1.119e-03    2.037   0.0417 *
X8.9am                         1.004e-02  1.142e-03    8.788  < 2e-16 ***
X9.10am                        7.743e-02  1.140e-03   67.899  < 2e-16 ***
X10.11am                       1.279e-02  1.125e-03   11.372  < 2e-16 ***
X1.2pm                         8.953e-03  1.153e-03    7.762 8.39e-15 ***
X2.3pm                         6.463e-03  1.103e-03    5.857 4.70e-09 ***
X6.7pm                         3.929e-02  1.076e-03   36.515  < 2e-16 ***
Previous.Average.Speed.mph.   -5.353e-03  3.474e-05 -154.074  < 2e-16 ***
Average.Speed.Afterwards.mph. -4.388e-03  3.522e-05 -124.586  < 2e-16 ***
Temp.F.                        4.190e-04  1.948e-05   21.511  < 2e-16 ***
Humidity                      -2.291e-02  1.172e-03  -19.555  < 2e-16 ***
Wind.Speed.mph.               -2.834e-04  3.166e-05   -8.953  < 2e-16 ***
Overcast                       3.749e-03  5.288e-04    7.089 1.35e-12 ***
Heavy.Rain                     9.092e-02  4.861e-03   18.705  < 2e-16 ***
Scattered.Clouds               7.873e-03  8.359e-04    9.419  < 2e-16 ***
Light.Drizzle                 -2.118e-02  1.943e-03  -10.900  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2182 on 1026746 degrees of freedom
Multiple R-squared:  0.6139,   Adjusted R-squared:  0.6139
F-statistic: 2.591e+04 on 63 and 1026746 DF,  p-value: < 2.2e-16
```

5-FOLD cv

```
[1] "***Fold_Number***"
[1] 1
[1] "***Training_Fold_MSE***"
[1] 0.02250041
[1] "***Test_Fold_MSE***"
[1] 0.02367965
[1] "***Fold_Number***"
[1] 2
[1] "***Training_Fold_MSE***"
[1] 0.02299586
[1] "***Test_Fold_MSE***"
[1] 0.02170023
[1] "***Fold_Number***"
[1] 3
[1] "***Training_Fold_MSE***"
[1] 0.02274397
[1] "***Test_Fold_MSE***"
[1] 0.02270292
[1] "***Fold_Number***"
[1] 4
[1] "***Training_Fold_MSE***"
[1] 0.02273968
[1] "***Test_Fold_MSE***"
[1] 0.02271997
```

```
[1] "***Fold_Number***"
[1] 5
[1] "***Training_Fold_MSE***"
[1] 0.02269053
[1] "***Test_Fold_MSE***"
[1] 0.02291777

> print(ffold_errors)
           [,1]       [,2]       [,3]       [,4]       [,5]
[1,] 0.02250041 0.02299586 0.02274397 0.02273968 0.02269053
[2,] 0.02367965 0.02170023 0.02270292 0.02271997 0.02291777

[1] "***Average_Testing_Sets_MSE***"
[1] 0.02274411
```

Appendix C: Result for Generalized Linear Model

```
Call:
glm(formula = Traveling.Time.min. ~ ., family = Gamma(link = "inverse"),
    data = Mar395, start = c(0.21, 1, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
        0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 55, 55, 57, 55.9, 0.96, 29.96, 7, 8.1,
        0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2663  -0.0704  -0.0236   0.0394   3.6441

Coefficients: (3 not defined because of singularities)
                     Estimate Std. Error  t value Pr(>|t|)
(Intercept)         2.771e+01  5.231e-02  529.739  < 2e-16 ***
X110.04651         -2.404e+01  5.350e-02 -449.285  < 2e-16 ***
X110P04650          8.189e-01  9.693e-02    8.449  < 2e-16 ***
X110P04651         -2.772e+01  5.179e-02 -535.244  < 2e-16 ***
X110.04652         -2.671e+01  5.208e-02 -512.846  < 2e-16 ***
X110P04652         -2.708e+01  5.198e-02 -520.919  < 2e-16 ***
X110.16046         -2.623e+01  5.229e-02 -501.671  < 2e-16 ***
X110P16046         -2.761e+01  5.182e-02 -532.867  < 2e-16 ***
X110.04653         -1.066e+01  7.241e-02 -147.284  < 2e-16 ***
X110P04653         -2.775e+01  5.178e-02 -535.811  < 2e-16 ***
X110.04654         -2.802e+01  5.169e-02 -542.072  < 2e-16 ***
X110P04654         -2.732e+01  5.189e-02 -526.398  < 2e-16 ***
X110.04655         -2.621e+01  5.229e-02 -501.177  < 2e-16 ***
X110P04655         -2.759e+01  5.179e-02 -532.769  < 2e-16 ***
X110.04656         -2.054e+01  5.687e-02 -361.129  < 2e-16 ***
X110P04656         -2.738e+01  5.184e-02 -528.151  < 2e-16 ***
X110.04657         -1.564e+01  6.363e-02 -245.819  < 2e-16 ***
X110P04657         -2.732e+01  5.183e-02 -527.034  < 2e-16 ***
X110.04298         -2.742e+01  5.178e-02 -529.560  < 2e-16 ***
X110.04299         -2.372e+01  5.351e-02 -443.294  < 2e-16 ***
```

```
X110P04299            -2.725e+01  5.178e-02 -526.134  < 2e-16 ***
X110P04298            -2.749e+01  5.169e-02 -531.864  < 2e-16 ***
X110.04300            -2.354e+01  5.362e-02 -438.935  < 2e-16 ***
X110P04300            -2.496e+01  5.256e-02 -475.002  < 2e-16 ***
X110.04301            -2.521e+01  5.243e-02 -480.924  < 2e-16 ***
X110P04301            -2.660e+01  5.189e-02 -512.544  < 2e-16 ***
X110.04302            -2.692e+00  8.796e-02  -30.607  < 2e-16 ***
X110P04302            -2.579e+01  5.218e-02 -494.207  < 2e-16 ***
X110.04303             8.461e-01  9.641e-02    8.777  < 2e-16 ***
X110N04302            -2.443e+01  5.222e-02 -467.851  < 2e-16 ***
X110.04301.1           9.516e+00  8.564e-02  111.121  < 2e-16 ***
X110N04301            -2.542e+01  5.200e-02 -488.753  < 2e-16 ***
X110.04300.1          -2.280e+01  5.277e-02 -432.109  < 2e-16 ***
X110N04300            -1.705e+01  5.585e-02 -305.187  < 2e-16 ***
X110.04299.1          -1.011e+01  6.146e-02 -164.464  < 2e-16 ***
X110N04299            -2.652e+01  5.180e-02 -511.993  < 2e-16 ***
X110N04298            -2.805e+01  5.163e-02 -543.292  < 2e-16 ***
X110.04298.1          -1.912e+01  5.460e-02 -350.155  < 2e-16 ***
X110.04657.1          -2.723e+01  5.173e-02 -526.425  < 2e-16 ***
X110N04657            -2.743e+01  5.171e-02 -530.485  < 2e-16 ***
X110.04656.1          -1.527e+01  5.727e-02 -266.630  < 2e-16 ***
X110N04656            -2.742e+01  5.171e-02 -530.338  < 2e-16 ***
X110.04655.1          -2.053e+01  5.387e-02 -381.138  < 2e-16 ***
X110N04655            -2.770e+01  5.168e-02 -535.934  < 2e-16 ***
X110.04654.1          -2.624e+01  5.188e-02 -505.739  < 2e-16 ***
X110N04654            -2.739e+01  5.171e-02 -529.651  < 2e-16 ***
X110.04653.1          -2.805e+01  5.164e-02 -543.207  < 2e-16 ***
X110N04653            -2.780e+01  5.167e-02 -537.916  < 2e-16 ***
X110.16046.1          -1.044e+01  6.138e-02 -170.027  < 2e-16 ***
X110N16046            -2.771e+01  5.169e-02 -536.027  < 2e-16 ***
X110.04652.1          -2.621e+01  5.190e-02 -505.077  < 2e-16 ***
X110N04652            -2.700e+01  5.176e-02 -521.727  < 2e-16 ***
X110.04651.1          -2.698e+01  5.176e-02 -521.157  < 2e-16 ***
X110N04651            -2.773e+01  5.167e-02 -536.689  < 2e-16 ***
X110.04650            -2.412e+01  5.238e-02 -460.469  < 2e-16 ***
X110N04650                    NA         NA       NA       NA
Monday                -7.243e-03  2.625e-03   -2.759  0.00579 **
Tuesday               -2.629e-02  2.706e-03   -9.716  < 2e-16 ***
Wednesday             -2.670e-02  2.788e-03   -9.575  < 2e-16 ***
Thursday              -1.883e-02  2.796e-03   -6.734 1.65e-11 ***
Friday                -1.178e-04  2.560e-03   -0.046  0.96331
Saturday              -2.487e-02  2.738e-03   -9.084  < 2e-16 ***
Sunday                        NA         NA       NA       NA
X0.1am                -8.697e-03  4.547e-03   -1.913  0.05580 .
X1.2am                 2.249e-03  4.689e-03    0.480  0.63148
X2.3am                 7.567e-03  4.683e-03    1.616  0.10617
X3.4am                 2.619e-02  4.501e-03    5.819 5.91e-09 ***
X4.5am                 3.001e-02  4.534e-03    6.619 3.61e-11 ***
X5.6am                 7.561e-02  4.716e-03   16.034  < 2e-16 ***
X6.7am                 3.541e-02  4.604e-03    7.691 1.47e-14 ***
X7.8am                -7.217e-02  4.264e-03  -16.924  < 2e-16 ***
X8.9am                -1.060e-01  4.156e-03  -25.510  < 2e-16 ***
```

```
X9.10am                    -1.757e-01  4.113e-03  -42.725  < 2e-16 ***
X10.11am                    -6.695e-02  4.461e-03  -15.010  < 2e-16 ***
X11.12am                     4.190e-02  5.610e-03    7.469 8.10e-14 ***
X0.1pm                     2.023e-03  5.408e-03    0.374  0.70837
X1.2pm                     1.002e-02  4.676e-03    2.143  0.03210 *
X2.3pm                    -4.836e-03  4.579e-03   -1.056  0.29094
X3.4pm                    -2.566e-02  4.384e-03   -5.853 4.83e-09 ***
X4.5pm                    -1.254e-02  4.435e-03   -2.828  0.00468 **
X5.6pm                     9.013e-02  3.941e-03   22.869  < 2e-16 ***
X6.7pm                     1.082e-01  3.778e-03   28.652  < 2e-16 ***
X7.8pm                     5.788e-03  3.992e-03    1.450  0.14709
X8.9pm                    -1.372e-02  4.239e-03   -3.236  0.00121 **
X9.10pm                    -2.080e-02  4.210e-03   -4.941 7.78e-07 ***
X10.11pm                    -1.037e-02  4.226e-03   -2.455  0.01410 *
X11.12pm                           NA         NA       NA       NA
Previous.Average.Speed.mph.    1.514e-02  7.208e-05  210.009  < 2e-16 ***
Average.Speed.Afterwards.mph.  9.291e-03  7.680e-05  120.977  < 2e-16 ***
Temp.F.                    7.510e-03  1.653e-04   45.440  < 2e-16 ***
Dew.Point.F.               -7.474e-03  1.757e-04  -42.533  < 2e-16 ***
Humidity                   3.774e-01  1.108e-02   34.052  < 2e-16 ***
Pressure.in.               2.097e-02  3.591e-03    5.840 5.21e-09 ***
Visibility.mile.           -8.892e-03  6.898e-04  -12.890  < 2e-16 ***
Wind.Speed.mph.            -2.933e-03  1.330e-04  -22.042  < 2e-16 ***
Gust.Speed.mph.             1.079e-03  8.036e-05   13.425  < 2e-16 ***
Precip.in.                -3.469e-01  4.311e-02   -8.048 8.42e-16 ***
Light.Rain                -8.677e-01  1.139e-01   -7.621 2.53e-14 ***
Overcast                  -8.568e-01  1.139e-01   -7.520 5.50e-14 ***
Mostly.Cloudy              -8.562e-01  1.138e-01   -7.525 5.27e-14 ***
Heavy.Rain                 -9.613e-01  1.138e-01   -8.447  < 2e-16 ***
Scattered.Clouds           -8.529e-01  1.140e-01   -7.479 7.49e-14 ***
Partly.Cloudy              -8.576e-01  1.141e-01   -7.518 5.59e-14 ***
Unknown                    -8.332e-01  1.149e-01   -7.250 4.18e-13 ***
Light.Snow                 -8.302e-01  1.140e-01   -7.282 3.29e-13 ***
Clear                     -8.786e-01  1.143e-01   -7.686 1.51e-14 ***
Light.Freezing.Rain        -9.713e-01  1.145e-01   -8.485  < 2e-16 ***
Ice.Pellets               -9.226e-01  1.150e-01   -8.023 1.03e-15 ***
Light.Ice.Pellets          -5.905e-01  1.132e-01   -5.216 1.83e-07 ***
Heavy.Ice.Pellets          -9.549e-01  1.188e-01   -8.038 9.13e-16 ***
Rain                      -9.361e-01  1.137e-01   -8.229  < 2e-16 ***
Light.Drizzle              -9.299e-01  1.144e-01   -8.130 4.28e-16 ***
Mist                      -9.118e-01  1.151e-01   -7.924 2.29e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.08367908)
```

Appendix D: Result for Sure Independence Screening

(1) Result of Five-Fold Cross-Validation for SIS

SIS Thershold	MSE
2.4912347	0.03108394
2.35756132	0.03108549
2.47221197	0.03116441
2.5279592	0.03141853
2.5163674	0.03142042
2.51904644	0.03142103
2.5271904	0.03142508
2.52748274	0.03142888
2.5301245	0.03143891
2.503974	0.03144582
2.5269482	0.03145405
2.52776839	0.03146279
2.52556347	0.03156491
2.53046627	0.03159899
2.54542345	0.03204769
2.54559945	0.03205124

(1) SIS Result for Threshold Value as "2.4912347"

```
[1] "Screening Threshold"
[1] 2.491235

[1] "Number of Variables after screening"
[1] 108

[1] "Variables Left"
[1] "X110.04651" "X110P04650" "X110P04651" "X110.04652" "X110P04652" "X110.16
046" "X110P16046" "X110.04653" "X110P04653" "X110.04654"
[11] "X110P04654" "X110.04655" "X110P04655" "X110.04656" "X110P04656" "X110.0
4657" "X110P04657" "X110.04298" "X110.04299" "X110P04299"
[21] "X110P04298" "X110.04300" "X110P04300" "X110.04301" "X110P04301" "X110.0
4302" "X110P04302" "X110.04303" "X110N04302" "X110.04301.1"
[31] "X110N04301" "X110.04300.1" "X110N04300" "X110.04299.1" "X110N04299" "X1
10N04298" "X110.04298.1" "X110.04657.1" "X110N04657" "X110.04656.1"
[41] "X110N04656" "X110.04655.1" "X110N04655" "X110.04654.1" "X110N04654" "X1
10.04653.1" "X110N04653" "X110.16046.1" "X110N16046" "X110.04652.1"
[51] "X110N04652" "X110.04651.1" "X110N04651" "X110.04650" "X110N04650" "Mond
ay" "Tuesday" "Wednesday" "Thursday" "Friday"
[61] "Saturday" "Sunday" "X0.1am" "X1.2am" "X2.3am" "X3.4am" "X4.5am" "X5.6am
" "X6.7am" "X7.8am"
[71] "X8.9am" "X9.10am" "X10.11am" "X11.12am" "X0.1pm" "X1.2pm" "X2.3pm" "X3.
4pm" "X4.5pm" "X5.6pm"
[81] "X6.7pm" "X7.8pm" "X8.9pm" "X9.10pm" "X10.11pm" "X11.12pm" "Temp.F." "De
w.Point.F." "Pressure.in." "Wind.Speed.mph."
[91] "Gust.Speed.mph." "Precip.in." "Light.Rain" "Overcast" "Mostly.Cloudy"
  "Heavy.Rain" "Scattered.Clouds" "Partly.Cloudy" "Unknown" "Light.Snow"
[101] "Clear" "Light.Freezing.Rain" "Ice.Pellets" "Light.Ice.Pellets" "Heavy.
Ice.Pellets" "Rain" "Light.Drizzle" "Mist"
```

```
[1] "Removed Variables"
[1] "Previous.Average.Speed.mph." "Average.Speed.Afterwards.mph." "Humidity"
"Visibility.mile."

Call:
glm(formula = Traveling.Time.min. ~ ., family = Gamma(link = "inverse"),
    data = Mar395_SIS_train, start = start.SIS)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1240  -0.1164  -0.0351   0.0660   4.3477

Coefficients: (2 not defined because of singularities)
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          1.952e+09  6.167e+09   0.317   0.7516
X110.04651          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04650          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04651          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04652          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04652          -1.952e+09  6.167e+09  -0.317   0.7516
X110.16046          -1.952e+09  6.167e+09  -0.317   0.7516
X110P16046          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04653          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04653          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04654          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04654          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04655          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04655          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04656          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04656          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04657          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04657          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04298          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04299          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04299          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04298          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04300          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04300          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04301          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04301          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04302          -1.952e+09  6.167e+09  -0.317   0.7516
X110P04302          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04303          -1.952e+09  6.167e+09  -0.317   0.7516
X110N04302          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04301.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04301          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04300.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04300          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04299.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04299          -1.952e+09  6.167e+09  -0.317   0.7516
X110N04298          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04298.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110.04657.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04657          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04656.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04656          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04655.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04655          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04654.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04654          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04653.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04653          -1.952e+09  6.167e+09  -0.317   0.7516
X110.16046.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N16046          -1.952e+09  6.167e+09  -0.317   0.7516
```

```
X110.04652.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04652          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04651.1        -1.952e+09  6.167e+09  -0.317   0.7516
X110N04651          -1.952e+09  6.167e+09  -0.317   0.7516
X110.04650          -1.952e+09  6.167e+09  -0.317   0.7516
X110N04650          -1.952e+09  6.167e+09  -0.317   0.7516
Monday              -3.702e-02  3.675e-03 -10.073  < 2e-16 ***
Tuesday             -1.828e-01  3.951e-03 -46.262  < 2e-16 ***
Wednesday           -9.876e-02  4.127e-03 -23.930  < 2e-16 ***
Thursday            -1.365e-01  3.905e-03 -34.958  < 2e-16 ***
Friday              -1.277e-03  3.624e-03  -0.353   0.7245
Saturday            -6.322e-02  3.860e-03 -16.380  < 2e-16 ***
Sunday                      NA         NA      NA       NA
X0.1am              -2.661e-02  6.516e-03  -4.084 4.42e-05 ***
X1.2am               5.167e-03  6.682e-03   0.773   0.4393
X2.3am               3.317e-02  6.700e-03   4.950 7.42e-07 ***
X3.4am               5.911e-02  6.441e-03   9.178  < 2e-16 ***
X4.5am               1.069e-01  6.516e-03  16.413  < 2e-16 ***
X5.6am               2.172e-01  6.777e-03  32.048  < 2e-16 ***
X6.7am               1.036e-01  6.599e-03  15.704  < 2e-16 ***
X7.8am              -2.457e-01  6.090e-03 -40.350  < 2e-16 ***
X8.9am              -3.607e-01  6.036e-03 -59.759  < 2e-16 ***
X9.10am             -5.295e-01  5.746e-03 -92.162  < 2e-16 ***
X10.11am            -7.592e-02  6.512e-03 -11.660  < 2e-16 ***
X11.12am             9.963e-02  8.052e-03  12.374  < 2e-16 ***
X0.1pm               1.466e-02  7.738e-03   1.894   0.0582 .
X1.2pm               1.051e-01  6.740e-03  15.595  < 2e-16 ***
X2.3pm               8.732e-02  6.584e-03  13.262  < 2e-16 ***
X3.4pm              -5.495e-03  6.313e-03  -0.870   0.3841
X4.5pm               2.767e-02  6.399e-03   4.324 1.53e-05 ***
X5.6pm              -3.800e-02  6.182e-03  -6.147 7.89e-10 ***
X6.7pm              -3.446e-01  5.539e-03 -62.213  < 2e-16 ***
X7.8pm              -6.487e-02  6.020e-03 -10.776  < 2e-16 ***
X8.9pm              -3.717e-02  6.065e-03  -6.129 8.86e-10 ***
X9.10pm             -4.607e-02  6.003e-03  -7.673 1.68e-14 ***
X10.11pm            -2.763e-02  6.050e-03  -4.566 4.97e-06 ***
X11.12pm                    NA         NA      NA       NA
Temp.F.             -5.215e-03  1.507e-04 -34.606  < 2e-16 ***
Dew.Point.F.         5.125e-03  1.335e-04  38.394  < 2e-16 ***
Pressure.in.         6.575e-02  5.540e-03  11.869  < 2e-16 ***
Wind.Speed.mph.     -3.282e-04  1.972e-04  -1.665   0.0960 .
Gust.Speed.mph.      2.517e-03  1.207e-04  20.851  < 2e-16 ***
Precip.in.          -1.059e+00  6.474e-02 -16.362  < 2e-16 ***
Light.Rain          -1.904e+00  1.690e-01 -11.267  < 2e-16 ***
Overcast            -1.880e+00  1.694e-01 -11.099  < 2e-16 ***
Mostly.Cloudy       -1.871e+00  1.694e-01 -11.049  < 2e-16 ***
Heavy.Rain          -2.156e+00  1.694e-01 -12.727  < 2e-16 ***
Scattered.Clouds    -1.939e+00  1.693e-01 -11.449  < 2e-16 ***
Partly.Cloudy       -1.880e+00  1.698e-01 -11.072  < 2e-16 ***
Unknown             -2.029e+00  1.708e-01 -11.877  < 2e-16 ***
Light.Snow          -2.196e+00  1.693e-01 -12.974  < 2e-16 ***
Clear               -1.902e+00  1.700e-01 -11.188  < 2e-16 ***
Light.Freezing.Rain -2.102e+00  1.699e-01 -12.369  < 2e-16 ***
Ice.Pellets         -1.971e+00  1.706e-01 -11.551  < 2e-16 ***
Light.Ice.Pellets   -2.127e+00  1.684e-01 -12.631  < 2e-16 ***
Heavy.Ice.Pellets   -1.883e+00  1.760e-01 -10.697  < 2e-16 ***
Rain                -2.034e+00  1.690e-01 -12.039  < 2e-16 ***
Light.Drizzle       -1.856e+00  1.695e-01 -10.950  < 2e-16 ***
Mist                -1.859e+00  1.704e-01 -10.910  < 2e-16 ***
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.1214687)

    Null deviance: 655067  on 718539  degrees of freedom
Residual deviance:  56023  on 718433  degrees of freedom
AIC: -1772729

Number of Fisher Scoring iterations: 25


[1] "***Average_Testing_Sets_MSE***"
[1] 0.03108394
```

Appendix E: Result for Generalized Linear Model with Inverse Gaussian distribution and "inverse squared" link function

```
glm(formula = Traveling.Time.min. ~ ., family = inverse.gaussian(link = "1/mu^2"),
   data = Mar395_train, start = c(0.21, 1, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
     0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 55, 55, 57, 55.9, 0.96, 29.96, 7,
     8.1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0))

Deviance Residuals:
   Min      1Q   Median      3Q     Max
 -0.607  29.074  49.843  58.048  184.627

Coefficients: (3 not defined because of singularities)
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           3787.712   6092.431  0.622   0.534
X110.04651           -1887.450   8424.678 -0.224   0.823
X110P04650             288.041   9369.111  0.031   0.975
X110P04651           -5906.200   6864.284 -0.860   0.390
X110.04652           -3967.076   7694.626 -0.516   0.606
X110P04652           -4545.563   7511.878 -0.605   0.545
X110.16046           -3418.603   8010.437 -0.427   0.670
X110P16046           -5620.477   7020.937 -0.801   0.423
X110.04653              -6.963   9396.853 -0.001   0.999
X110P04653           -5979.054   6864.485 -0.871   0.384
X110.04654           -6785.043   6273.872 -1.081   0.279
X110P04654           -4980.467   7274.229 -0.685   0.494
X110.04655           -3382.478   7965.749 -0.425   0.671
X110P04655           -5611.112   6836.746 -0.821   0.412
X110.04656            -907.401   8926.935 -0.102   0.919
X110P04656           -5120.242   7039.070 -0.727   0.467
X110.04657            -320.996   9137.985 -0.035   0.972
X110P04657           -4967.352   6893.632 -0.721   0.471
X110.04298           -5198.914   6602.034 -0.787   0.431
X110.04299           -1732.934   7917.459 -0.219   0.827
X110P04299           -4828.698   6613.472 -0.730   0.465
X110P04298           -5173.115   5656.535 -0.915   0.360
X110.04300           -1690.902   7867.506 -0.215   0.830
X110P04300           -2128.635   7300.558 -0.292   0.771
```

```
X110.04301              -2253.953   7273.731  -0.310    0.757
X110P04301               -3550.083   6748.285  -0.526    0.599
X110.04302               -120.811   8239.655  -0.015    0.988
X110P04302               -2717.347   7099.372  -0.383    0.702
X110.04303                273.951   9168.493  0.030    0.976
X110N04302               -2094.407   6274.824  -0.334    0.739
X110.04301.1               95.012   6769.377  0.014    0.989
X110N04301               -2645.324   6272.138  -0.422    0.673
X110.04300.1             -1480.198   6528.039  -0.227    0.821
X110N04300                -602.138   6713.428  -0.090    0.929
X110.04299.1              -196.169   6815.790  -0.029    0.977
X110N04299               -3705.621   6020.401  -0.616    0.538
X110N04298               -6828.113   5169.921  -1.321    0.187
X110.04298.1              -746.279   6851.656  -0.109    0.913
X110.04657.1             -4788.162   5964.280  -0.803    0.422
X110N04657               -5193.585   5863.395  -0.886    0.376
X110.04656.1              -384.914   6938.056  -0.055    0.956
X110N04656               -5174.938   5848.550  -0.885    0.376
X110.04655.1              -998.381   6787.966  -0.147    0.883
X110N04655               -5798.309   5742.899  -1.010    0.313
X110.04654.1             -3430.026   6266.954  -0.547    0.584
X110N04654               -5117.112   5904.857  -0.867    0.386
X110.04653.1             -6912.124   5268.006  -1.312    0.189
X110N04653               -6055.849   5709.982  -1.061    0.289
X110.16046.1              -100.939   7019.766  -0.014    0.989
X110N16046               -5817.635   5830.207  -0.998    0.318
X110.04652.1             -3425.747   6348.782  -0.540    0.589
X110N04652               -4442.987   6008.541  -0.739    0.460
X110.04651.1             -4399.206   6020.286  -0.731    0.465
X110N04651               -5889.027   5665.035  -1.040    0.299
X110.04650              -1994.410   6502.488  -0.307    0.759
X110N04650                      NA         NA      NA       NA
Monday                  104.039   2427.368   0.043    0.966
Tuesday                -85.221   2697.789  -0.032    0.975
Wednesday                 72.530   2670.654   0.027    0.978
Thursday                160.818   2500.618   0.064    0.949
Friday                13.092   2315.599   0.006    0.995
Saturday                26.218   2501.835   0.010    0.992
Sunday                    NA         NA      NA       NA
X0.1am                  6.758   4267.466   0.002    0.999
X1.2am                 34.430   4356.263   0.008    0.994
X2.3am                 38.845   4315.893   0.009    0.993
X3.4am                 29.036   4116.751   0.007    0.994
X4.5am                 22.769   4115.565   0.006    0.996
X5.6am                 56.192   4220.125   0.013    0.989
X6.7am                -13.656   4152.980  -0.003    0.997
X7.8am               -181.401   4014.924  -0.045    0.964
X8.9am               -264.321   3930.568  -0.067    0.946
X9.10am               -211.749   3908.965  -0.054    0.957
X10.11am               -135.834   4158.837  -0.033    0.974
X11.12am                 90.554   5023.351   0.018    0.986
X0.1pm                 48.401   5012.393   0.010    0.992
X1.2pm                 79.503   4344.819   0.018    0.985
X2.3pm                 59.016   4270.680   0.014    0.989
X3.4pm                  2.830   4158.971   0.001    0.999
X4.5am                 22.769   4115.565   0.006    0.996
X5.6am                 56.192   4220.125   0.013    0.989
X6.7am                -13.656   4152.980  -0.003    0.997
X7.8am               -181.401   4014.924  -0.045    0.964
X8.9am               -264.321   3930.568  -0.067    0.946
X9.10am               -211.749   3908.965  -0.054    0.957
X10.11am               -135.834   4158.837  -0.033    0.974
X11.12am                 90.554   5023.351   0.018    0.986
```

```
X0.1pm                      48.401   5012.393   0.010    0.992
X1.2pm                      79.503   4344.819   0.018    0.985
X2.3pm                      59.016   4270.680   0.014    0.989
X3.4pm                       2.830   4158.971   0.001    0.999
X4.5pm                      30.234   4177.909   0.007    0.994
X5.6pm                     194.238   3716.733   0.052    0.958
X6.7pm                     189.498   3507.356   0.054    0.957
X7.8pm                      -2.172   3613.569  -0.001    1.000
X8.9pm                       3.922   3983.642   0.001    0.999
X9.10pm                    -22.506   3956.447  -0.006    0.995
X10.11pm                    -2.080   3948.537  -0.001    1.000
X11.12pm                        NA         NA      NA       NA
Previous.Average.Speed.mph.      51.968     76.271   0.681    0.496
Average.Speed.Afterwards.mph.    39.464     80.125   0.493    0.622
Temp.F.                     35.481    175.259   0.202    0.840
Dew.Point.F.                11.211    198.310   0.057    0.955
Humidity                   894.667  11723.040   0.076    0.939
Pressure.in.               185.819   3497.324   0.053    0.958
Visibility.mile.            -9.968    628.984  -0.016    0.987
Wind.Speed.mph.              1.221    130.669   0.009    0.993
Gust.Speed.mph.              3.842     72.869   0.053    0.958
Precip.in.               -1477.213  46675.966  -0.032    0.975
Light.Rain               -6001.496 109800.737  -0.055    0.956
Overcast                 -6047.534 109806.586  -0.055    0.956
Mostly.Cloudy            -5955.233 109754.010  -0.054    0.957
Heavy.Rain               -6572.162 109806.356  -0.060    0.952
Scattered.Clouds         -6084.366 109690.601  -0.055    0.956
Partly.Cloudy            -6118.419 110078.384  -0.056    0.956
Unknown                  -6040.595 110618.450  -0.055    0.956
Light.Snow               -5971.156 109942.256  -0.054    0.957
Clear                    -6050.332 110151.756  -0.055    0.956
Light.Freezing.Rain      -6202.639 110473.475  -0.056    0.955
Ice.Pellets              -6114.949 110916.119  -0.055    0.956
Light.Ice.Pellets        -5496.676 109589.271  -0.050    0.960
Heavy.Ice.Pellets        -6176.623 113835.441  -0.054    0.957
Rain                     -6127.204 109706.280  -0.056    0.955
Light.Drizzle            -6189.849 110247.265  -0.056    0.955
Mist                     -6202.456 110787.494  -0.056    0.955

(Dispersion parameter for inverse.gaussian family taken to be 95364.09)

    Null deviance:    3695483  on 719490  degrees of freedom
Residual deviance: 1631583326  on 719381  degrees of freedom
AIC: 4613216

Number of Fisher Scoring iterations: 25
```

Appendix F: Result for Generalized Linear Model with Inverse Gaussian distribution and "log" link function

```
glm(formula = Traveling.Time.min. ~ ., family = Gamma(link = "log"),
    data = Mar395_train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.16020  -0.05338  -0.00168   0.04456   2.65717
```

```
Coefficients: (3 not defined because of singularities)
                    Estimate Std. Error  t value Pr(>|t|)
(Intercept)        -1.846e+00  2.868e-03 -643.640  < 2e-16 ***
X110.04651          1.884e+00  2.053e-03  917.741  < 2e-16 ***
X110P04650          1.099e-01  2.061e-03   53.320  < 2e-16 ***
X110P04651          3.202e+00  2.078e-03 1540.591  < 2e-16 ***
X110.04652          2.658e+00  2.082e-03 1276.690  < 2e-16 ***
X110P04652          2.864e+00  2.071e-03 1382.491  < 2e-16 ***
X110.16046          2.541e+00  2.076e-03 1223.806  < 2e-16 ***
X110P16046          3.151e+00  2.060e-03 1529.538  < 2e-16 ***
X110.04653          7.065e-01  2.063e-03  342.424  < 2e-16 ***
X110P04653          3.232e+00  2.072e-03 1559.415  < 2e-16 ***
X110.04654          3.510e+00  2.070e-03 1695.069  < 2e-16 ***
X110P04654          2.981e+00  2.065e-03 1443.683  < 2e-16 ***
X110.04655          2.504e+00  2.067e-03 1211.114  < 2e-16 ***
X110P04655          3.156e+00  2.056e-03 1534.990  < 2e-16 ***
X110.04656          1.419e+00  2.059e-03  689.250  < 2e-16 ***
X110P04656          2.995e+00  2.063e-03 1451.624  < 2e-16 ***
X110.04657          9.486e-01  2.058e-03  460.860  < 2e-16 ***
X110P04657          2.898e+00  2.042e-03 1419.076  < 2e-16 ***
X110.04298          2.887e+00  2.040e-03 1415.418  < 2e-16 ***
X110.04299          1.320e+00  2.034e-03  648.688  < 2e-16 ***
X110P04299          2.578e+00  2.104e-03 1224.977  < 2e-16 ***
X110P04298          3.241e+00  2.065e-03 1569.538  < 2e-16 ***
X110.04300          1.185e+00  2.037e-03  581.774  < 2e-16 ***
X110P04300          1.307e+00  2.065e-03  632.810  < 2e-16 ***
X110.04301          1.505e+00  2.071e-03  726.733  < 2e-16 ***
X110P04301          2.085e+00  2.066e-03 1009.142  < 2e-16 ***
X110.04302         -4.428e-01  2.059e-03 -215.057  < 2e-16 ***
X110P04302          1.734e+00  2.057e-03  843.025  < 2e-16 ***
X110.04303         -2.665e-01  2.183e-03 -122.079  < 2e-16 ***
X110N04302          1.705e+00  1.551e-03 1099.060  < 2e-16 ***
X110.04301.1       -4.529e-01  1.551e-03 -291.993  < 2e-16 ***
X110N04301          2.012e+00  1.588e-03 1266.799  < 2e-16 ***
X110.04300.1        1.448e+00  1.588e-03  911.874  < 2e-16 ***
X110N04300          8.679e-01  1.542e-03  562.782  < 2e-16 ***
X110.04299.1        4.097e-01  1.544e-03  265.361  < 2e-16 ***
X110N04299          2.418e+00  1.546e-03 1563.442  < 2e-16 ***
X110N04298          3.512e+00  1.545e-03 2272.495  < 2e-16 ***
X110.04298.1        1.128e+00  1.599e-03  705.370  < 2e-16 ***
X110.04657.1        2.829e+00  1.601e-03 1766.396  < 2e-16 ***
X110N04657          2.986e+00  1.554e-03 1921.581  < 2e-16 ***
X110.04656.1        9.327e-01  1.554e-03  600.277  < 2e-16 ***
X110N04656          2.970e+00  1.550e-03 1916.579  < 2e-16 ***
X110.04655.1        1.386e+00  1.550e-03  893.815  < 2e-16 ***
X110N04655          3.144e+00  1.564e-03 2009.962  < 2e-16 ***
X110.04654.1        2.460e+00  1.565e-03 1571.295  < 2e-16 ***
X110N04654          2.968e+00  1.562e-03 1899.980  < 2e-16 ***
X110.04653.1        3.508e+00  1.559e-03 2249.877  < 2e-16 ***
X110N04653          3.226e+00  1.559e-03 2068.784  < 2e-16 ***
X110.16046.1        6.602e-01  1.558e-03  423.852  < 2e-16 ***
X110N16046          3.196e+00  1.582e-03 2020.784  < 2e-16 ***
X110.04652.1        2.522e+00  1.581e-03 1594.999  < 2e-16 ***
X110N04652          2.787e+00  1.535e-03 1815.953  < 2e-16 ***
X110.04651.1        2.777e+00  1.535e-03 1808.466  < 2e-16 ***
X110N04651          3.145e+00  1.553e-03 2024.649  < 2e-16 ***
X110.04650          1.874e+00  1.550e-03 1209.656  < 2e-16 ***
X110N04650                 NA         NA       NA       NA
Monday              3.186e-02  7.673e-04   41.527  < 2e-16 ***
Tuesday             3.651e-02  8.389e-04   43.524  < 2e-16 ***
Wednesday           2.955e-02  8.629e-04   34.243  < 2e-16 ***
Thursday            3.884e-02  8.355e-04   46.488  < 2e-16 ***
Friday              3.117e-02  7.241e-04   43.042  < 2e-16 ***
```

```
Saturday                    7.152e-03  7.915e-04    9.036  < 2e-16 ***
Sunday                             NA         NA       NA       NA
X0.1am                     -5.726e-03  1.296e-03   -4.417 1.00e-05 ***
X1.2am                     -4.896e-03  1.313e-03   -3.730 0.000191 ***
X2.3am                     -9.014e-03  1.299e-03   -6.937 4.01e-12 ***
X3.4am                     -5.828e-03  1.236e-03   -4.715 2.42e-06 ***
X4.5am                      1.437e-02  1.234e-03   11.648  < 2e-16 ***
X5.6am                      3.652e-02  1.258e-03   29.027  < 2e-16 ***
X6.7am                     -5.673e-03  1.263e-03   -4.491 7.09e-06 ***
X7.8am                     -1.822e-02  1.290e-03  -14.116  < 2e-16 ***
X8.9am                      2.467e-02  1.310e-03   18.832  < 2e-16 ***
X9.10am                     9.429e-02  1.307e-03   72.120  < 2e-16 ***
X10.11am                    3.766e-02  1.300e-03   28.973  < 2e-16 ***
X11.12am                    5.152e-03  1.505e-03    3.423 0.000620 ***
X0.1pm                      8.927e-03  1.493e-03    5.979 2.25e-09 ***
X1.2pm                      3.125e-02  1.284e-03   24.334  < 2e-16 ***
X2.3pm                      2.309e-02  1.265e-03   18.256  < 2e-16 ***
X3.4pm                      8.656e-04  1.252e-03    0.691 0.489278
X4.5pm                      7.776e-03  1.254e-03    6.200 5.64e-10 ***
X5.6pm                     -4.455e-03  1.240e-03   -3.594 0.000326 ***
X6.7pm                     -3.055e-02  1.240e-03  -24.649  < 2e-16 ***
X7.8pm                     -1.978e-02  1.220e-03  -16.205  < 2e-16 ***
X8.9pm                     -1.757e-02  1.209e-03  -14.535  < 2e-16 ***
X9.10pm                    -1.702e-02  1.203e-03  -14.155  < 2e-16 ***
X10.11pm                   -1.076e-02  1.200e-03   -8.966  < 2e-16 ***
X11.12pm                           NA         NA       NA       NA
Previous.Average.Speed.mph.  -1.544e-02  2.995e-05 -515.378  < 2e-16 ***
Average.Speed.Afterwards.mph. -1.270e-02  2.982e-05 -425.862  < 2e-16 ***
Temp.F.                     7.836e04  7.239e-05   10.824  < 2e-16 ***
Dew.Point.F.               -2.322e-04  7.837e-05   -2.962 0.003055 **
Humidity                    1.631e-02  4.163e-03    3.918 8.92e-05 ***
Pressure.in.                1.866e-02  1.180e-03   15.818  < 2e-16 ***
Visibility.mile.            4.502e-03  2.078e-04   21.670  < 2e-16 ***
Wind.Speed.mph.             3.196e-04  4.144e-05    7.712 1.24e-14 ***
Gust.Speed.mph.            -2.894e-04  2.530e-05  -11.439  < 2e-16 ***
Precip.in.                  5.747e-02  1.602e-02    3.588 0.000333 ***
Light.Rain                 -6.604e-01  3.649e-02  -18.100  < 2e-16 ***
Overcast                   -6.461e-01  3.650e-02  -17.702  < 2e-16 ***
Mostly.Cloudy              -6.505e-01  3.648e-02  -17.831  < 2e-16 ***
Heavy.Rain                 -6.087e-01  3.665e-02  -16.606  < 2e-16 ***
Scattered.Clouds           -6.430e-01  3.649e-02  -17.619  < 2e-16 ***
Partly.Cloudy              -6.431e-01  3.657e-02  -17.583  < 2e-16 ***
Unknown                    -6.240e-01  3.669e-02  -17.005  < 2e-16 ***
Light.Snow                 -6.364e-01  3.658e-02  -17.399  < 2e-16 ***
Clear                      -6.509e-01  3.664e-02  -17.766  < 2e-16 ***
Light.Freezing.Rain        -6.871e-01  3.675e-02  -18.695  < 2e-16 ***
Ice.Pellets                -6.673e-01  3.690e-02  -18.086  < 2e-16 ***
Light.Ice.Pellets          -7.054e-01  3.643e-02  -19.361  < 2e-16 ***
Heavy.Ice.Pellets          -6.394e-01  3.780e-02  -16.916  < 2e-16 ***
Rain                       -6.563e-01  3.648e-02  -17.994  < 2e-16 ***
Light.Drizzle              -6.589e-01  3.663e-02  -17.989  < 2e-16 ***
Mist                       -6.395e-01  3.674e-02  -17.405  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Gamma family taken to be 0.02145065)

    Null deviance: 656541  on 719490  degrees of freedom
Residual deviance:  12602  on 719381  degrees of freedom
AIC: -2857975

Number of Fisher Scoring iterations: 8
```

Bibliography

Arnott, R. (2001). *The Economic Theory of Urban Traffic Congestion: A Microscopic Research Agenda.* Retrieved from https://pdfs.semanticscholar.org/588e/efacacf690e390dba9477caf3e1736a867c2.pdf

Bagnell, J., Kakade, S., NG, A., & Schneider, J. (2004). *Policy Search by Dynamic Programming.* Retrieved from https://www.ri.cmu.edu/pub_files/pub4/bagnell_james_2003_1/bagnell_james_2003_1.pdf

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press.

Bellman, R. (1961). *Adaptive control processes: a guided tour.*

Brown, K. Q. (1979). *Dynamic programming in computer science.* Retrieved from http://repository.cmu.edu/cgi/viewcontent.cgi?article=3346&context=compsci

Bureau of Transportation Statistics. (2017). *Transportation Economic Trends Report.* Retrieved from https://www.bts.gov/browse-statistical-products-and-data/transportation-economic-trends/tet-2017-about-report

CATT Lab. (2018). *RITIS*. Retrieved from http://www.cattlab.umd.edu/?portfolio=ritis

Das, S., & Dey, D. (2007). *On Bayesian Analysis of Generalized Linear Models: A New Perspective.* Retrieved from https://pdfs.semanticscholar.org/9b45/d29215d3db59bc3e45e0b874e7a5dd7e9a71.pdf

Dijkstra, E. W. (1959). *A Note on Two Problems in Connexion with Graphs.* Retrieved from http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf

Elmaghraby, S. (1970). *The Concept of "State" in Discrete Dynamic Programming.* Retrieved from https://ac.els-cdn.com/0022247X70900661/1-s2.0-0022247X70900661-main.pdf?_tid=9b4f606d-b3e5-4ef4-89c1-d81423f81592&acdnat=1523287699_a2d3a718eebf139be77f88beb81e8ab9

Fan, J., & Song, R. (2010). *Sure independence screening in generalized linear models with NP-dimensionality.* Retrieved from http://orfe.princeton.edu/~jqfan/papers/09/GLIMSIS.pdf

Fan, J., & Song, R. (2017). *Sure Independence Screening.* Retrieved from https://orfe.princeton.edu/~jqfan/papers/17/SISReview.pdf

Feng, C., Wang, H., Lu, N., Chen, T., He, H., Lu, Y., & Tu, X. (2014). *Log-transformation and its implications for data analysis.* Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/

Golden, B. L. (1988). *Algorithms for Routing and Scheduling: An Introduction.*

Hawas, Y. E. (2013). *Simulation-Based Regression Models to Estimate Bus Routes and Network Travel Times.* Retrieved from https://www.nctr.usf.edu/wp-content/uploads/2013/12/jpt16.4_Hawas.pdf

IBM Corporation. (2012). *IBM's Smarter Cities Challenge Boston Report.* Retrieved from https://www.smartercitieschallenge.org/assets/cities/boston-united-states/documents/boston-united-states-full-report-2012.pdf

Johnson, J., & Busemeyer, J. (2001). *Multiple-stage Decision-making: The Effect of Planning Horizon Length on Dynamic Consistency*. Retrieved from https://www.users.miamioh.edu/johnsojg/lab/papers/JB_TD01.pdf

Knuth, D. (1997). The Art of Computer Programming. In *Fundamental Algorithms, Third Edition* (pp. 238–243).

Kumar, D. (2013). *Optimization Methods: Dynamic Programming - Introduction.* Retrieved from http://www.nptel.ac.in/courses/105108127/pdf/Module_5/M5L1_LN.pdf

Lay, M. (2009). *The Handbook of Road Technology, 4th edition.*

Li, X. (2012). *Quantitative Analysis of Traffic Efficiency and its influencing Factors.* Retrieved from http://www.qianluntianxia.com/lunwen/344/679341.html

National Association of City Transportation Officials. (2013). *Urban Street Design Guide.* Retrieved from https://islandpress.org/books/urban-street-design-guide

Nelder, J., & Wedderburn, R. (2012). *Generalized Linear Models.* Retrieved from https://pdfs.semanticscholar.org/105f/0072f191a4ceb7c381fc4fd93f460aabf6b1.pdf

Ngo, T. H. (2016). *Generalized Linear Models for Non-Normal Data.* Retrieved from http://support.sas.com/resources/papers/proceedings16/8380-2016.pdf

Powell, W. B. (2008). *What You Should Know About Approximate Dynamic Programming.* Retrieved from http://adp.princeton.edu/Papers/Powell-NRLWhat%20you%20should%20know%20about%20approximate%20dynamic%20programming.pdf

Powell, W. B. (2011). *Approximate Dynamic Programming.*

Rajbhandari, R. (2012). *Exploring the Applicability of Commercially Available Speed and Travel Time Data around Border Crossings.* Retrieved from https://static.tti.tamu.edu/tti.tamu.edu/documents/186051-00001.pdf

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). *Cross-Validation.* Retrieved from https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_565

Smith, D. K. (2003). *Networks and Graphs.* Horwood Publishing, Limited.

Tibshirani, R. (1996). *Regression Shrinkage and Selection via the Lasso.* Retrieved from https://www.jstor.org/stable/2346178?seq=1#page_scan_tab_contents

U.S. Department of Transportation, Federal Highway Administration. (1974). *Highway Functional Classification Concepts, Criteria and Procedures.* Retrieved from http://onlinemanuals.txdot.gov/txdotmanuals/pln/pln_ape.pdf

Wootton, J., García-Ortiz, A., & S.M., A. (1995). *Intelligent transportation systems: A global perspective.* Retrieved from https://www.sciencedirect.com/science/article/pii/089571779500137Q