A Connected Vehicle Based Coordinated Adaptive Navigation System

Wenbo Zhu

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Yinhai Wang, Chair

Xuegang (Jeff) Ban

Don MacKenzie

Program Authorized to Offer Degree:

Civil & Environmental Engineering

University of Washington

Abstract

A Connected Vehicle Based Coordinated Adaptive Navigation System

Wenbo Zhu

Chair of the Supervisory Committee:
Yinhai Wang, Professor
Civil & Environmental Engineering

Real-time data communication and analysis are important to support many smart transportation applications. The connected vehicle (CV) technology leads to a system in which vehicles can communicate with other vehicles, transportation infrastructure, and other devices with communication capabilities. With the increase of data availability, there is a great need for algorithms to process and utilize the data to improve the system efficiency and mobility.

This study developed an adaptive navigation algorithm based on the data collection and communication functions in the connected vehicle system. Specifically, the algorithm will utilize real-time traffic information to adaptively recommend the optimal path considering both the user cost and system impact. To quantify the travel cost associated with different paths, a link cost function is developed to estimate both the link travel time and delay at the downstream intersection.

An empirical intersection delay function is derived from the stochastic queueing theory models. The developed function can support link cost estimation for interrupted traffic flow on local streets, which is a limitation for previous navigation algorithms. Based on the CV communication capabilities, two specific dynamic navigation algorithms have been developed to suggest the optimal paths that dynamically minimize the user cost and system impact, respectively.

The developed navigation algorithms have been implemented in a microscopic simulation model using VISSIM application programming interface (API) functions. Multiple experiments have been conducted to test the CV navigation algorithms in a virtual traffic environment based on the urban street network in downtown Bellevue, WA. Experiment results reveal that CV navigation algorithms are effective in reducing both the user and system cost compared to the static navigation used by non-CVs. The benefits of adaptive navigation algorithms will increase with the CV market penetration, and the maximum benefit is achieved when the CV penetration rate reaches around 60%. In the studied network, the marginal benefit of using the dynamic system optimum navigation over the dynamic user equilibrium navigation is relatively small (e.g., around 1%) comparing with the total user cost. Further experiments show that the developed CV navigation algorithms can work effectively during non-recurrent congestions through properly balancing historical and real-time traffic information.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Now that my doctoral study comes close to the end, I would like to use this opportunity to express my great appreciation to everyone who supported me throughout this journey. I would never be able to complete this work without their advice, help, and love.

First of all, I would like to thank my academic supervisor, Dr. Yinhai Wang for his expert research advice and guidance. He offered great help in finding my thesis direction and progressing research tasks. I am more thankful for the research and teaching opportunities he provided during my time in the lab. These opportunities help me become capable to complete and deliver quality work as an independent researcher. On top of all these things, Dr. Wang has also contributed huge time and enthusiasm in building a productive and supportive research environment, which is essential for the success of every member in the group.

I am sincerely grateful to Dr. Sumit Roy, for serving as the GSR in my doctoral supervisory committee, and Dr. Jeff Ban and Dr. Don MacKenzie, for serving as faculty members on my doctoral supervisory committee and reading committee. They have provided me with valuable advice and suggestions to improve my work from different perspectives. I also appreciate advice and recommendations from Dr. Michele Shaffer from the Department of Statistics in my general exam. Additionally, I am thankful to the Pacific Northwest Transportation Consortium (PacTrans) for funding this research project and supporting my conference travels.

I would like to express my warm thanks towards my friends and fellows in the STAR Lab. We collaborated on many research projects and journal papers. They have offered great help in my study, work, and life. I also thank everyone who participated in my presentations and gave me conductive suggestions.

I feel fortunate in studying at the University of Washington, which provides a welcoming and respectful working and learning environment. During my time at UW, I made a lot of friends and served the student community in many activities and volunteering events. I also received great help from the Career & Internship Center in my job searching process. I am proud to be a UW alumnus and am excited to meet every Husky in the future.

I would also like to use this opportunity to express special thanks to my girlfriend, Yihong, and my cat, Blue. With their company, I have experienced countless moments of surprise, happiness, and affection.

Last but not least, I wish to express my deepest appreciation to my parents. They always stand behind me and give me persistent care and encouragements. With their support, I feel prepared to face any challenges and am ready to target my new career objectives for years to come.

Chapter 1. INTRODUCTION

1.1 BACKGROUND

Past several years have witnessed a major change with regard to data availability in the transportation domain. After decades of technology development in the traffic management systems, vehicle-based technology has become the major focus in the next wave of innovation (Mahmassani, 2016). Connected vehicle systems (CVS) will enable information transfer from vehicle to vehicle (V2V), from vehicle to infrastructure (V2I), and from vehicle to any devices with communication capabilities (V2X). Such a system will provide large volumes of diverse, multi-source, and high-resolution data to support numerous applications in traffic planning, operation, and system optimization.

Figure 1.1 presents a typical CVS considered in this study. The implementation of CVS requires the installation of On-Board Units (OBUs) to vehicles and Road-Side Units (RSUs) to the roadway infrastructure. According to the United States Department of Transportation (USDOT), connected vehicle (CV) technology is readily available in the vehicle manufactural industry (USDOT, 2016). In 2016, the National Highway Traffic Safety Administration (NHTSA) issued a Notice of Proposed Rulemaking (NPRM) that targets to eventually require V2V devices in all new light vehicles. On the infrastructure side, the Vehicle to Infrastructure Deployment Coalition (V2I DC) has created a Signal Phase and Timing (SPaT) challenge that encourages the deployment of V2I devices in at least one corridor or network in each state by January 2020 (NOCoE, 2018). The goal of the challenge is to help state and local transportation infrastructure owners and operators better prepare for vehicles equipped with OBUs as well as test the implementation and impact of V2I applications.

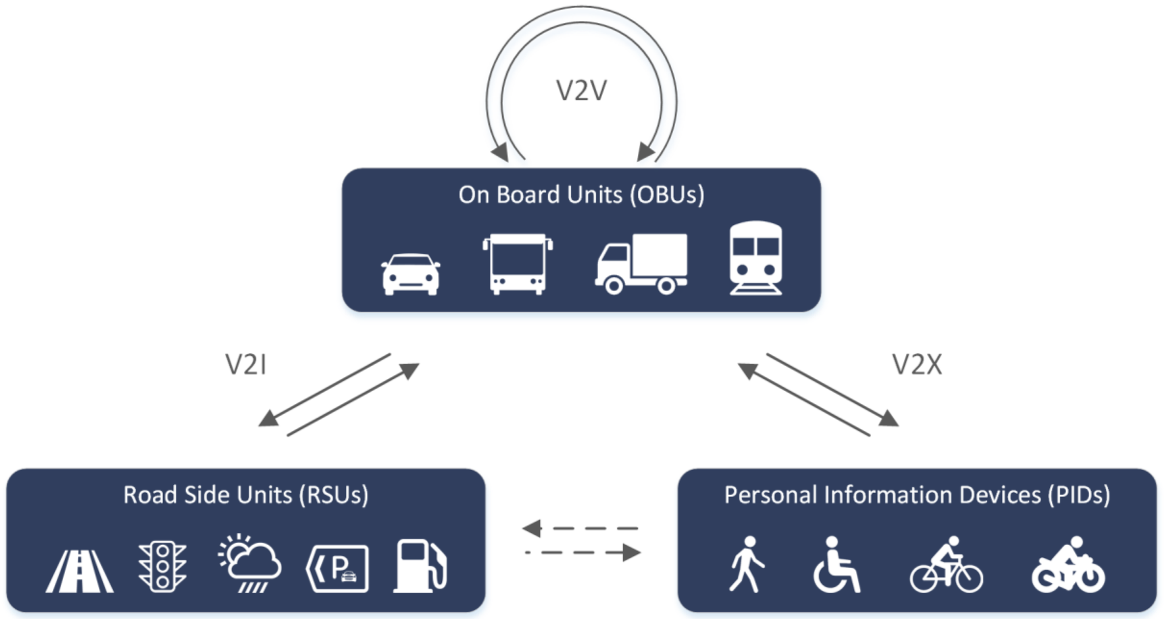


Figure 1.1. Illustration of a typical connected vehicle system.

With all the efforts in the deployment of CVS, however, a gap still exists between the technology readiness and practical applications. Practical challenges have been revealed in the implementation process such as management at scale and cost/benefit assessment. To understand the concerns and challenges before the broad deployment of the CV technology, USDOT has funded three pilot programs to evaluate the feasibility of CVS in New York City, Tampa, and Wyoming (USDOT, 2018). The CVS deployed in these pilot sites include an integration of OBUs, RSUs, and mobile devices, and each site has designed and developed a CV-based application framework that specifically meet the region’s unique transportation needs. One of the important objectives of these pilot project is to measure the impacts of CV applications, in order to ensure that the deployment of CVS will deliver the expected benefits in safety, mobility, efficiency, and environment.

V2I devices provide CVs with data collected and processed from the transportation infrastructure. That said, transportation operators still face challenges collecting and processing traffic information. According to the 2012 National Traffic Signal Report Card, the overall performance of 311,000 traffic signals in the U.S. was disappointingly rated at a level of D+ (NTOC, 2012). The report also assessed detailed grading of these signals in five designated themes, including management, signal operations, signal timing practices, traffic monitoring and data collection, and maintenance. Among these themes, traffic monitoring and data collection was rated the lowest, with a score of F. The implementation of RSUs will significantly improve the data collection capability of traffic signals, but there still exists a great need for methodologies to process and analyze the collected data.

One of the important benefits of CV based applications is to enhance the system mobility and efficiency. In a complicated road network, multiple alternative paths can exist for a particular trip. Previous ways of selecting the optimal path usually rely on the historical traffic data and users' familiarity with the road. These navigation methods usually result in a non-optimal traffic assignment with some roads over-traveled and others less-traveled. If the real-time traffic information can be obtained and utilized, it is possible to develop a more efficient navigation algorithm that can reduce both the user cost and system impact. Such an algorithm can be implemented with the communication capabilities of the CVS, and it is expected that both CVs and non-CVs will benefit from a more efficient traffic assignment in the roadway network.

1.2 PROBLEM STATEMENT

Although a number of existing navigation applications have already been using real-time traffic data, some limitations still exist in the current navigation systems. First of all, the estimation of link travel cost mainly relies on the vehicle speed data. This method usually can generate sufficiently accurate estimates for uninterrupted traffic flow (e.g., freeways and highways with controlled access). But for urban streets, the travel time is mainly controlled by the wait time at signalized intersections. Thus, the vehicle speed data are not enough to estimate the actual travel time on each link. To efficiently navigate in different types of roads, it is necessary to obtain data from both vehicles and traffic control signals so that the algorithm can work consistently for both uninterrupted and interrupted traffic flow.

The current navigation system helps users to identify the optimal path based on certain criteria (e.g., shortest travel time). In such a framework, each user's optimization procedure is independent from others and everyone aims to maximize their own benefit. This will lead the system to a dynamic user equilibrium (DUE) state where no one has the incentive to switch to a different path. But the DUE state is not necessarily the optimal state for the entire system. The total system cost can be further reduced if some vehicles can navigate based on the system cost rather than the individual user cost. The new equilibrium state in which the total system cost is minimized is called the dynamic system optimum (DSO) state. To estimate the total cost of a transportation system, it is necessary to know the traffic volume in each link. In the existing navigation applications, however, such an optimization is not achievable without real-time data communication between vehicles and the infrastructure.

The current navigation applications calculate the optimal route at the start of a trip. Given the randomness in traffic, the recommended route might not continue to be optimal when the vehicle is traveling in the road network. Although some applications can recommend alternative routes based on real-time traffic data, users are usually reluctant to switch routes when the benefit is small. In addition, the uncertainty nature of traffic also makes users skeptical about the estimated benefit of the alternative route. This will result in under-utilization of the real-time traffic data, which leads to inferior navigation results.

To address the aforementioned limitations, this study aims to develop a more efficient navigation algorithm that can adaptively recommend the optimal path based on the real-time traffic information. The whole algorithm is inspired by the rapidly evolving CV technology and the increasing need for system optimization. Such an algorithm will be of significant value to public agencies and private companies willing to efficiently manage a fleet of CVs, and will provide insights to traffic operators and policy makers for traffic planning and system optimization in the CV environment.

1.3 Research Objectives

The goal of this study is to develop an adaptive navigation algorithm based on data collection and communication functions in CVS. Specifically, two types of data sources, infrastructure (including intersections and road links) and connected vehicles, are utilized in the study. It is assumed that connected vehicles can receive real-timed traffic flow and travel time information from the infrastructure and the proposed navigation methodology will utilize such information to determine the real-time optimal path in the road network.

The proposed navigation solution will combine a set of innovative algorithms, including a link cost estimation function, a dynamic shortest-path algorithm based on the probabilistic dynamic programming, and a system optimal navigation algorithm.

As the fundamental of the navigation algorithm, this study will first develop a link cost function with special emphasis on the estimation of intersection delay. A general form of link cost – traffic flow relationship will be derived from traffic flow and queueing theory models. The link cost function is validated in various traffic scenarios with different levels of traffic flow randomness. In addition to quantify the link cost in the navigation procedure, an important use of the link cost function is to estimate the system cost associated with each routing decision. With

CVs receiving the real-time traffic flow data collected by the infrastructure, the expected system cost can be estimated as a combination of the user cost and the additional cost imposed on existing vehicles on the link.

The second layer of the method involves a dynamic searching method for the optimal path based on the real-time traffic information. Different from existing navigation methods in which the optimal path is calculated based on historical traffic data or traffic information at the start of the trip. The dynamic navigation method involves a Bayesian framework to predict future traffic conditions when the vehicle is expected to arrive at the downstream links. The future traffic condition is predicted based on both the historical and real-time traffic data. In the Bayesian framework, the historical traffic pattern is used as the "prior" belief and the real-time traffic data will be incorporated to update the "prior" belief and generate more reliable "posterior" predictions of future traffic flow. Such a framework is more robust under nonrecurrent events when real-time traffic is significantly different from the historical data. The classical shortest-path algorithm is adapted into the form of probabilistic dynamic programming in the sense that the cost of downstream links depends on the previous visited links and the cumulative travel cost.

In terms of the navigation objective, the proposed navigation algorithm supports both user equilibrium and system optimum navigations. While the DUE navigation will lead the system into the DUE state, the DSO navigation is able to achieve a better traffic assignment in terms of reducing the total system cost. The navigation algorithm also allows CVs to adaptively change the routing decisions based on real-time traffic information obtained during the trip, so that the efficient network traffic assignment can be consistently achieved.

Both numerical experiments and microscopic simulations will be conducted to implement and validate the proposed methodology. The experiments will quantify the benefits of the proposed navigation algorithm, as well as the impacts at different CV market penetration rates. The proposed study is of great potential for public sectors and private companies that would operate and optimize a fleet of connected vehicles. The analysis results will also provide insights on communication requirements and policies related to future deployment of connected vehicles and devices.

The following bullet points summarize the key contributions of the project

1. Incorporated the intersection delay into the link cost estimation so that the navigation results are more accurate on local roads;
2. Enabled system optimum navigation function for connected vehicles to reduce the system impact; and
3. Quantified the benefit of the proposed navigation method in experiments based on a real-world traffic environment.

1.4 Study Scope

This section clearly describes the scope of the study to help readers further understand the research objectives and avoid any confusions.

The study focuses on developing a navigation algorithm assuming a CVS has been deployed with data communication and processing functions enabled. The proposed navigation algorithm is implemented in a programming environment and tested in a virtual traffic scenario. The proposed navigation algorithm is designed to be consistent with the prototyped CV messages and applications. However, the study will not produce a software application that is ready to be applied to real-world CVs.

The proposed navigation algorithm targets to find the optimal path under existing signal control plans. The algorithm is validated under both fixed and actuated signal control scenarios. However, the navigation algorithm does not involve coordinatively adjusting traffic signals in react to upcoming CVs. While CV based traffic signal coordination has great potential in reducing the travel cost, the optimization task is significantly different from that of the vehicle navigation and cannot be directly integrated in the same framework.

The proposed navigation algorithm is able to suggest the optimal path under different criteria. In order to quantify the algorithm impact, this study assumes vehicles will follow the navigation guidance when traveling in the road network. Such an assumption can be achieved by autonomous driving functions or policies/incentives related to CV navigation. The expected algorithm impact will provide significant insight for future policy and incentive development.

Additionally, the deployment of the CVS is expected to improve the traffic system in multiple ways. For example, traffic congestion can be mitigated by eco-driving and speed harmonization applications and safety functions can reduce the number of incidents on road. However, this study focuses on analyzing the benefit of a more efficient network traffic assignment resulting from the proposed navigation algorithm. Other CV introduced benefits related to the basic traffic flow characteristics and traffic conditions, also promising, will not be discussed in this study.

Chapter 2. STATE OF THE ART

2.1 CONNECTED VEHICLE SYSTEMS

In a CVS, OBUs and RSUs periodically collet vehicle and infrastructure data, including the position, speed, and status (e.g., brake, acceleration, and deceleration) from vehicles and the intersection geometry, signal control plan (e.g., phase and timing), and signal priority status from the infrastructure. Such information is initially stored as snapshots in its memory buffer. The stored snapshots will be broadcasted and shared with other CV devices within the communication range. Based on snapshots generated from the same vehicle, the trajectory can be reconstructed with the vehicle's position and speed at different time points. Knowledge of traffic control plan can also be obtained from the roadmap constructed from snapshots of the same infrastructure. In this way, CV applications can be developed for both vehicles and the infrastructure to optimize the system from different perspectives.

2.1.1 *Communication Requirements*

In 1999, the Federal Communications Commission (FCC) allocated 75 MHz of bandwidth at 5.9 GHz as for Dedicated Short Range Communications (DSRC) specifically used for traffic safety and mobility applications (FCC, 1999). Compared with other means of communication, DSRC communication has lower latency and is more reliable (e.g., under weather events), which is essential to support safety and mobility functions for fast-travelling vehicles.

The Society of Automotive Engineers (SAE) developed the J2735 dictionary as the recommended standard for DSRC messages (SAE, 2006). Below summarize common message types recommended in the J2735 standard

- BSM (Basic Safety Message) includes the vehicle situational data (e.g., position, heading, speed, etc.). This is the core information used in CV safety applications such as collision warning and driving assist.
- SPaT (Signal Phase and Timing) provides the current signal control status and timing information. Such information is collected from the signal controller and broadcasted by RSUs.
- MAP (Intersection Geometry) provides the layout of the intersection, including geometric information for each lane and lane group.
- SRM (Signal Request Message) is sent by OBUs to RSUs to view the current signal status or request signal pre-emption or priority.
- SSM (Signal Status Message) contains the current signal status and pending pre-emption and priority requests. It is sent by RSUs in response to SRM.
- TIM (Traveler Information Message) is sent by RSUs providing roadway conditions and traffic advice (e.g., congestions, incidents, and events).

Figure 2.1 illustrates a typical CVS, where CVs and RSUs are communicating with different types of message. Note that the traffic in the figure is a mixed flow with CVs and non-CVs, and only CVs are able to send and receive messages.

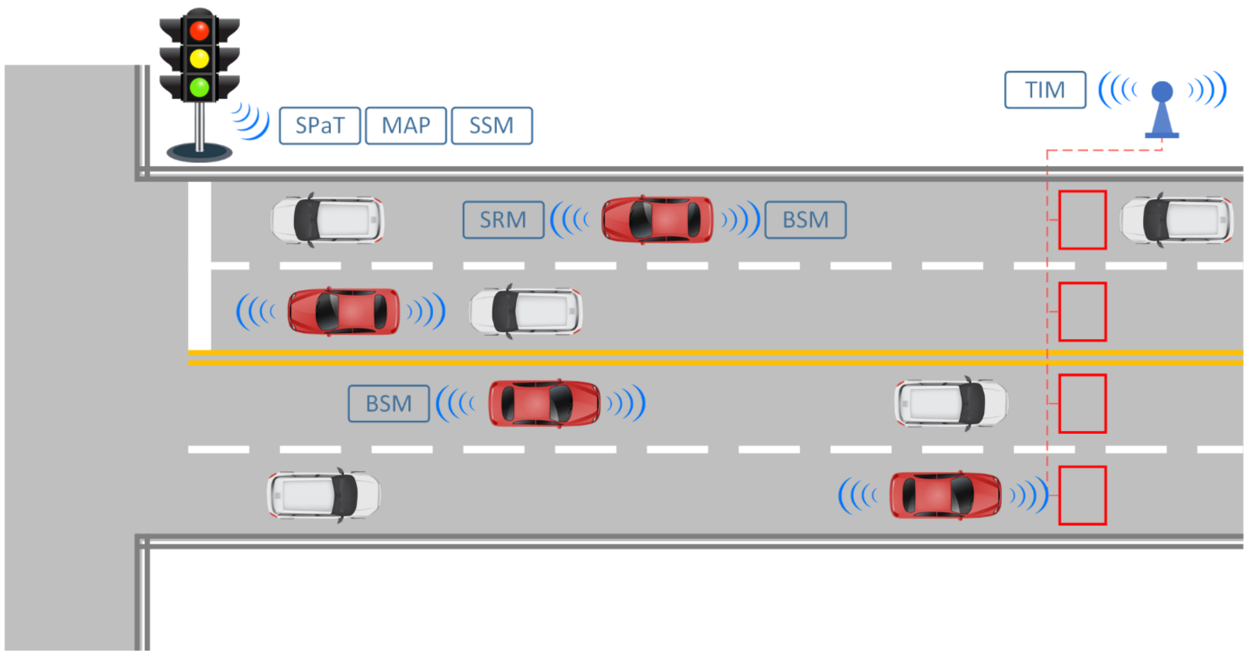


Figure 2.1. Illustration of communications and standard message types in a CVS.

While DSRC was encouraged by U.S. regulators and has been adopted by a number of automakers, in recent years there is an increasing interest of using 5G as the CV communication channel instead of DSRC. For example, Ford recently announced that it will stop supporting DSRC and choose 5G as the communication channel (Turpen, 2019). It is believed that 5G broadband can transmit messages to a longer range and at a faster speed. Additionally, 5G is more open for a variety types of devices as cellular network providers have been heavily investigating in the communication infrastructure upgrades.

2.1.2 *CV Applications*

Various types of applications can be developed based on V2V, V2I and V2X communication of different DSRC messages. The USDOT has assess and prototyped a number of CV applications (USDOT, 2015). These applications can be grouped into seven major categories.

- V2I Safety applications aim to improve traffic safety based by providing connected vehicles with infrastructure data. Such applications include infrastructure-based warning messages (e.g., red light, and work zone warning) and driving assist functions (e.g., curve speed and stop sign gap assist).
- V2V Safety applications provide vehicles with collision alerts or driving assists based on data received from other OBUs. Typical applications include the Forward Collision Warning (FCW) and Left Turn Assist (LTA).
- Agency Data applications aim to use CV technology to enhance the traffic monitoring and data collection for transportation agencies. For example, connected vehicles can report various types of traffic data when traveling, such as origin-destination information, intersection delay, and pavement status.
- Environment applications utilize CV data to reduce the traffic related environmental impact. For example, CV based speed harmonization will improve eco-driving and reduce emission. Smart parking management can also reduce time spent on parking search.
- Road Weather applications aim to provide vehicles with weather related alerts and recommend treatment plans to infrastructure owners and operators.
- Mobility applications aim to improve system efficiency through dynamic management of vehicles, traffic signals and other infrastructure. One typical application is the Advanced Traveler Information System (ATIS), based on which other applications can be developed such as the smart signal management system, adaptive driving assist, and dynamic transit and freight operations.
- Smart Roadside applications aim to improve the freight system efficiency. Example applications include wireless truck inspection and smart truck parking management.

A review of the prototype CV applications shows that majority of CV applications focus on improving traffic safety, primarily because safety is the top concern of local governments and transportation agencies. CV potentials in system mobility and network efficiency, however, are not equivalently emphasized. Additionally, existing CV applications focusing on mobility and efficiency usually provide reactive services (e.g., sending alerts, broadcasting traffic information) rather than proactive solutions. The latter ones are more complicated to build as they often require communication and collaboration among various components in the CVS.

2.2 Vehicle Route Choice Problems

The vehicle route choice problem can be generally classified into routing and navigation problems. But in some previous studies, these two terms were used interchangeably and lacked clear distinction. This section clearly defined the two types of problems and summarizes the related research.

2.2.1 *Vehicle Routing and Navigation*

The Vehicle Routing Problem (VRP) targets to find the optimal route for a vehicle that faces multiple customers or requests at different locations. Based on the specific formulation, the VRP methodology can be applied to various business situations such as truck dispatching, demand-responsive transit, and rideshare services (Dantzig and Ramser, 1959; Ghiani et al., 2003). The goal is to minimize the total routing cost for an individual vehicle or a fleet of vehicles. Depending on the specific problem, the routing cost can be formulated travel distances, travel times, fuel consumption, or a combination of different types of cost.

Based on the classical VRP formulation, further studies have incorporated practical constraints in solving the optimization problem. Such constraints include vehicle capacity limits, service time windows, and service order constraints (e.g., goods must be picked-up first before delivering to the destination) (Pillac et al., 2013). With the improvement of the real-time information availability, dynamic vehicle routing algorithms have also been developed to adaptively change vehicle routes based on the stochastic demand information. Dynamic vehicle routing requires real-time vehicle positioning and management so that an adjusted route can be calculated and executed in react to new service requests revealed en route. A number of variants of dynamic VRP have been developed to suit different business models. For example, a new service request may be accepted or rejected depending on the feasibility and cost of answering the request (Gendreau et al., 1999; Powell, 1996). For VRP with less constraints in vehicle capacity and service delay, researchers have developed methods that adaptively changes vehicle routes to address all new requests (Ichoua et al., 2006; Secomandi, 2000).

VRP targets to find the optimal sequence of locations to visit, and the path or cost between different locations are usually pre-determined. Thus, the vertices between service request locations are usually omitted and the routing cost between any two locations is deterministic. In a complicated road network, however, navigation method needs to be applied to find the optimal path between two vertices.

The Vehicle Navigation Problem (VNP) targets to find the optimal path from the origin to the destination in a complicated network. Similar to VRP, the optimal solution for VNP can be defined with different types of travel cost (e.g., distance, travel time, and fuel consumption). The travel cost associated with each link can be generally regarded as the "distance" of the link. Thus, the optimal path for VNP can be solved by the classical shortest-path algorithm. Multiple algorithms have been developed to find the shortest-path from the origin to the destination. Typical algorithms include the Bellman-Ford algorithm (Bellman, 1958; Ford, 1956), Dijkstra's algorithm (Dijkstra, 1959), and Floyd's algorithm (Floyd, 1962). The following section will introduce the classical and improved shortest-path algorithms, which are commonly used in navigation problems.

2.2.2 *Shortest-Path Algorithms*

The roadway network is represented as a directed graph $G(V, E)$, where $\{V\}$ is the set of vertices representing intersections and $\{E\}$ is the set of edges representing road links. The number of vertices and edges can be expressed by $|V|$ and $|E|$, respectively. Let V_i denote a specific vertex and E_{ij} denote the directed edge from V_i to V_j. The origin and destination vertices are represented by V_O and V_D, respectively.

In this example, we use the travel time as the measure of "distance", and the shortest path corresponds to the path with the minimum total travel time. Let $t_{E_{ij}}$ represent the travel time associated with the edge E_{ij} and T_i is the time of arrival at V_i (i.e., the sum of travel times from all preceding edges). The conventional Bellman-Ford algorithm is an iterative approach. Let F_{V_i} denote the minimum travel time from V_i to V_D, then the system must satisfy the following set of equations

$$F_{V_i} = \min_{j \neq i} \left(F_{V_j} + t_{E_{ij}} \right), \;\; i \neq D \tag{2.1}$$

$$F_{V_D} = 0$$

The basic method to solve for the problem is a successive approximation approach, which can be illustrated as

1. Start with an initial sequence of travel times $\left\{ F_{V_i}^{(0)} \right\}$.
2. In iteration k, set $F_{V_i}^{(k)} = \min_{j \neq i} \left(F_{V_j}^{(k-1)} + t_{E_{ij}} \right)$ and $F_{V_D}^{(k)} = 0$.
3. Repeat step 2 until the result converges to the solution.

The original form of the Bellman algorithm assumes that every two vertices are linked by a direct edge (Bellman, 1958). Thus, the initial travel time sequence can be simply set as the direct travel cost between each vertex and the destination (i.e., $F_{V_i}^{(0)} = t_{E_{iD}}$). The edge cost can be set as infinity (i.e., $t_{E_{ij}} = \infty$) in the case that there is no direct edge from V_i to V_j. The above iterative algorithm will converge after at most $(|V| - 1)$ iterations. In each iteration, the algorithm needs to check all $|E|$ edges. This yields a time complexity of $O(|V| \cdot |E|)$.

Dijkstra's algorithm also solves the single-source shortest-path problem and is typically faster than the Bellman-Fold algorithm. Dijkstra's algorithm finds the shortest path using a double labeling approach illustrated as following steps

1. Initialize the network, set the arrival time to all vertices as infinity ($T_{V_i} = +\infty, \forall i$) and the arrival time to the origin vertex as zero ($T_{V_O} = 0$).
2. Initialize an empty parent hash map P to store the parent vertex of each vertex. For example, $P_{V_j} = V_i$ means V_i is the parent vertex of V_j (i.e., V_i is the immediate upstream vertex of V_j based on the current path searching process).
3. Initialize the "temporarily visited" group of vertices as $\{V_O\}$, and the "unvisited" group of vertices including all other vertices. Create an empty group of "permanently visited" vertices.
4. Find the vertex V_i with the minimum arrival time in the "temporarily visited" group. For all its unvisited neighbor vertices V_j, if we have $T_{V_i} + t_{E_{ij}} < T_{V_j}$, then update vertex V_j with the new arrival time $T_{V_j} = T_{V_i} + t_{E_{ij}}$ and update the parent hash map with $P_{V_j} = V_i$. Move V_j into the "temporarily labeled" group.
5. Move V_i into "permanently labeled" group.

6. Repeat steps 4-5 until the destination vertex V_D is in the “permanently visited” group. The shortest path can then be found through steps 7-10.
7. Initialize the path vertex set as $\{V_D\}$.
8. Let V_i denote the leftmost vertex in the set, if $V_i \neq V_O$, append its parent vertex of P_{V_i} to the left of the path vertex set.
9. Repeat step 8 until V_O is the leftmost vertex in the set. The shortest path consists of directed edges connecting every two adjacent vertices in the set from left to right.

The major computation load of Dijkstra’s algorithm attributes to searching for the minimum distance node in the “temporarily visited” group. Research efforts have been devoted to speed up classical Dijkstra’s algorithm. From the data structure perspective, researchers have been using sorted data structures (e.g., bucket, heap/priority queue) to store the “temporarily visited” group of vertices. The vertex with the smallest distance is always placed first in the group every time after adding or removing a vertex. In this fashion, the shortest-path searching is faster and the computation efficiency is greatly improved. Other studies focused on improving the algorithm from the methodology perspective (Wagner and Willhalm, 2007). Such techniques include goal-directed search (A* algorithm) (Hart et al., 1968), bidirectional search (Luby and Ragde, 1989), and graph partition (Möhring et al., 2007).

The aforementioned shortest-path algorithms are designed for road network with static travel cost. But actual traffic is stochastic due to variations in multiple factors such as the traffic demand, driving behaviors, and traffic control methods. Optimization of the stochastic problem depends on how the stochastic traffic is modeled and the objective function. If the link travel cost is modeled as a random variable with known distribution, the optimal path that minimizes the total expected travel time can be found using the classical shortest-path algorithm (Frank, 1969). With the development of real-time data collection and analysis methods, a number of studies have been focusing on dynamic navigation algorithms based on real-time information revealed during the trip. Eklund et al. developed a dynamic shortest-path algorithm for a changing traffic network and tested the effectiveness of the method in a simulation where the road network is partially destroyed by an earthquake (Eklund et al., 1996). Chen et al. incorporated the real-time data collected from traffic sensors into the navigation algorithm (Chen et al., 2012). The optimal path can be calculated based on different criteria defined by both precise information (e.g., distance, lane type) and fuzzy data (e.g., travel time, traffic volume). Further studies developed adaptive navigation algorithm that can adaptively change the routing decision in response to the new traffic information. Based on the conditional probability distribution of link travel time, Xiao and Lo developed a navigation algorithm that adaptively calculate the optimal path using the en route traffic information (Xiao and Lo, 2014). The algorithm is recalled to update the optimal path when the vehicle collects or receives new traffic data during the trip. Instead of calculating the complete path, Fu developed a closed-loop adaptive navigation method which suggests only the immediate next link (Fu, 2001). The future link time is modeled as a random variable with known mean and variation to quantify the expected travel time to the destination.

2.2.3 *Traffic Assignment*

The previous sections summarized different methodologies that help an individual vehicle to find the optimal path independent from other vehicles' route choices. Thus, these methods are also referred to as independent navigation mechanisms (INMs). In the real-world road network, the link travel cost is a function of the traffic flow. Thus, a particular vehicles route choice will also affect other vehicles using the same route. If all vehicles have perfect knowledge about the network travel cost and chooses the optimal route calculated from the shortest-path algorithm, this leads to a navigation game in which each vehicle targets to minimize its own cost (e.g., travel time). A Nash equilibrium of such a navigation game is also called the user equilibrium (UE) traffic assignment. Under UE, each vehicle chooses its optimal path depending on other vehicles' route choices, and no vehicle can further reduce its travel cost by switching to another path. Rosenthal proved that the Nash equilibria (i.e., the UE traffic assignment) can be reached by solving an equivalent integer programming model (Rosenthal, 1973). Let q_E denote the number of vehicles choosing edge E, and $t_E(q_E)$ is the edge travel time expressed as a function of the edge traffic flow. The vehicle route choices are represented by a set of decision variables defined as

$$x_v^P = \begin{cases} 1 & \text{if vehicle } v \text{ chooses path } P \\ 0 & \text{otherwise} \end{cases}$$

where a path P is a collection of connected edges. Let $\mathbb{P}_v$ denote the set of all possible paths for vehicle v to travel from its origin to its destination. The equivalent integer programming model is expressed as

$$\begin{aligned} \min \quad & \textstyle\sum_{E\in\{E\}} \sum_{y=0}^{q_E} t_E(y) && && (2.2) \\ \text{s.t.} \quad & \textstyle\sum_{P\in\mathbb{P}_v} x_v^P = 1, && \forall\, v \\ & q_E = \textstyle\sum_v \sum_{P\ni E} x_v^P, && \forall\, E \in \{E\} \\ & x_v^P \in \{0, 1\}, && \forall\, v, P \in \mathbb{P}_v \end{aligned}$$

Given a network graph with finite number of edges, solution to (2.2) must exist. It is also proved that any solution gives rise to a Nash equilibrium (and also the UE traffic assignment) (Rosenthal, 1973). However, it is practically difficult to solve the above integer programming problem due to the non-linear objective function and huge number of variables. To overcome such limitations, Du et al. proposed a distributed algorithm in which vehicles sequentially update their route choices to approximate the UE traffic assignment (Du et al., 2015a). The algorithm is inspired by the real-time information exchange capabilities through V2V communication. In the navigation process, each vehicle knows the current route choices of all other vehicles, and the final route decisions is a joint equilibrium considering all vehicles' user optimality. This type of navigation system is generally referred to as the coordinated navigation mechanism (CNM).

Under UE traffic assignment, each vehicle's travel cost is minimized given the route choice decisions of all other vehicles. However, UE is not necessarily the optimal state in terms of minimizing the system cost. If we focus on minimizing the total travel cost from all vehicles, the optimal solution is a system optimum (SO) traffic assignment. Following the notations defined in Equation 2.2, the SO traffic assignment problem can be expressed as

$$\begin{aligned} \min \quad & \textstyle\sum_{E\in\{E\}} q_E t_E(q_E) && && (2.3) \\ \text{s.t.} \quad & \textstyle\sum_{P\in\mathbb{P}_v} x_v^P = 1, && \forall\, v \\ & q_E = \textstyle\sum_v \sum_{P\ni E} x_v^P, && \forall\, E \in \{E\} \\ & x_v^P \in \{0,1\}, && \forall\, v, P \in \mathbb{P}_v \end{aligned}$$

The SO traffic assignment requires vehicles to collaboratively choose their routes. Under SO traffic assignment, some vehicle might be able to further reduce its individual travel cost by switching to a different path, but this behavior will increase the total travel time and thus the system is not optimized.

The SO assignment requires collaboration among all vehicles and perfect knowledge of the entire network. However, such assumptions are impossible to achieve in the real-word conditions. Therefore, a number of route guidance methods have been developed to approximate the SO assignment. For example, some studies developed a multi-agent platform to model the complicated transportation management system (Groot et al., 2015; Kammoun et al., 2014). In such a framework, vehicles receive navigation guidance from a centralized traffic authority, which collects real-time traffic data and calculate the system optimal traffic assignment. Another way towards the system optimum navigation is to directly quantify the system cost at the individual level (Chen and Kempe, 2008; Çolak et al., 2016). The travel cost is formulated as a linear combination of the “private” component (i.e., the individual travel cost) and the “external” component (i.e., extra costs the individual imposed on others). This method does not require a centralized authority to provide navigation guidance, but it assumes that the travel cost can be explicitly expressed as a function against traffic volume and all vehicles have access to real-time network traffic information.

The computational complexity of solving for an equilibrium traffic assignment highly depends on how we model the network traffic flow. Static traffic assignment (STA) problems assume that the traffic flow is time-invariant, and the link flow can be directly represented as the aggregation of all vehicles choosing the link. This yields static UE and SO traffic assignments as illustrated by Equations 2.2 and 2.3, respectively. Dynamic traffic assignment (DTA) problems consider longer analysis time horizons and includes more realistic traffic flow constraints (e.g., flow propagation, queue spillback, etc.). In DTA models, traffic demand and travel times both vary over time, which requires more computational resources and higher data availability. With the development of intelligent transportation systems (ITS), DTA models have been extensively studied in the past several decades. Based on actual flow constraints, DTA models are widely applied to predict future traffic flows and congestions for various traffic planning and facility evaluation purposes.

Note that in DTA models, the traffic assignment equilibriums must also be redefined in the dynamic traffic flow environment. This leads to the dynamic user equilibrium (DUE) and dynamic system optimum (DSO) assignment. One of the earliest DUE studies was done by Friesz et al. (Friesz et al., 1993), who defined the DUE traffic assignment conditions by extending Wardrop’s first principle of static UE traffic assignment (Wardrop, 1952). The DUE definition is slightly rephrased and presented as follows

If, at each instant in time, for each origin-destination pair, the travel costs for all users are identical and equal to the minimum travel cost on all feasible routes, then the dynamic traffic flow pattern is in a dynamic user equilibrium (DUE) state.

Following this definition, research efforts have been devoted to provide analytical DUE solutions (Bliemer and Bovy, 2003; Friesz et al., 1993; Ran et al., 1996). Ban et al. (2008) rewrote the DUE conditions based on the discrete-time link-node traffic flow model and developed an iterative algorithm to solve the non-linear complementarity problem (NCP). It is commonly agreed that DUE is

The DSO traffic assignment targets to achieve the optimal traffic conditions from the system perspective, which is to minimize the total system travel time of all users. One of the earliest pieces of DSO research was completed by Merchant and Nemhauser (Merchant and Nemhauser, 1978), who proposed an original form of the discrete-time DSO model. Following studies have discussed various extensions of the DUE model, such as reformulating the problem as path-based and continuous time models (Ma et al., 2014; Qian et al., 2012). As real-world traffic flows will not follow DSO, researchers have been focusing on analyzing the optimal traffic operation in special scenarios such as emergency evacuations. Thus, many DSO studies have simplified the model to a single-destination problem (Muñoz and Laval, 2006; Shen and Zhang, 2014; Ziliaskopoulos, 2000), so that the total system travel cost can be minimized in such special situations.

In addition to analytical modeling, microscopic simulation provides another important approach to analyze DTA problems. Compared with the analytical approach, which mainly focuses on the macroscopic level, simulation-based DTA models rely on traffic simulators to realize traffic flow constraints at the microscopic or mesoscopic level (Peeta and Ziliaskopoulos, 2001). Thus, simulation-based DTA models have the advantage of circumventing some limitations of the analytical functions and keeping track of individual vehicles or vehicle groups. Additionally, simulation-based DTA models can often be easily deployed in the actual ITS context as the model functions are based on real-time interaction and information exchange with the traffic simulator (Mahmassani, 2001).

An important limitation of simulation-based DTA models is that the model results highly rely on the underlying traffic simulators, which were usually developed and used by a certain group of researchers. Although this does not necessarily influence the research insights, simulation-based DTA models still lack transferability between different traffic simulators as well as between simulators and the real-world traffic network. Additionally, mesoscopic traffic simulators rely on simplified traffic flow models (e.g., cell transmission model), which do not completely overcome limitations of the analytical functions. Microscopic traffic simulators, although based on more realistic car-following models, are often computationally inefficient (Yang and Koutsopoulos, 1996). Thus, many simulation-based DTA models require a trade-off between accuracy and computation speed.

2.3 TRAVEL TIME MODELING

Travel time is a commonly used measure of travel cost. Knowledge of road travel time and its relationship with other traffic flow metrics is important to support the aforementioned navigation algorithms. This section introduces common methods to model travel time for different types of facilities.

2.3.1 *Link Travel Time*

The travel time on a link can be computed as the reciprocal of the space mean speed. The fundamental traffic flow diagram gives the relationship between traffic flow rate, speed and density (Mannering and Washburn, 2013). The speed-flow relationship can be separated into two branches, with one representing the free-flow condition and the other representing the congestion condition. In the free-flow condition, the speed first keeps almost constant at the free-flow speed when the flow rate is low. With the flow rate increasing, the speed slightly decreases until the flow rate reaches the capacity. The congestion branch of the speed-flow curve starts from the origin, as fully congested traffic has zero flow rate and zero speed. As the congested traffic gradually recover, the speed increases with the flow rate and the trend is commonly expressed in a parabolic shape.

Based on the fundamental traffic flow diagrams, travel time increases with the flow rate when the traffic is not congested. Simplified functions have been developed to approximate the relationship between travel time and flow. One commonly used travel time function is developed by the Bureau of Public Roads (BPR). The BPR link travel time function is expressed as

$$t(q) = t_0\left(1 + a\left(\frac{q}{c}\right)^b\right) \tag{2.4}$$

where t_0 is the free-flow travel time, q is the traffic flow rate (volume), and c is the capacity of the link. a and b are function parameters and can vary depending on the specific case studied.

The BPR function describes a general positive relationship between link travel time and the flow rate. However, it is important to note that the BPR function is specifically designed for macroscopic analysis of uninterrupted traffic flow. Thus, BPR function is generally applied in traffic assignment tasks in highway networks for planning purposes. But it is usually not appropriate for modeling interrupted traffic flow on local roads.

2.3.2 *Intersection Delay*

Delay at intersections is a major component of the total travel time for interrupted traffic flow. Unlike link travel time function where extra travel time is caused by congestions, the intersection delay is caused by vehicles stopping and waiting at red signals. Thus, the intersection delay can be calculated by the wait time function in the queueing theory. Variants of queueing models have been developed to address different types of arrival and departure flows. For a typical signalized intersection, the departure flow rate can be modeled as a periodic uniform distribution. During red signal, the departure rate is zero as no vehicle is allowed to travel. When the green signal starts and there exists a vehicle queue, the departure flow rate becomes equal to the saturation flow rate, q_s. When the vehicle queue is cleared, the departure rate reduces and keeps the same as the arrival rate for the remaining green signal. Hence, the departure flow at a signalized intersection is usually modeled as a piecewise deterministic flow in the queueing analysis. Note that here we only consider the case when the traffic is not saturated (i.e., arrival traffic is lower than the saturation flow rate).

To understand the maximum arrival rate that can be served by the given signal control plan, the intersection capacity q_c can be expressed as

$$q_c = \frac{g}{C} q_s \tag{2.5}$$

where g and C are the effective green time and the cycle length, respectively, of the signal control plan. When the vehicle arrival rate is smaller than the intersection capacity, the average vehicle wait time can be estimated using queueing theory functions. However, the specific function to estimate the intersection delay also relies on the type of the arriving traffic. The following sections introduce models to estimate average vehicle delay for deterministic and stochastic vehicle arrivals, respectively.

2.3.3 *Deterministic Vehicle Arrivals*

Deterministic traffic arrival means that the appearance of vehicles is exactly pre-defined. Note that the word "deterministic" does not necessarily infer uniform traffic arrival. The vehicle arrival is deterministic as long as the traffic flow rate and vehicle headway can be exactly described by any time-dependent function. Following this definition, deterministic traffic flow includes uniform arrival rate, piecewise arrival rate, and changing arrival rate with some pre-defined function.

Figure 2.2 shows the traffic flow diagram assuming uniform vehicle arrival at signalized intersections. As the arrival traffic flow is smaller than the intersection capacity, vehicle queues can be fully discharged within each cycle.

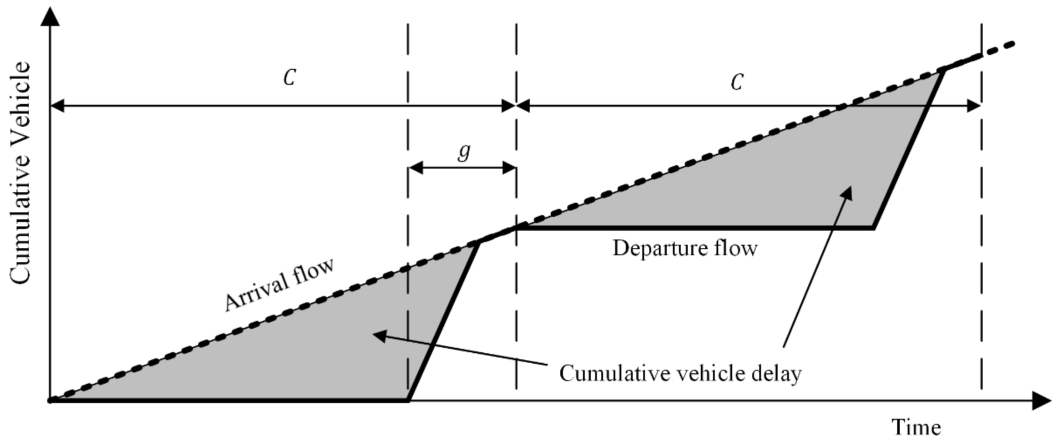


Figure 2.2. Traffic flow diagram for signalized intersection (uniform arrival).

The enclosed area between cumulative arrival departure vehicle curves (i.e., the shaded area in Figure 2.2) represents the cumulative vehicle delay. Let q denote the deterministic arrival flow rate, the average vehicle delay can be explicitly calculated as

$$d_1 = \frac{0.5C(1-g/C)^2}{1-\min(1,X)g/C} \tag{2.6}$$

where $X = q/q_c$ is the degree of saturation. Note that Equation 2.6 only works for the situation when the arrival flow rate is equal or smaller than the intersection capacity (i.e., $q \leq q_c$). When

$q > q_c$, the green signal length is not enough to serve the traffic and the average vehicle delay will increase as the vehicle queue extends over time. HCM introduced an incremental delay term that increases linearly with the analysis period when the intersection is over-saturated (TRB, 2010). The incremental delay is expressed as

$$d_2 = 900T\left((X-1)+\sqrt{(X-1)^2}\right) \tag{2.7}$$

where T is the length of the analysis period. Note that $d_2 = 0$ when $q < q_c$ as there is no undischarged vehicles when the uniform vehicle arrival rate is smaller than the intersection capacity. Combining the two equations, the general average vehicle delay function for deterministic arrival flow is expressed as

$$d = d_1 + d_2 = \frac{0.5C(1-g/C)^2}{1-\min(1,X)g/C} + 900T\left((X-1)+\sqrt{(X-1)^2}\right) \tag{2.8}$$

2.3.4 *Stochastic Vehicle Arrivals*

Real-world traffic flow is not perfectly deterministic. Thus, probabilistic models have been developed to analyze stochastic traffic flow. Previous studies have modeled stochastic vehicle arrivals using different probability distributions, including Poisson and Binomial vehicle arrivals (Dion et al., 2004; Miller, 1963; Newell, 1960; Zhu et al., 2017).

For Poisson vehicle arrivals, the number of vehicles observed (k) in a certain time period (T) is assumed to follow a Poisson distribution, which can be expressed in the form of a probability mass function as

$$P(k) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{2.9}$$

where $\lambda = qT$ is the expected number of observed vehicles during the time period. Given the Poisson distributed vehicle appearance, the time interval between each two consecutive vehicles (i.e., headway, h) will follow an exponential distribution, which is expressed in the form of a probability distribution function as

$$P(h) = \theta e^{-\theta h} \tag{2.10}$$

where $\theta = q$ is the average flow rate. The expected headway is $\mathrm{E}(h) = \theta^{-1}$, which equals to the inversed flow rate.

Binomial vehicle arrival model divides the continuous time period into discrete and equally spaced time points, and it assumes that the arrival of a vehicle can only occur at those time points. At each time point, there is a fixed probability, α, to observe a new vehicle. To model vehicle delay at a traffic signal, the interval between time points is usually set equivalent to the saturation vehicle headway so that during green signals exact one vehicle can depart at one time point. Thus, the relationship between the arriving flow rate and the probability of observing a new vehicle is

$$\alpha = \frac{q}{q_s} \tag{2.11}$$

Compared with the deterministic flow, stochastic traffic arrival at an intersection will cause additional delay due to random cycle failures. The arrival flow may occasionally exceed the intersection capacity during a short period, which leads to undischarged vehicle queues.

In one of the fundamental intersection delay study, Webster derived the equation to calculate the average delay per vehicle under Poisson vehicle arrivals, which can be expressed as (Webster, 1958)

$$d = \frac{C(1-g/C)^2}{2(1-Xg/C)} + \frac{X^2}{2q(1-X)} \tag{2.12}$$

Note that the first term is essentially equal to the uniform delay term (d_1) in the HCM intersection delay function when the traffic is under-saturated (i.e., $q \leq q_c$). The second term calculates the incremental delay caused by occasional unserved vehicles due to random vehicle arrivals. Based on field results, Webster further added an empirical correction term to reduce the estimation by around 10% (Webster, 1958). The revised function is expressed as

$$d = \frac{C(1-g/C)^2}{2(1-Xg/C)} + \frac{X^2}{2q(1-X)} - 0.65\left(\frac{q_c}{q^2}\right)^{\frac{1}{3}} X^{2+\frac{g}{C}} \tag{2.13}$$

Under Binomial vehicle arrivals, the average delay per vehicle can be evaluated using the following equation (Beckmann et al., 1956; Newell, 1960)

$$d = \frac{r^2}{2(1-\alpha)(g+r)} + \frac{g+r}{2[g-\alpha(g+r)]} + O\left(r^{\frac{1}{2}}\right) \tag{2.14}$$

where g, r are effective green and red times, respectively, measured in units of time intervals.

In addition to model stochastic vehicle arrivals using probability distributions, some studies also developed empirical intersection delay function based on field measurements or simulation results. It is important to note that the randomness in traffic can also depend on the flow rate. Under high traffic volumes, for example, vehicles move closer with less variations in the headway. As a result, the traffic flow is approximately uniform distributed. Additionally, under low traffic volumes, the randomness in traffic flow has little impact on the vehicle delay. Thus, the average vehicle delay for stochastic traffic arrival should be close to the results from Equation 2.8 when the flow rate is very high or close to zero. A widely accepted way to model delay is to approximate the deterministic function results when the degree of saturation (X) is very small or very large, and then use a smooth curve to connect these two sections. HCM recommended adding a random delay term to the deterministic incremental delay function to approximate the stochastic incremental delay (TRB, 2010), which is expressed as

$$d_2' = 900T\left((X-1) + \sqrt{(X-1)^2 + \frac{mkI}{q_c T}X}\right) \tag{2.15}$$

where parameters m, k, I are adjustments for flow randomness, signal control plan, and signal propagation, respectively. Note that in a roadway network, the degree of randomness in traffic depends on the upstream traffic control plan. When there exists an upstream signalized intersection, the downstream traffic flow will likely to follow a piecewise distribution similar to the departure flow rate from the upstream intersection. However, given the multi-way intersection and different types of traffic control policies (e.g., turns on red), the actual traffic flow is usually a mix of the deterministic and stochastic vehicle arrivals. An accurate estimation of intersection delay needs to take both traffic flow types into consideration.

Chapter 3. STUDY DATA

3.1 ROADWAY GEOMETRICS

The study focuses on developing a CV based navigation algorithm that works for local road network with signalized intersections. The urban street network from downtown Bellevue is used as the test site of this study. The downtown Bellevue has a grid road system from Main Street (south) to NE 12th Street (north) and from Bellevue Way NE (west) to 112th Ave NE (east). The road network includes 35 intersections and 57 major bi-directional road links. The grid road system is effective for testing navigation algorithms in the sense that it provides sufficient alternative paths for any pair of origin and destination. This road network is also an ideal location to test CV applications as the city is planning to launch a network of connected and autonomous vehicles to serve people's commuting needs in the downtown area (Ryan, 2018). This plan involves deployment of CV devices in the studied network in the very near future.

Geometrical information (e.g., length, curve, and the number of lanes) of the road links was obtained from the City of Bellevue. Figure 3.1 presents the studied road network and the distribution of the road link length. The average link length is 664.4 ft and differences in link lengths are mostly within 10%. Road links in two directions have similar lengths, creating square-sized blocks in the area.

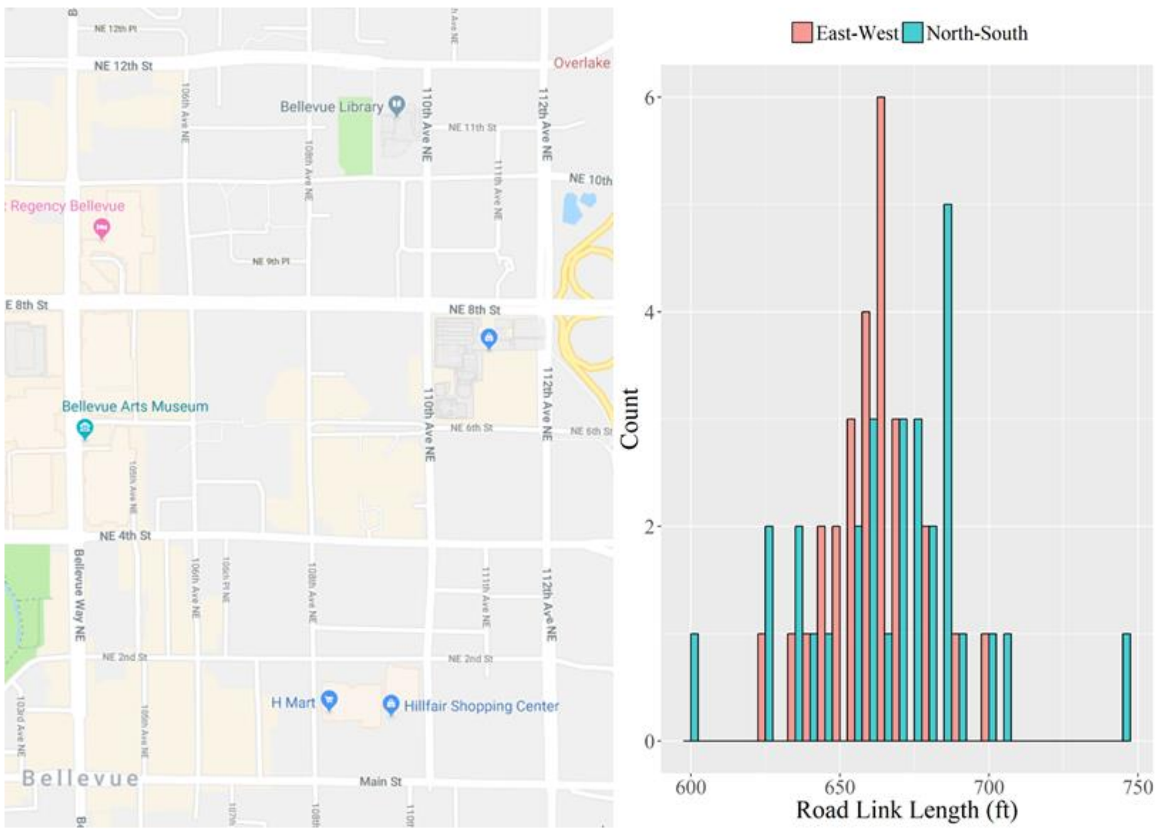


Figure 3.1. Studied roadway network and link length distribution.

3.2 TRAFFIC DATA

To test the navigation algorithm in real-word traffic conditions, this study also include the background traffic data in the analysis. The City of Bellevue has collected traffic count by movements for each intersection during the midday off-peak period (i.e., 1 – 2 PM) in 2017. Figure 3.2a shows the traffic counts for the NE 12th Street (north) and Bellevue Way NE intersection. For each direction, traffic counts were summarized for each of the three movements (i.e., left-turn, through, and right-turn).

Link traffic volumes were calculated by aggregating movement traffic counts in the same direction. Note that there are two ways of calculating the link traffic volume. For a particular link, the traffic volume can be calculated as 1) the flow entering the link from the upstream intersection;

2) the flow exiting the link to the downstream intersection. Results from the two methods do not necessarily agree as some trips may begin or end in the middle of the link (e.g., at roadside buildings and parking lots). This study applied the latter way to calculate the link traffic volume as we are primarily interested in the link travel time, which is mainly controlled by its downstream intersection.

The link traffic volume (veh/h/ln) is shown in Figure 3.2b. According to the figure, traffic conditions are different in different parts of the area. More traffic is observed in the boundary roads and near the Bellevue Transit Center (in the center of the network).

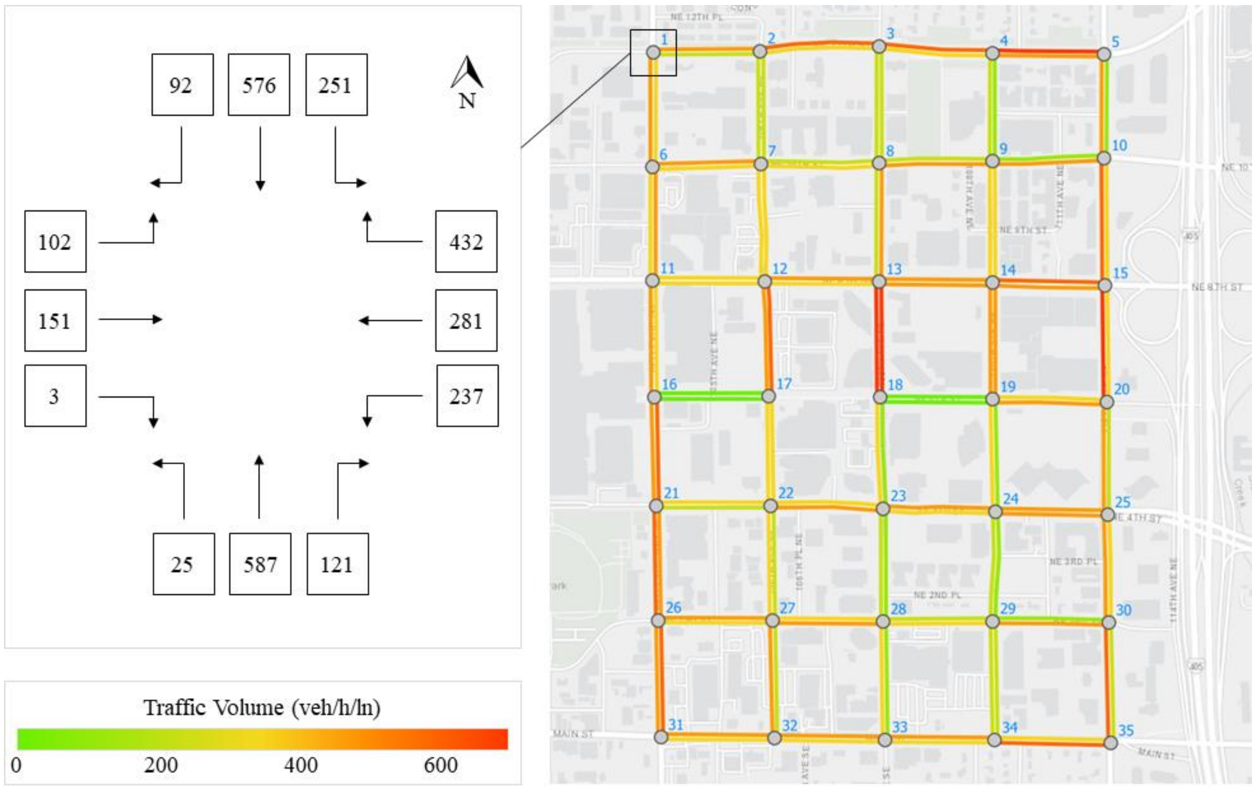


Figure 3.2. Intersection traffic count and link traffic volume of the studied network.

3.3 Simulation Model

A microscopic traffic simulation model was developed by the City of Bellevue based on the actual road geometry and signal control in the studied network. According to the microscopic model, all intersections are controlled by actuated traffic signal with the “gap-out” method. For each intersection, the signal timing parameters can be found in the corresponding VISSIM Ring Barrier Controller (RBC) file. More details about the actuated signal control can be found in Section 4.2.2.

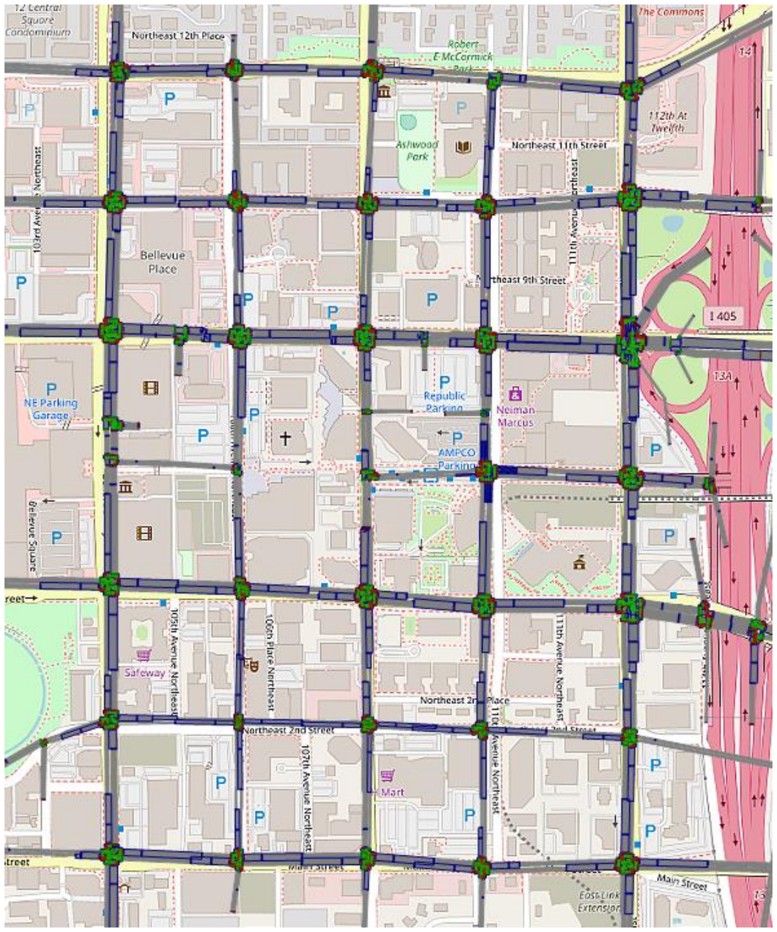


Figure 3.3. Microscopic simulation model of the studied network.

The microscopic simulation model was calibrated by the City of Bellevue with the actual traffic data and has been used for planning and management purposes in the downtown Bellevue area. Specifically, this involves the configuration of vehicle composition, speed distributions, conflict areas, priority rules, and reduced speed areas. Figure 3.4 shows the layout of a particular intersection with related VISSIM objects placed at proper locations based on calibration results.

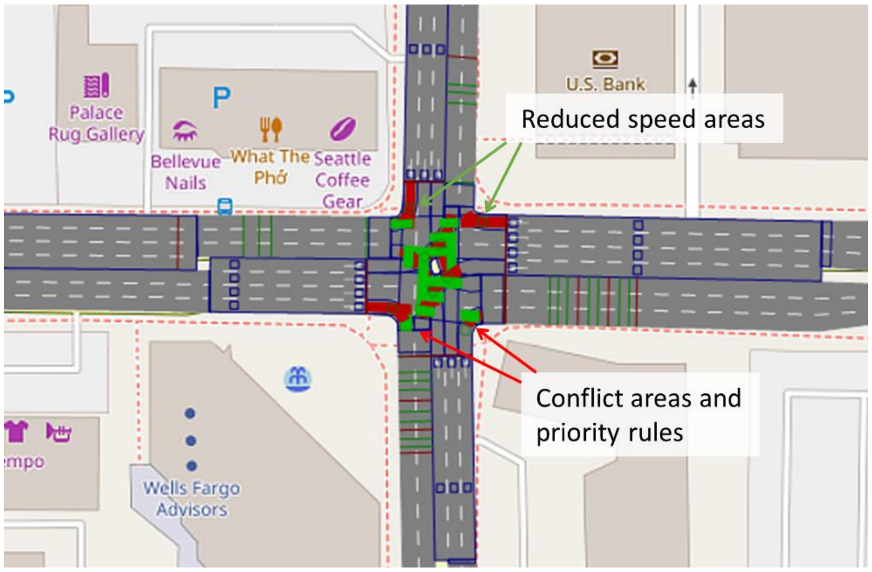


Figure 3.4. Layout of an intersection in the VISSIM model.

The simulation model will be used as the fundamental traffic environment to test CV navigations developed in this study. More details of the data collection and navigation algorithm implementation will be introduced in Chapter 7.

Chapter 4. LINK COST ESTIMATION

Link cost estimation is the fundamental of the navigation algorithm to find the path with the minimum total cost. In this study, travel time is used as the measure of travel cost. As aforementioned, most previous studies used macroscopic travel time functions (e.g., BPR function) to calculate link travel time and validate the navigation algorithm. This method is effective in analyzing the travel cost at the macroscopic scale, such as traffic planning and system design tasks. However, in terms of microscopic traffic analysis (e.g., travel time on local roads and intersection wait time estimation), the BPR function fails to capture the detailed relationship between travel cost and traffic flow rate. To address such limitations, a link cost function is developed to estimate the time spent on a particular link and its downstream intersection. The link cost function describes the general relationship between travel time and traffic flow rate, and the function parameters can be learned from traffic data collected by CV devices. Numerical and simulation experiments have also been conducted to validate the link cost function in various scenarios.

According to the fundamental traffic flow theory, the travel time-flow curve consists of two branches: the uncongested and congested scenarios (Mannering and Washburn, 2013). Figure 4.1a shows the theoretical travel time-flow curve. In the uncongested branch, travel time starts at the free-flow travel time and increases with the flow until the traffic flow reaches capacity. In the congested branch, the increase of traffic density causes flow to drop and travel time to increase. Figure 4.1b shows the observed travel time and traffic volume data from a particular link, and we can also clearly observe the uncongested and congested branches of the travel time-flow curve. Note that the complete curve cannot be expressed as a function, as it allows the same traffic flow at two different travel time values. In this study, we only consider the uncongested branch where travel time is an increasing function of traffic flow. The following sections will also focus on deriving the travel time-flow relationship when the traffic flow has not reached the capacity.

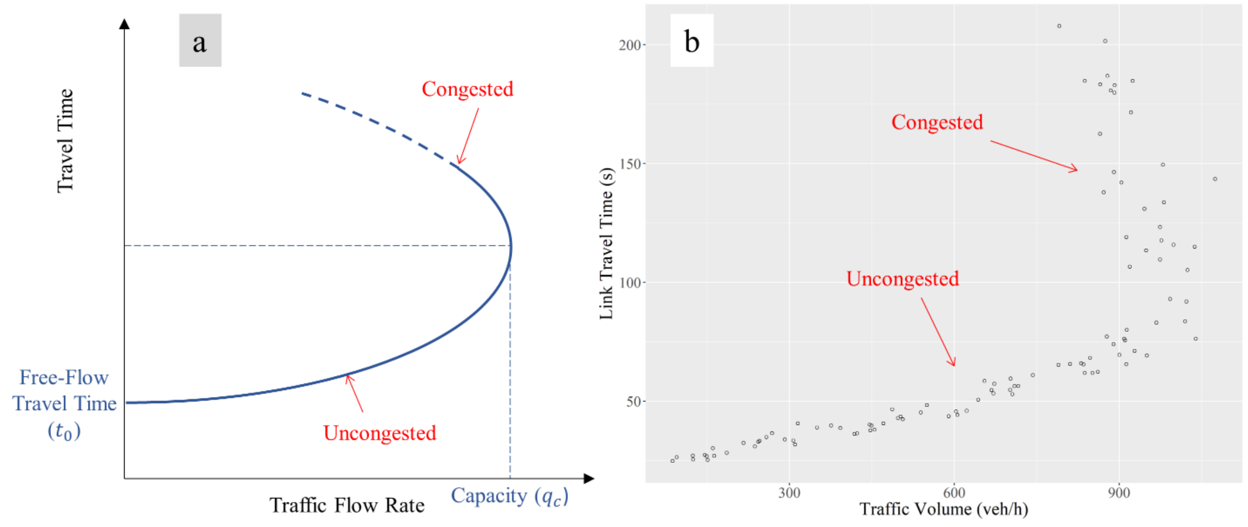


Figure 4.1. Theoretical and observed travel time-flow curves.

4.1 Base Form

This section illustrated the development of the link cost function, with special emphasis on the downstream intersection delay estimation. Thus, the time spent on a particular link consists of two components 1) the travel time on the link; 2) the wait time (delay) at the downstream intersection. The base form of the link cost function is expressed as

$$t(q) = t_0 + d(q) = \frac{l}{v_0} + d(q) \tag{4.1}$$

where $t(q)$ is the time spent on the link. $t_0 = l/v_0$ is the free-flow link travel time under the design speed (l and v_0 are the length of the link and the design speed, respectively). $d(q)$ is the delay at the downstream intersection written as a function of the traffic flow rate, q. While the link travel time can be directly calculated, estimation of the intersection delay is more complicated and the detailed function development will be illustrated in the following sections. Instead of summarizing a practical function with pre-determined parameters, this study focuses on analyzing the general relationship between the intersection delay and the traffic flow rate. A general form of the intersection delay function is derived from the traffic flow and queueing theory, and the function parameters can be learned from the actual traffic data. In this way, the link cost function can utilize the traffic data collected from the CVS and can be uniquely learned for each individual link.

Note that in the base form we assume that link travel time, t_0, is independent of the traffic flow rate. In the real-world situation, however, with the increase of link traffic volume, additional delay may be caused due to congestion-related reasons such as lane switching and shock waves. But in local roads, congestion is not the main reason of delay as the traffic flow is primarily controlled by the signalized intersections. Additionally, it is hard to separate link congestion delay and intersection delay when collecting travel time data. Thus, in the link cost function, the congestion delay is also included in the intersection delay function $d(q)$. In the experiments, the function parameters are learned using the total time spent on the link and the intersection, the congestion delay will also be captured in $d(q)$, which is an increasing function with the traffic flow rate.

4.2 INTERSECTION DELAY FUNCTION

The intersection delay estimation problem has been extensively studied, and multiple different intersection delay functions can be found in the literature. An important limitation of previously functions is that they often rely on pre-defined function parameters learned from previous field studies and lack the flexibility for different types of signal control plan under various traffic conditions. To overcome the limitation, this study focuses on developing an empirical function based on the general relationship between intersection delay and the traffic flow rate. The function parameters will be learned from actual data collected at each individual link and its downstream intersection. Thus, the link cost function will be eventually unique for each link with its particular signal control and traffic condition.

4.2.1 *Pre-Timed Signal Control*

The estimation of average delay per vehicle at pre-timed traffic signals has been extensively studied in previous research. Section 2.3 has summarized major theoretical and empirical intersection delay functions. In this section, we will develop a general form of intersection delay function with empirical parameters that can be learned from actual traffic data.

For uniform vehicle arrivals, the average vehicle delay can be explicitly calculated from Equation 2.6 when the traffic signal is under-saturated. Here we rewrite the function as

$$d_1 = \frac{(C-g)^2}{2C}\left(1-\frac{q}{q_s}\right)^{-1} \quad 0 \le q \le q_c \tag{4.2}$$

For stochastic vehicle arrivals, the situation is more complicated as occasional cycle failures can happen even when traffic signal is under-saturated. The average delay per vehicle can be theoretically estimated if the stochastic vehicle arrivals follow known distributions. For Poisson vehicle arrivals, the average intersection delay can be estimated using Equation 2.13. Here we rewrite the equation for notation consistency

$$d = \frac{(C-g)^2}{2C}\left(1-\frac{q}{q_s}\right)^{-1} + \frac{q}{2q_c^2}\left(1-\frac{q}{q_c}\right)^{-1} - O\left(q^{\frac{4}{3}}\right) \tag{4.3}$$

For Binomial vehicle arrivals, the average intersection delay can be estimated using Equation 2.14. Here we also rewrite the equation with consistent notations

$$d = \frac{(C-g)^2}{2C}\left(1-\frac{q}{q_s}\right)^{-1} + \frac{C}{2g}\left(1-\frac{q}{q_c}\right)^{-1} + O\left((C-g)^{\frac{1}{2}}\right) \tag{4.4}$$

Note that HCM also has developed the incremental delay function for stochastic vehicle arrival (Equation 2.15). But the HCM function is a practical delay relationship which uses an arbitrary function to fit the field observed delay data. This study, on the other hand, will focusing on summarizing theoretical delay models based on which an empirical delay function can be developed.

Comparing the two stochastic intersection delay models, some common features can be summarized as below

- The first term of stochastic delay is the same as the uniform delay, d_1;
- The trend of the incremental delay is mainly controlled by $\left(1-\frac{q}{q_c}\right)^{-1}$;
- Empirical terms with lower orders than the incremental delay are included in both models.

While the first feature is intuitive, the second feature can be explained considering the extreme condition. When the stochastic arrival flow rate approaches the intersection capacity (i.e., $q \to q_c$), the average vehicle delay will grow to infinity. This is because undischarged vehicle queues caused by occasional cycle failures are expected to carry over from cycle to cycle forever as the signal timing plan is just sufficient to serve future traffic. Thus, the average vehicle delay will keep increasing as the undischarged vehicle queue grows.

Based on the common features summarized from previously stochastic intersection delay functions, we developed a general function that describes the overall trend of average delay per vehicle under stochastic vehicle arrivals, which is expressed as

$$d = d_1 + a\left(1-\frac{q}{q_c}\right)^{-1}, \quad 0 \le q < q_c \tag{4.5}$$

where d_1 is the uniform delay as expressed by Equation 4.2. Note that instead of relying on a pre-determined rate, we introduce a function parameter, a, to describe the trend of incremental delay caused by random vehicle arrivals. The function parameter can be empirically learned from actual travel time data to ensure that the function results match the field measurements. As a result, it is unnecessary to include an empirical term to further adjust the delay results.

It is important to note that both Poisson and Binomial delay functions (Equations 4.3 and 4.4) assume completely stochastic vehicle arrivals and infinite study period (hence the delay approximate infinity when the traffic flow rate reaches the capacity). However, the real-world traffic flow is usually a mix of deterministic and stochastic flows. As a result, the actual intersection delay is smaller than the results calculated from theoretical stochastic delay functions. To further consider that only part of the traffic flow is stochastic, an additional parameter, b, is added into the delay function. The revised function is expressed as

$$d = d_1 + a\left(1 - b\frac{q}{q_c}\right)^{-1} \tag{4.6}$$

The new parameter b represents the level of stochasticity in the traffic flow, and ranges between 0 and 1 when the traffic flow is between uniform and stochastic flows. The value of b can also be estimated based on actual travel time measurements. Compared with traditional intersection delay functions, the above function has the adaptivity to various scenarios with different parameter settings:

- If $a = 0$, the function equals to the theoretical uniform delay function;
- If $a \neq 0, b = 0$, the function equals to uniform delay function with a fixed empirical correction factor;
- If $a > 0, b = 1$, the function approximates the theoretical stochastic delay function with the value of a properly estimated.

Consider that the real-world traffic flow is between the uniform and stochastic scenarios, the developed delay function can adapt to different cases with parameters learned from actual traffic data. As parameter b is usually smaller than 1, the delay model allows occasional oversaturation (i.e., q is slightly higher than q_c) for short time periods, which can potentially be observed in the actual data. Thus, the constraint of traffic flow $q < q_c$ is relaxed. But it is unlikely to observe traffic flows significantly higher than the capacity.

One advantage of the developed delay function is that empirically learning the value of b can handle errors in the intersection capacity estimation. Given the complexity in intersection geometry and signal control, it is difficult to reach an accurate estimation of the capacity for a particular direction or movement. When the estimation of q_c is inaccurate, the value of b can be adjusted accordingly to ensure that the delay function fits actual data. Ideally, with the accurate capacity, b is in the range of 0 and 1 with 1 corresponding to completely stochastic vehicle arrivals. But if the capacity estimation is inaccurate, for example, if we overestimate the intersection capacity, we might have the empirical estimate of b greater than 1 to balance the inaccurate capacity value.

All previous delay functions are based on a one-lane traffic signal. Given the number of lanes n for a particular movement at the intersection, the final form of the multi-lane intersection delay function is

$$d = d_1 + a\left(1 - b\frac{q}{n \times q_c}\right)^{-1} \tag{4.7}$$

where q_c now specifically represents the per-lane traffic capacity.

4.2.2 *Actuated Signal Control*

The last section introduces the development of the general average delay function for multi-lane intersection with pre-timed traffic signal control. The intersections in the studied network, however, are controlled by actuated signal control plan. The layout of traffic detectors for a typical actuated signal is illustrated in Figure 4.2. Two vehicle detectors are placed in each lane: the presence detector is placed at the stop line to detect if there is any vehicle waiting during the red time, and the passage detector is placed a short distance (e.g., around 40 ft in this study) from the signal head to actuate green signal extension upon arriving vehicle in the current phase.

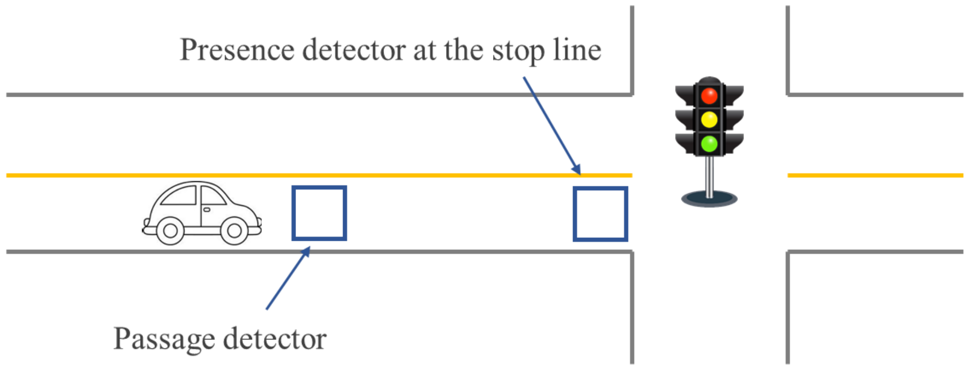


Figure 4.2. Detector layout for a typical actuated signal control.

The green time for an actuated signal control is determined by three control parameters: minimum green interval, maximum green interval, and vehicle extension (or passage time). The minimum green interval is the shortest period that the signal must remain green to satisfy drivers' expectation of green signal length and pedestrian crossing needs. The maximum green interval marks the maximum amount of time the green signal can be displayed from the moment that a waiting vehicle in the conflicting phases is firstly detected. The vehicle extension controls the amount that the green time is extended after reaching the minimum green time. If a new vehicle arrives at the passage detector within the current green signal indication, the green time is extended by an additional vehicle extension (to the maximum of the maximum total green time period). Figure 4.3 illustrates the procedure of how green time is determined by the signal control parameters. In the signal timing design, the vehicle extension is mainly determined by the maximum allowable headway. By adjusting the vehicle extension time, we can control the maximum allowable headway that could actuate consecutive green signal extensions. Any vehicle gaps higher than the maximum allowable headway will terminate the phase. Thus, this is also referred as "gap-out" actuated operation in the signal control terminology.

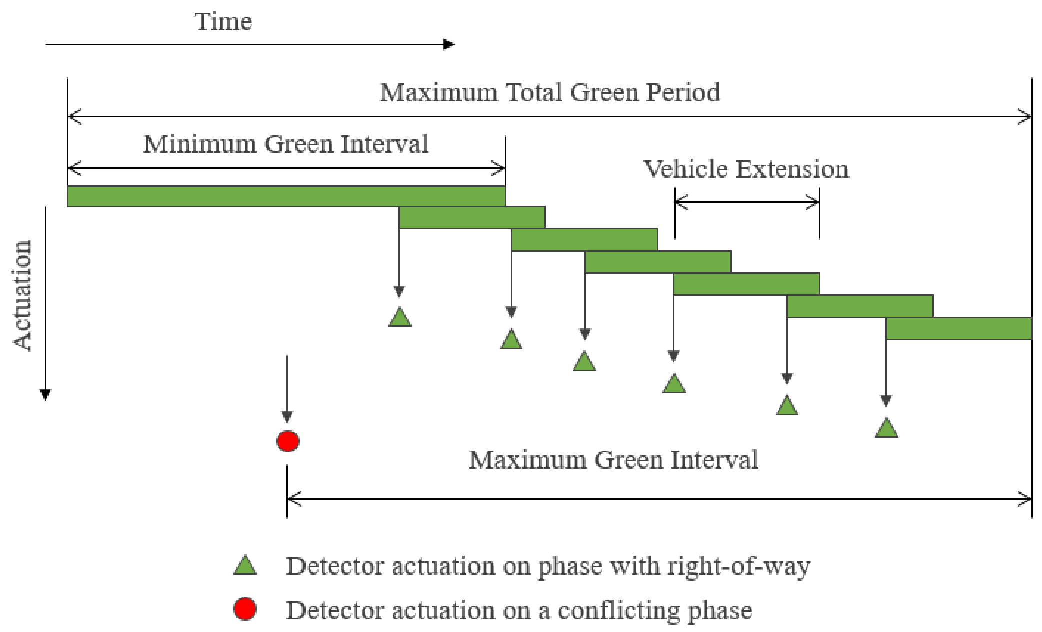
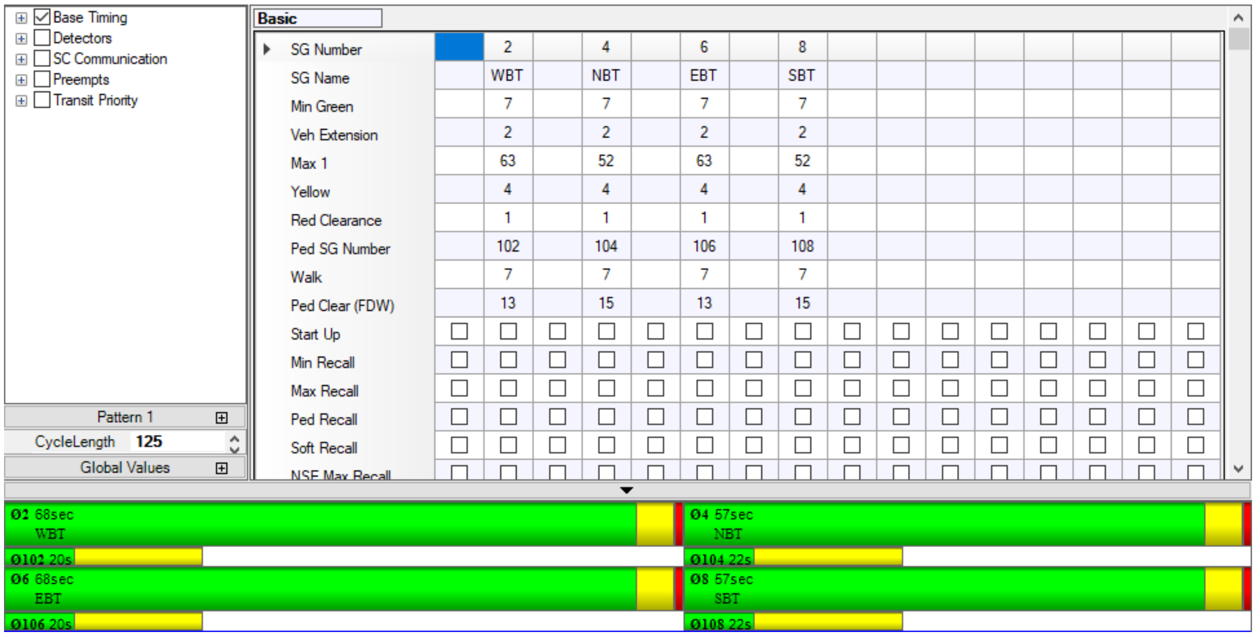

Time
Maximum Total Green Period
Minimum Green Interval
Vehicle Extension
Actuation
Maximum Green Interval
Detector actuation on phase with right-of-way
Detector actuation on a conflicting phase

Figure 4.3. Illustration of actuated signal control parameters.

Actuated signal control allows the green signal length to change adaptively in response to traffic demand. Compared with pre-timed control, actuated signal control usually has lower delay at the cost of installing and maintaining traffic detectors (Koonce and Rodegerdts, 2008). Due to the non-fixed signal timing, previous studies did not specifically develop the theoretical delay model for actuated signals. This study will further revise the empirical delay function through developing the theoretical delay model for actuated signals.

As aforementioned in Chapter 3, all intersections in the studied network are controlled by actuated signal plans, and actuated signal control parameters can be obtained in the VISSIM RBC interface. Figure 4.4 shows the RBC interface of a particular actuated signal control plan. Note that the intersection delay function developed in the previous section is based on pre-timed signal control. As the studied network is controlled by actuated signals, the intersection delay function needs to be further adjusted to consider the changing green period.



☑ Base Timing
☐ Detectors
☐ SC Communication
☐ Preempts
☐ Transit Priority

Pattern 1
CycleLength 125
Global Values

Basic

SG Number		2		4		6		8								
SG Name		WBT		NBT		EBT		SBT								
Min Green		7		7		7		7								
Veh Extension		2		2		2		2								
Max 1		63		52		63		52								
Yellow		4		4		4		4								
Red Clearance		1		1		1		1								
Ped SG Number		102		104		106		108								
Walk		7		7		7		7								
Ped Clear (FDW)		13		15		13		15								
Start Up	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Min Recall	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Max Recall	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Ped Recall	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Soft Recall	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
NSE Max Recall	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Figure 4.4. RBC interface of an actuated signal control plan in the studied network.

Equation 4.7 expresses the pre-timed intersection delay as two components: the uniform delay term (d_1) and the stochastic incremental delay term. Here we assume that the stochastic incremental delay for the actuated signal control still follows the trend of $(1 - b \times q/(n \times q_c))^{-1}$ because:

- The stochastic incremental delay mainly occurs when the traffic demand is high; and
- Under high traffic demand, the actuated signal operation is mainly controlled by the maximum green time and performs very similar to the pre-timed signal control.

Through learning empirical parameters based on the field data, the stochastic incremental delay term can automatically adjust to actuated signal control operation given the stochastic delay follows the same trend. The uniform delay function, however, is derived from fixed green signal length and needs to be adjusted for actuated signal operation. In the following paragraphs of this section, we will derive the function to estimate the uniform delay term for actuated signal control plan.

The uniform intersection delay is defined as the average wait time per vehicle under uniform vehicle arrivals. As the vehicles arrive at a fixed rate, the actuated signal operation will also follow some specific pattern. For example, when the vehicle arrival rate is given (i.e., the uniform traffic flow rate is known), the total green period can also be determined as each signal cycle will face exactly the same vehicle arrival pattern. Calculating the total green period requires analyzing the exact number of vehicle extensions being actuated, and this is directly related to the vehicle arrival rate in the traffic flow and the vehicle extension time in the actuated signal settings.

According to Figure 4.4, the actuated signal control uses two seconds as the vehicle extension time. Further investigation of the VISSIM model reveals that this vehicle extension value has been used in all intersections in the studied network. Note that two seconds is approximately the saturation vehicle headway, and this indicates that the actuated signal operation will only extend the green indication period if the next vehicle closely follows the proceeding vehicle at the saturation headway. Under uniform vehicle arrival assumption, this can only occur when queueing vehicles are being discharged. Thus, in an actuated traffic signal with the vehicle extension time equal to the saturation vehicle headway, the green signal period will be adjusted to only serve the waiting vehicles for each phase. Therefore, the balance of traffic flow can be expressed as

$$q_s g = q(g + r) \tag{4.8}$$

where the left part represents the departure traffic, which equals to the saturation flow rate (q_s) multiply the effective green time (g); and the right part of the equation represents the arrival traffic, which equals to the traffic flow rate (q) multiply the cycle length ($C = g + r$). This gives the effective green time as a function of the traffic flow rate

$$g = \frac{q}{q_s - q} r \tag{4.9}$$

Equation 4.9 gives the actuated signal green period as an increasing function of the traffic flow rate. Note that in the actuated signal control plan, the green indication time is also controlled by the minimum and maximum green signal period. Let g_{min} and g_{max} denote the minimum and maximum effective green period, respectively, the effective green time is constrained by

$$g_{min} \leq g \leq g_{max} \tag{4.10}$$

This gives the constraint of traffic flow rate

$$\frac{g_{min}}{g_{min}+r} q_s \leq q \leq \frac{g_{max}}{g_{max}+r} q_s \tag{4.11}$$

Take Equation 4.9 into the uniform delay function (Equation 4.2), the average delay per vehicle can be estimated as

$$d_1(q) = \frac{r^2}{2(g+r)} \left(1 - \frac{q}{q_s}\right)^{-1} = \frac{r}{2} \left(\frac{q_s - q}{q_s}\right) \left(1 - \frac{q}{q_s}\right)^{-1} = \frac{r}{2} \quad \frac{g_{min}}{g_{min}+r} q_s \leq q \leq \frac{g_{max}}{g_{max}+r} q_s \tag{4.12}$$

This gives the uniform delay of the actuated signal control as a constant (i.e., half of the effective red time) when the green time is between the minimum and maximum green period. Recall that $\frac{g_{max}}{g_{max}+r} q_s = \frac{g_{max}}{C} q_s$ is essentially the capacity (q_c) for the actuated signal. When $q > q_c$, the signal is over-saturated, and the uniform delay will grow with the analysis period and eventually approach infinity. The over-saturation case is not considered in this study.

When $0 \leq q < \frac{g_{min}}{g_{min}+r} q_s$, the green indication period will be fixed at the minimum green time. The uniform delay is

$$d_1(q) = \frac{r^2}{2(g_{min}+r)}\left(1-\frac{q}{q_s}\right)^{-1} = \frac{r}{2}\left(\frac{r}{g_{min}+r}\right)\left(1-\frac{q}{q_s}\right)^{-1} \quad 0 \le q < \frac{g_{min}}{g_{min}+r}q_s \tag{4.13}$$

The overall trend of the uniform delay for actuated signal control can be expressed as

$$d_1(q) = \begin{cases} \frac{r}{2}\left(\frac{r}{g_{min}+r}\right)\left(1-\frac{q}{q_s}\right)^{-1} & 0 \le q < \frac{g_{min}}{g_{min}+r}q_s \\ \frac{r}{2} & \frac{g_{min}}{g_{min}+r}q_s \le q \le q_c \end{cases} \tag{4.14}$$

Based on the above equation, the relationship between traffic flow rate and average delay per vehicle at an actuated signal is illustrated in Figure 4.5. The bold curve shows the change of the average vehicle delay, which first follows the pre-timed uniform delay function of $g = g_{min}$, and then remains when the green time is being actuated (i.e., $g_{min} \le g \le g_{max}$). When $0 \le q < \frac{g_{min}}{g_{min}+r}q_s$, we have $\frac{r}{2} - \frac{g_{min}}{2}\left(\frac{r}{g_{min}+r}\right) \le d_1(q) < \frac{r}{2}$. Note that as the difference is $\frac{g_{min}}{2}\left(\frac{r}{g_{min}+r}\right) < \frac{g_{min}}{2}$. Given that the minimum green time is usually small (e.g., around 4 seconds), the slight delay difference at low traffic demand is often negligible when comparing with the intersection delay and link travel time. Thus, to simplify the delay function, we generally assume that the uniform delay is a constant when the signal control is under-saturated.

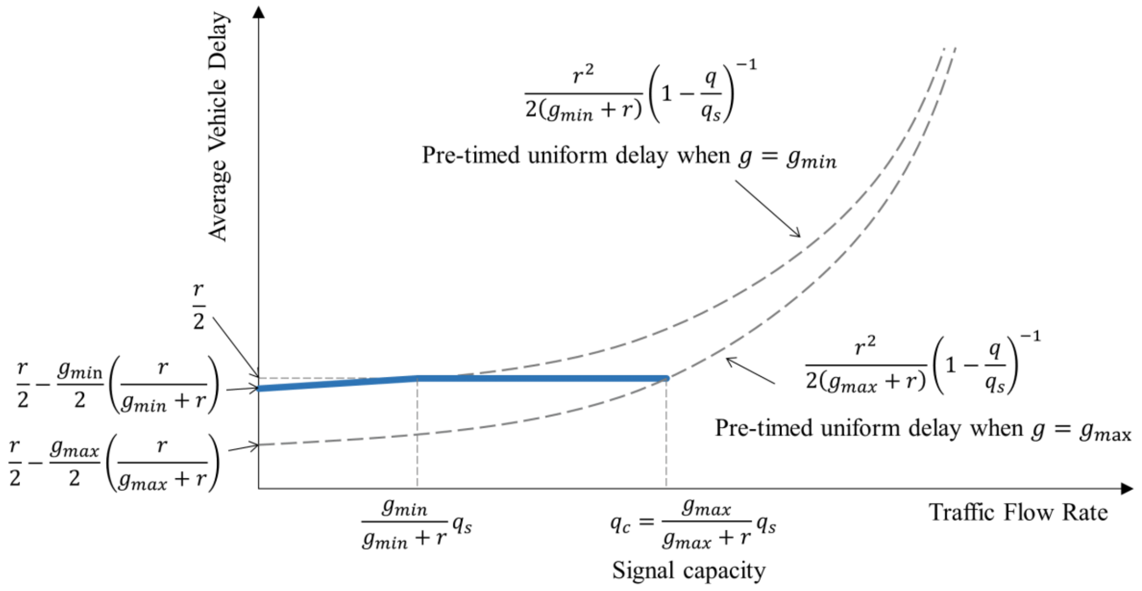


Figure 4.5. Average vehicle delay curve for actuated signal control.

Let d_u denote the constant uniform delay, the final average vehicle delay function for actuated signalized intersection is

$$d(q) = d_u + a\left(1 - b\frac{q}{n \times q_c}\right)^{-1} \tag{4.15}$$

It is important to note that the conclusion that the uniform delay remains constant when the green signal is being actuated relies on the signal setting that the vehicle extension approximately equals the saturation headway. As a result, vehicle leave the traffic signal at the saturation flow rate during the entire green signal period. If the vehicle extension time is set longer, then the average vehicle delay will have a more complicated relationship with the traffic flow rate.

According to the derivation, the constant uniform delay equals to half of the effective red time. For actuated signal control, the effective red time includes green times of conflicting phases and is determined by conflicting traffic flow rates. Here we assume that the traffic flows from different directions are independent and uses a constant term to generally represent the average effective red time of the studied phase. Such an assumption is reasonable when focusing on analyzing individual vehicle's travel time and system impact, as the route choice of an individual vehicle will unlikely cause a big change in the traffic demands from other directions. When estimating the delay function parameters, however, different delay functions need to be trained when the traffic condition changes. For example, an intersection may have different delay curves (i.e., delay functions with different parameters) during peak and off-peak hours. In the real-world applications, it is recommended that the delay function parameters being trained periodically to reflect the actual traffic scenario.

4.2.3 *Parameter Estimation*

Previous sections summarized the development of the average vehicle delay function for actuated signalized intersections in this study. The general delay function form has two empirical parameters (a and b) to be learned from the actual traffic data.

This study estimates the delay function parameters that minimizes the mean squared error (MSE) of the delay estimates. Let $\{d_i\}_{i=1}^{N}$ and $\{q_i\}_{i=1}^{N}$ denote the set of average vehicle delay and traffic flow rate collected at an intersection, the parameter estimation can be expressed in the form of a constrained optimization problem as

$$\min_{a,b} \quad \mathrm{MSE}(a,b) = \frac{1}{N}\sum_{i=1}^{N}\left(d_i - d_u - a\left(1 - b\frac{q_i}{n\times q_c}\right)^{-1}\right)^2 \tag{4.16}$$
$$\text{s.t.} \qquad a \geq 0$$
$$b \geq 0$$

Recall that if the intersection capacity can be accurately estimated, we should have $0 \leq \mathrm{b} \leq 1$. However, it is difficult to estimate the exact intersection capacity due to complex traffic control rules such as shared lane and phase by different movements. Thus, we remove the constraint $b \leq 1$ so that the model can handle the case when we overestimate the intersection capacity.

As the objective function is continuous and smooth (i.e., the second derivatives exist and are continuous), this study applied a Newton-based algorithm to find the optimal parameter estimates (Schnabel, 1985). The general procedure of the algorithm is

1. At iteration 0, choose a pair of initial parameters (a_0, b_0) to start;
2. At iteration k, compute a descent direction (s_k, t_k) for $\mathrm{MSE}(a,b)$ at (a_k, b_k) using Newton's method;
3. Update $(a_{k+1}, b_{k+1}) = (a_k + \gamma s_k, b_k + \gamma t_k)$, where γ is the line-search parameter;
4. Iterate steps 2-3 until the decrease of $\mathrm{MSE}(a,b)$ is smaller than the tolerance.

Newton’s method is guaranteed to converge if the starting point is sufficiently close to the global minimum. Thus, it is important to choose a reasonable starting point for the function parameters. According to the numerical and simulation experiments, this study found that $(SD(d), 0.5)$ is a robust starting point that converges to the global minimum in all test cases, where $SD(d) = \sqrt{\sum(d_i - \bar{d})/(N-1)}$ is the standard deviation of the average vehicle delay observations. Recall that the physical meaning of parameters a and b are the scale of the stochastic delay and the level of randomness in the traffic flow. The recommended starting point $(SD(d), 0.5)$ is essentially a reasonable guess of the parameter values.

After including the intersection delay function into the base form of the link cost function, the complete function for link cost estimation is

$$t(q) = t_0 + d_u + a\left(1 - b\frac{q}{n \times q_c}\right)^{-1} \tag{4.17}$$

Based on historical travel time and traffic flow rate, the delay function parameters can be learned using data collected at each road link. Thus, the delay function parameters will be different at each link with a specific traffic scenario. Such information will be stored in the RSUs and shared with other connected devices in the CVS.

4.3 NUMERICAL TESTS

Numerical experiments have been conducted to evaluate the accuracy of the proposed link cost function. The numerical experiments aim to validate the function in various traffic scenarios. As aforementioned, the actual traffic flow is a mix of two types: deterministic and stochastic vehicle arrivals. The traffic flow randomness also depends on both the flow rate and the upstream traffic control plan. Thus, in the numerical experiment, we considered different combinations of the two types of flows in the arrival traffic to mimic real-word traffic in various conditions.

4.3.1 *Study Scenario*

A typical traffic scenario is considered to test the developed stochastic delay function. The traffic scenario is based on a one-lane single direction road with a traffic signal operated by an actuated signal control plan illustrated in Figure 4.6.

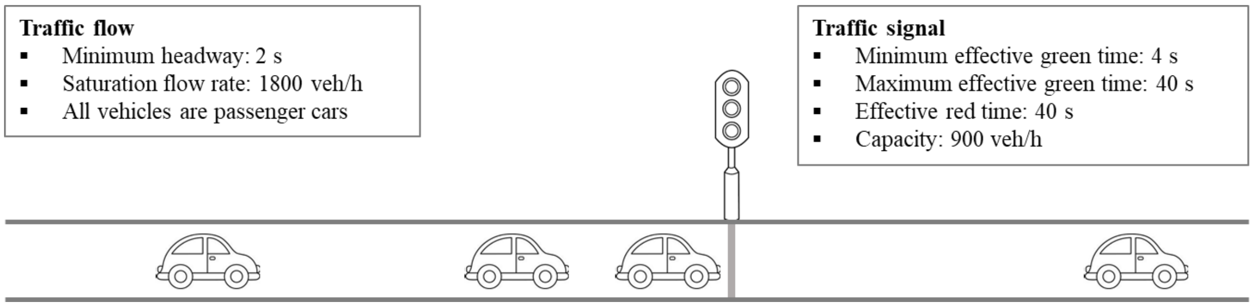

Traffic flow
▪ Minimum headway: 2 s
▪ Saturation flow rate: 1800 veh/h
▪ All vehicles are passenger cars
Traffic signal
▪ Minimum effective green time: 4 s
▪ Maximum effective green time: 40 s
▪ Effective red time: 40 s
▪ Capacity: 900 veh/h

Figure 4.6. Studied traffic scenario.

The basic input of the experiment is summarized as below.

- All vehicles are passenger cars
- There is only one lane to serve the arriving/depart vehicles
- The minimum vehicle headway is 2 seconds, which yields a saturation flow rate of 1800 veh/h.
- The traffic signal configurations are: minimum effective green time $g_{min} = 4$ s; maximum effective green time $g_{max} = 40$ s; and effective red time $r = 40$ s. This gives the intersection capacity of $q_c = 900$ veh/h.
- The analysis period is 3 hours.

In the numerical experiments, the average vehicle delay is computed at different levels of traffic demand comprised of both uniform and stochastic flows.

4.3.2 *Traffic Flow Generation*

A traffic generator is developed to create arrival traffic at a specific flow rate and the traffic flow randomness level. To calculate the average vehicle delay, the traffic generator needs to create the arrival time for each individual vehicle. Instead of sampling traffic counts from certain distributions (e.g., uniform, Poisson, and Binomial distributions) at a macroscopic level, the traffic generator will create random samples of vehicle headway, and then calculate each vehicle’s time of arrival at the signal head.

For uniform vehicle arrivals, the traffic generator will create a new vehicle at fixed rate because the headway is constant. For stochastic vehicle arrivals, Poisson distribution is widely used to describe the vehicle count distribution. Under Poisson vehicle arrival assumption, the vehicle headway follows an exponential distribution (as illustrated in Equation 2.10). However, previous studies have pointed out that Poisson vehicle arrivals may cause unrealistic traffic fluctuations as the exponential distribution allows arbitrary small intervals between vehicles (Newell, 1960).

Instead of directly using the exponential distribution to generate headway samples, we further consider the physical traffic flow constraint in the numerical experiment. In the real-world traffic flow, drivers tend to keep a safe distance (i.e., the saturation headway) between vehicles, and it is very rare to observe vehicle headway smaller than the saturation headway. In an exponential distribution, the probability density is highest near 0, thus may generate a noticeable amount of headway samples smaller than the saturation flow rate. To overcome this limitation, this study uses the Gamma distribution to generate vehicle headway samples.

The probability density function (PDF) of a Gamma distribution can be expressed as

$$P(h') = \frac{\beta^{\alpha}}{\Gamma(\alpha)} h'^{\alpha-1} e^{-\beta h'} \tag{4.18}$$

where α and β are shape and rate parameters, respectively. The expected headway is $\mathrm{E}(h') = \alpha/\beta$. Comparing with the exponential distribution PDF (Equation 2.10), the Gamma distribution has an additional term $h'^{\alpha-1}$. If the shape parameter α is greater than one, the gamma distribution will have a skewed bell shape which ensures extremely small headways being rarely sampled. In the numerical experiment, we use $\alpha = q^{-1}$ and $\beta = 1$ as the Gamma distribution parameters, so that we have the expected headway equal to the inversed flow rate (i.e., $\mathrm{E}(h') = \alpha/\beta = q^{-1}$). Additionally, as the expected headway is higher than the saturation flow rate (e.g., typically 2 s), we can also ensure $\alpha > 1$ which gives a bell-shaped Gamma PDF.

Another good feature of using Gamma distributed headway is that the vehicle arrival still flows a Poisson-like distribution. When vehicle headway follows an exponential distribution $h \sim \exp(\theta)$, the vehicle arrival times can be calculated as the cumulative sums of headways. According to probability theory, the sum of independent and identically distributed (i.i.d.) exponential random variables is a Gamma random variable, which is expressed as

$$X_k = \sum_{i=1}^{k} h_i \sim \mathrm{Gamma}(k, \theta) \tag{4.19}$$

Note that X_is are not independent. The Poisson distributed vehicle count can be written as

$$P_{Poisson}(Y = k | \lambda = qT) = P(X_k \leq T,\ X_{k+1} > T) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{4.20}$$

If using Gamma distributed headway (i.e., $h' \sim \mathrm{Gamma}(\alpha, \beta)$), the vehicle arrival times can be similarly calculated as the cumulative sums of headways. According to Gamma distribution features, the sum of i.i.d. Gamma random variables is also a Gamma random variable, which is

$$X'_k = \sum_{i=1}^{k} h'_i \sim \mathrm{Gamma}(k\alpha, \beta)$$

In this case, the probability mass function of vehicle count can be expressed as

$$P(Y' = k | T) = P(X'_k \leq T,\ X'_{k+1} > T)$$

Given that $\{X_k\}$ and $\{X_k'\}$ are both Gamma distributed sequences with increasing shape parameter and fixed rate parameter, the vehicle count based on Gamma distributed headway will follow a Poisson-like distribution.

The numerical experiment will test different degrees of randomness in the traffic flow to mimic real-world traffic situations. This is represented by the percentage of stochastic vehicle headways in the arrival traffic. For a 50% stochastic traffic flow, for example, half of the vehicle headways are constant (for uniform traffic flow) and the other half are sampled from the exponential distribution. Figure 4.7 shows the distribution of five-minute vehicle count at different degrees of randomness created by the traffic generator. The average flow rate is 600 veh/h and the average five-minute vehicle count is 50. Three degrees of randomness (i.e., 20%, 50%, and 100%) were tested. With low randomness in traffic (e.g., 20%), the vehicle counts densely distribute around the average value. When the degree of randomness increases, however, the distribution spreads, and the vehicle count has a greater variance.

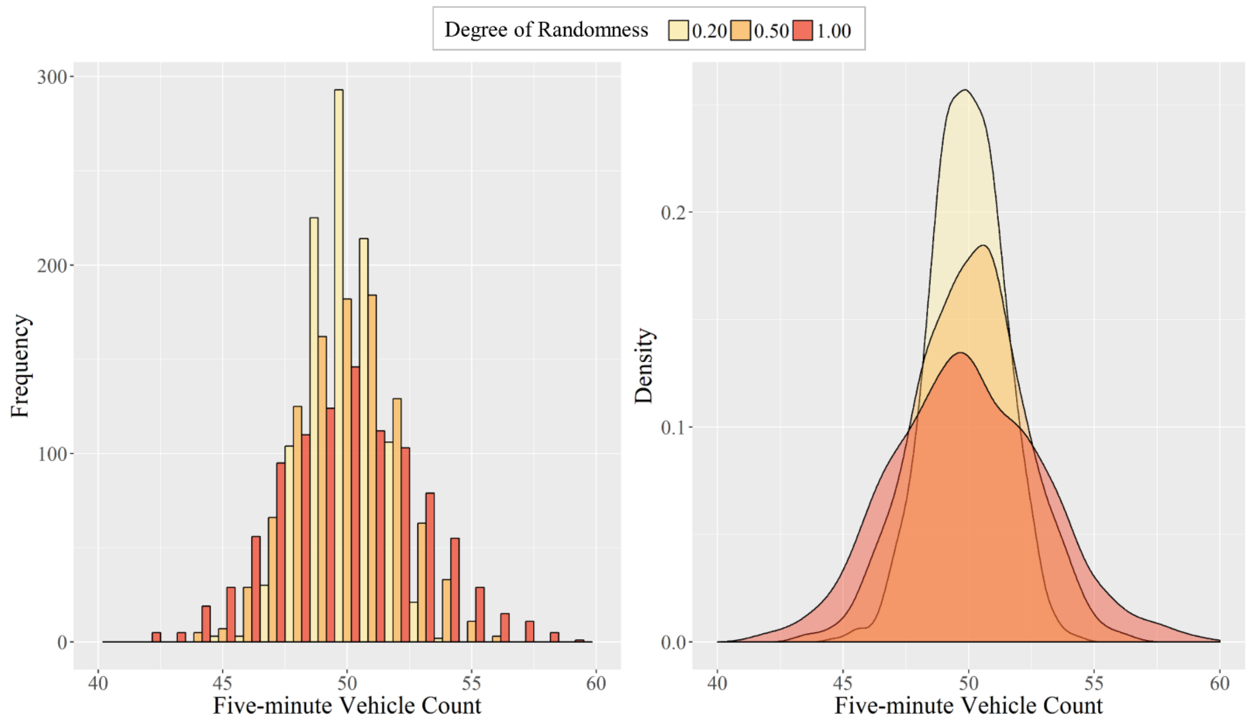


Figure 4.7. Five-minute vehicle count distribution for different degrees of randomness.

4.3.3 *Test Result*

To test the effectiveness of the intersection delay function in different scenarios, the numerical experiments calculate the average delay of various traffic demand and randomness levels. For each traffic flow rate and each degree of randomness, the numerical experiment is repeated ten times with different random seeds. The numerical experiment results are used to estimate the proposed delay function parameters, and a unique delay function is learned for each particular degree of randomness. Recall that theoretically, the uniform delay changes from $\frac{r}{2} - \frac{g_{min}}{2}\left(\frac{r}{g_{min}+r}\right)$ to $\frac{r}{2}$ when $g < g_{min}$, and remains constant (i.e., equal to $\frac{r}{2}$) when $g \geq g_{min}$. To balance the delay function accuracy in both cases, here we use $d_u = \frac{r}{2} - 1$ as the uniform delay term in the delay function. As the effective green time is four seconds in the numerical experiment, the estimation error cause by the constant uniform delay assumption is less than one second in all cases.

Figure 4.8 presents the scatterplot of average vehicle delay against the traffic flow rate for different traffic scenarios when the degree of randomness varies from 10% to 90%. The fitted delay function with parameters learned from the data are also presented in dashed curves. In general, the empirical delay functions approximately describe the delay trend in all scenarios. The fitted delay function can accurately reflect the delay increase when the traffic demand approximates the capacity with sufficient number of observations (e.g., the degree of randomness greater than 50%). In terms of result accuracy, the proposed delay function will slightly overestimate under low traffic demand and underestimate under moderate traffic flow rate. This is due to the approximation of the constant uniform delay assumption. In the numerical experiments, the root mean squared error (RMSE) is around 0.5-1 s/veh, which is smaller than half of the minimum effective green time as expected. The estimation error is negligible compared to the overall intersection delay.

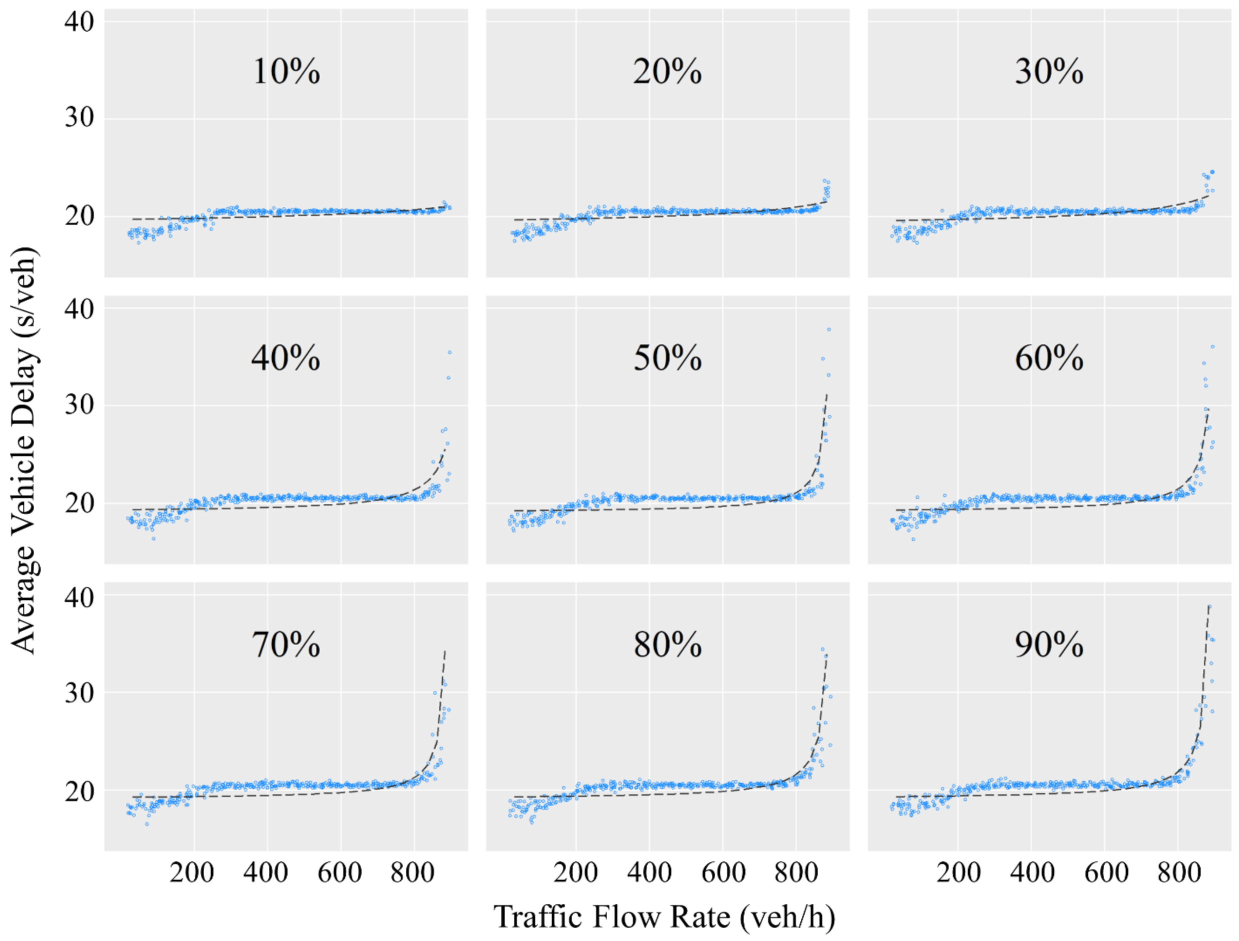


Figure 4.8. Average vehicle delay and fitted curves from numerical experiments.

Figure 4.9 shows the estimated delay function parameters at different levels of traffic flow randomness. Note that for each traffic flow scenario, the experiment is repeated ten times with different random seeds. In the figure, the mean values of the estimated parameter from the ten experiments are represented in points and the standard deviations are represented as error bars. With the increase of traffic flow randomness, both parameters tend to converge to some value. The stochasticity parameter b increases with the traffic flow randomness, and eventually approaches 1 when the vehicle arrivals are completely stochastic. Note that here we know the exact capacity (i.e., $q_c = 900\ \text{veh/h}$), thus b approaches 1 when the degree of traffic flow randomness increases. If the capacity estimation is inaccurate, the stochasticity parameter b will converge to a different value. The scale parameter a first decreases to balance the increase of b, and then stabilized at some positive value. Changes in the function parameter estimates indicate that the empirical delay function is adaptive in various traffic conditions.

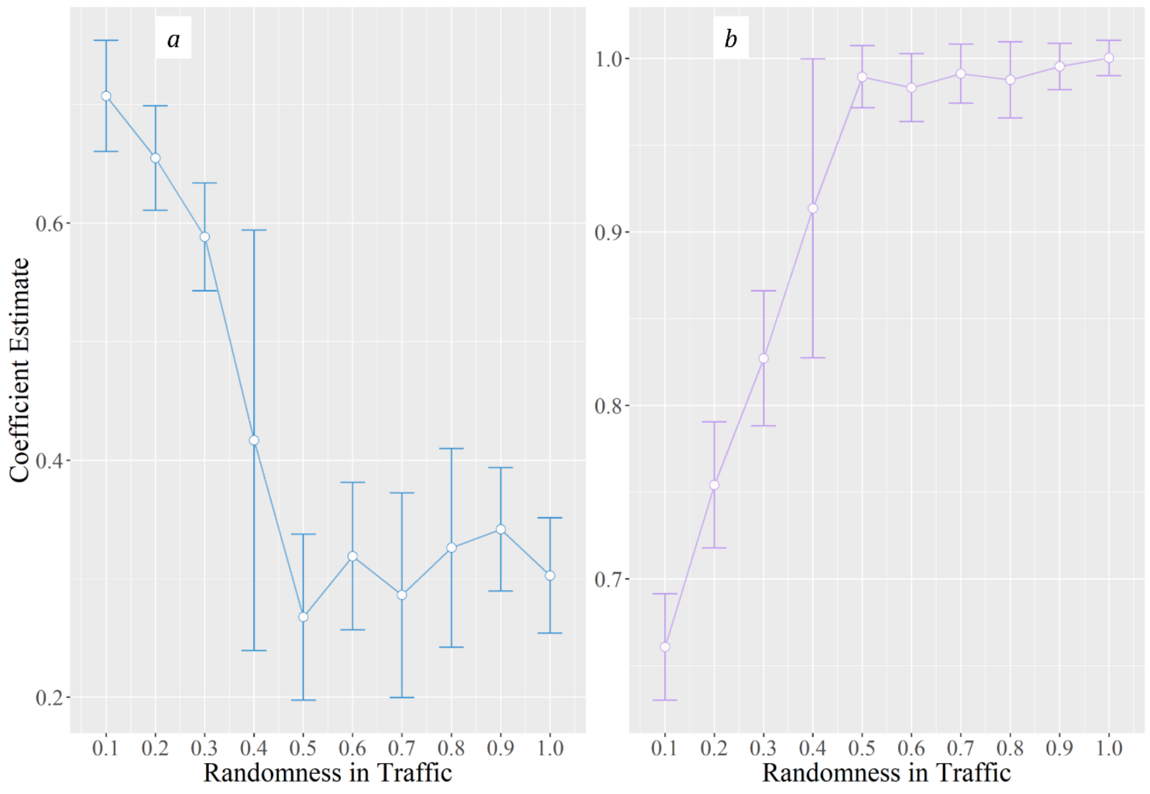


Figure 4.9. Relationship between delay function parameters and traffic flow randomness.

In the numerical experiment results, the average vehicle delay is almost flat for moderate traffic demand. This is because we assume traffic flows in different directions are independent and thus the effective red time does not change with the current phase traffic flow rate. In the real-world situation, however, traffic flows in different directions can be correlated to some extent (i.e., different directions may have similar traffic patterns). This will cause the effective red time to be positively correlated with the traffic demand. Thus, the "flat" part of delay results will be less observable, and the average vehicle delay will increase more smoothly with the traffic demand. It is expected that the empirical delay function can fit the results more accurately when the delay trend is smoother.

Although the numerical experiment is based on a typical one-lane actuated signalized intersection, the experiment conclusions can be applied to more general cases. In the following section, the empirical delay function will be used to fit delay results from the microscopic simulation model in order to set up the link travel time function used in the navigation algorithm.

4.4 SIMULATION TESTS

The empirical delay function is utilized to learn the relationship between link travel time and traffic volume in the VISSIM simulation model introduced in Section 3.3. The objectives of simulation experiments include: 1) validating the empirical delay function in the microscopic simulation environment with more complicated traffic characteristics (e.g., lane switching, acceleration/deceleration, and multi-lane intersections); and 2) developing link cost functions to be used in the CV navigation methodology.

To learn the link cost function, it is necessary to collect link travel time data under different traffic demand levels. Thus, we change the simulation vehicle input from 20% to 200% of the default traffic volume (i.e., vehicle counts collected during off-peak hours) to collect link travel time data at various traffic conditions. According to the simulation results, majority of the roads are congested when the vehicle input is twice of the default value. Therefore, we did not further increase the traffic input beyond 200% in the simulation experiments.

The average travel time vehicles spent on each link and its downstream intersection has been collected during the simulation runs. For each vehicle input level, the simulation is repeated ten times with different random seeds. Each simulation run lasts for two hours. The first hour is the warm-up period and vehicle travel time results are only collected in the second hour. In the following paragraphs, we introduce the detailed step of fitting the empirical delay function based on a particular link (and its downstream intersection). Other links in the network can be analyzed in the similar procedure.

The studied link is the southbound of Bellevue Way NE at the intersection of NE 8th St and Bellevue Way NE. Figure 4.10 shows the screenshot of the intersection in the microscopic simulation model and the traffic count for each movement. The intersection is located in the busy commercial area in downtown Bellevue and four directions have nearly balanced traffic demand.

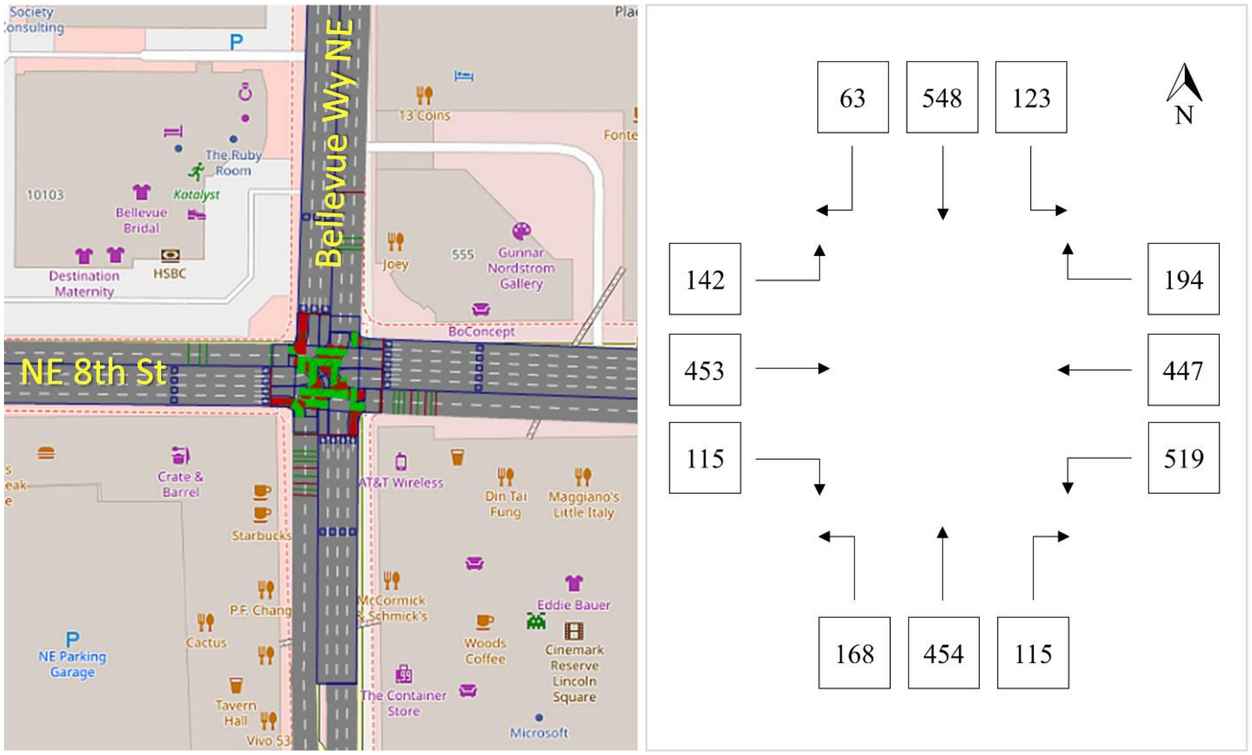


Figure 4.10. Example intersection in the microscopic simulation model.

Figure 4.11 shows the collected average link travel time at different traffic input rates. The scatter points are slightly jittered in the horizontal direction to avoid overlapping pointes. When the traffic is uncongested (lower part of the graph), the link travel time smoothly increases with the traffic volume. For the studied link, the traffic becomes congested when the vehicle input is higher than 160% of the default. As we only focus on analyzing the uncongested link cost function, traffic scenarios with traffic input rate higher than 160% are removed. The remaining simulation results are used to learn empirical parameters of the link cost function as expressed in Equation 4.17.

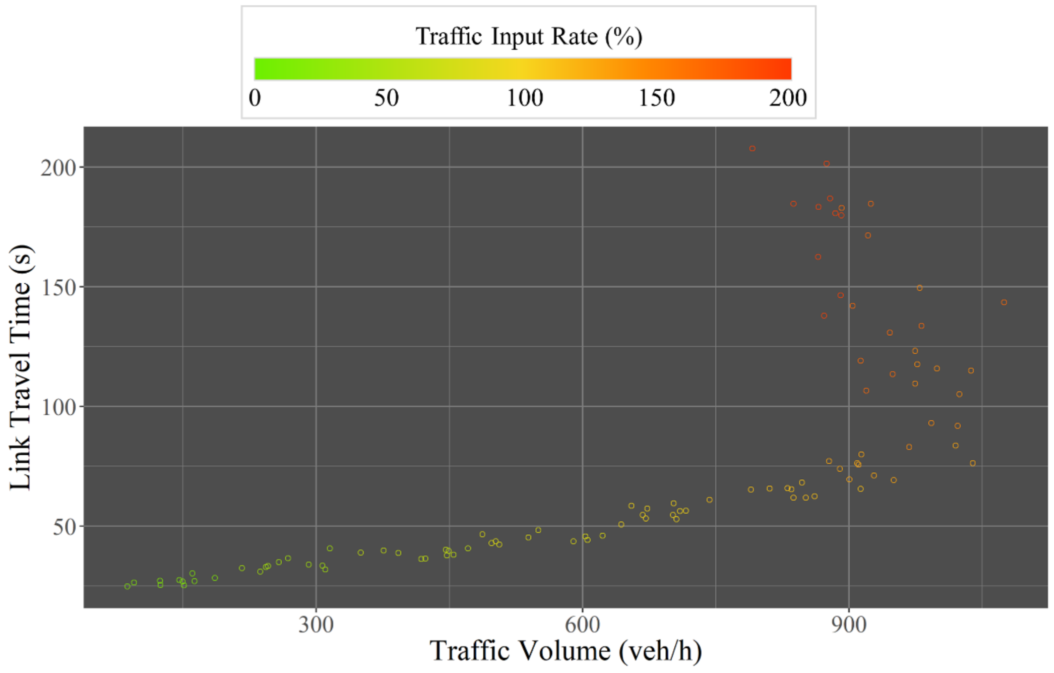


Figure 4.11. Signal control diagram of the studied intersection.

The first step is to estimate the pre-determined (or non-empirical) parameters in the link cost function. The base travel time, t_0, can be directly calculated from the link length and design speed. Given the link length of 661.4 ft and design speed of 50 mph, we have $t_0 = 9.02$ s. Estimation of the constant uniform delay term, d_u, is less straightforward as the signal control is actuated. Note that from the empirical delay function, the minimum possible delay we can obtain is d_u (i.e., the uniform delay assuming no randomness in traffic). Thus, we set d_u equal to a value slightly smaller than the minimum delay observed in the simulation results (e.g., $\min(d_i) - 10$ is used for the studied link). The intersection capacity q_c, again, cannot be accurately estimated due to the actuated signal timing and complicated traffic control methods (e.g., shared lane and phase by different movements). Here we simply assume the saturation headway is 2 seconds and the effective green indication ratio is 50%, which gives the intersection capacity of $q_c = 900\ \mathrm{veh/h}$. According to Figure 4.11, we slightly underestimate the intersection capacity (as the actual capacity is around 1000 veh/h). Recall that the stochasticity parameter b can adjust to handle inaccurate capacity estimate, and the assumption of $q_c = 900\ \mathrm{veh/h}$ is sufficiently accuracy to learn the empirical delay function.

The delay function parameters are estimated by minimizing the MSE of the function result. Figure 4.12 shows the simulation travel time results (in scatter pointes) and the fitted link cost function in dashed curve. In general, the fitted link cost function accurately describes the trend of link travel time results. The RMSE is 7.82 s/veh, and majority of the estimation error comes from the high traffic demand scenario. The simulation test proves that the proposed delay function is effective to capture the relationship between link travel time and traffic volume. Note that here we use the hourly average link travel time to fit the link cost function. If the link travel time is aggregated at shorter time intervals (e.g., 15 minutes), the travel time data will have greater variance, but the trend of the data will not be affected. When fitting the link cost function, it is recommended that the link travel time is aggregated by longer time intervals (e.g. an hour) to reduce the variance, so that the trend of the link travel time can be accurately captured.

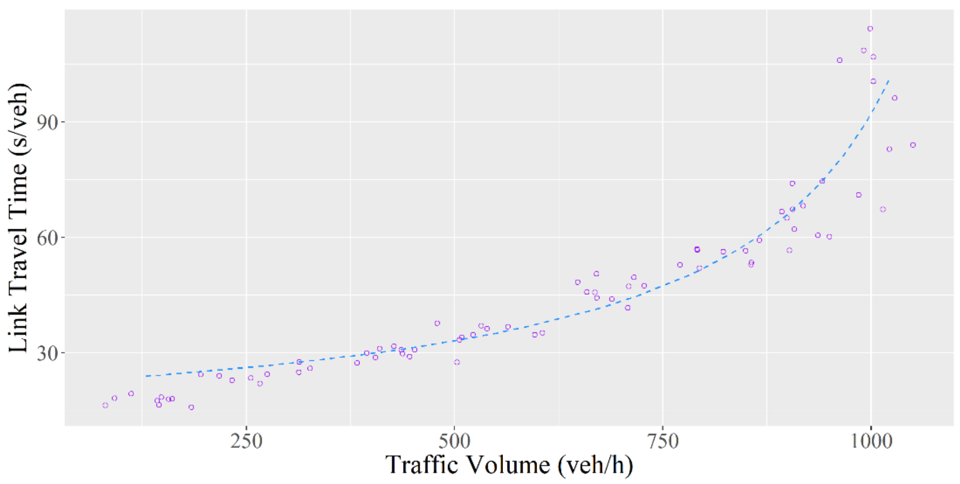


Figure 4.12. Simulated delay results and fitted delay function.

The fitted link cost function will be used in the navigation methodology to assess travel time and system impact of different route choice decisions. Note that the similar procedure is repeated for each link and a unique link cost function is developed based on travel time results collected at the link (and the downstream intersection) during the simulation runs. A complete list of curve-fitting results for all links is presented in Appendix A.

Chapter 5. DYNAMIC TRAFFIC ASSIGNMENT

This chapter describes the objectives of the CV navigation methodology in the DTA context. Specifically, this study considers time-variant traffic flows and travel times, and the CV navigation algorithms target to achieve some special cases of DTA (e.g., DUE or DSO). As the primary goal of this research is to provide navigation guidance to individual connected vehicles, the DTA models will focus on the microscopic vehicular behaviors instead of the macroscopic traffic flow analysis.

5.1 NETWORK REPRESENTATION

This section clearly defines notations used to represent traffic network throughout this paper. Note that some of the notations have also been used consistently in Chapter 2. The roadway network can be represented as a directed graph $G(V, E)$ consisting of edges and vertices, representing road links and intersections, respectively. Let $\{V\}$ denote the group of vertices and $\{E\}$ denote the group of edges. The number of vertices and edges can be expressed by $|V|$ and $|E|$, respectively. A path P is a collection of $|P|$ connected edges expressed as

$$P = \left\{E_P^1, \dots, E_P^{|P|}\right\} \tag{5.1}$$

For an individual vehicle v, let $\mathbb{P}_v$ denote the set of all possible paths from its origin to the destination, written as

$$\mathbb{P}_v = \left\{P_v^1, \dots, P_v^{|\mathbb{P}_v|}\right\} \tag{5.2}$$

Each path P_v^i is a unique path from the origin vertex to the destination vertex. For the specific vehicle, the total number of feasible paths is $|\mathbb{P}_v|$. The travel time (including both the link travel time and delay at the downstream intersection) associated with edge E is expressed as $t_E(q_E|T)$, which takes the form of the link travel time function (Equation 4.17). Note that the edge traffic volume, q_E, is a time-variant variable. We use $q_E|T$ to represent the edge traffic flow rate at time T.

To avoid confusion in notation, in this paper we use lower-case t to represent a time period (e.g., the edge travel time), and upper-case T to represent a time point (e.g., the trip start time). The time-variant traffic flow rate can be summarized from historical traffic data (e.g., average traffic volume aggregated by different time intervals).

5.2 Vehicle Classes

Real-world traffic is not homogeneous, and this study also considers a mixed traffic flow with different vehicle classes. Depending on the accessibility to real-time traffic information, vehicles can be classified into connected and non-connected vehicles (non-CVs). Different navigation algorithms can also be applied to each type of vehicles. Specifically, this study defines the following three vehicle classes with different navigation objectives

1. Non-connected vehicles that navigate using the historical traffic data;
2. Connected vehicles that collaboratively navigate towards the DUE state; and
3. Connected vehicles that collaboratively navigate towards the DSO state.

The following section will discuss the specific navigation method applied to each class. In the algorithm tests, we will also consider mixed traffic flows with different CV penetration rates to analyze the effectiveness of navigation methods.

5.3 VEHICLE NAVIGATION STRATEGIES

Three types of navigation goals are defined for the vehicle classes defined above, respectively. The following sections will clearly define the navigation objectives mathematically to guide the development of practical navigation algorithms.

5.3.1 *Static Navigation*

The static navigation algorithm is used by non-CVs, which targets to find the optimal path based on the historical traffic data. In the static navigation procedure, the edge travel times are time-invariant and can be pre-determined before the start of the trip. The optimal route is the path with the minimum expected cumulative travel time. Let x_v^P denote the route choice decision for vehicle v expressed as

$$x_v^P = \begin{cases} 1 & \text{if vehicle } v \text{ chooses path } P \\ 0 & \text{otherwise} \end{cases}$$

The vehicle navigation can be represented as an optimization problem expressed as

$$(x_v^P = 1 \Longrightarrow \tau_P = \pi_v), \ \ \forall v, P \in \mathbb{P}_v \tag{5.3}$$

where τ_P is the path travel time calculated as the sum of the static edge travel time

$$\tau_P = \textstyle\sum_{E_P^i \in P} t_{E_P^i} \tag{5.4}$$

π_v is the minimum travel time from all feasible paths, written as

$$\pi_v \equiv \min_{P \in \mathbb{P}_v} \tau_P \tag{5.5}$$

Given the time-invariant edge travel time t_E, the optimal route can be found using the classical shortest-path algorithm as described in Section 2.2.2. Note that the static navigation infers that the edge travel cost does not change in the optimal path searching process. But this does not limit to vehicles only using historical traffic data. If the vehicle collects and calculates the optimal path based on the edge travel time collected at the start of the trip, this is still considered as static navigation problem as the edge travel cost does not change in the optimal path searching process.

5.3.2 *Dynamic User Equilibrium Navigation*

The static navigation aims to find the optimal path based on the knowledge before or at the time of the start of the trip. However, the selected path may not continue to be "optimal" due to traffic incidents or non-recurrent congestions that cause changes in traffic during the trip. Additionally, independent route choice decisions may result in the optimal path being over-traveled. Hence, the static navigation algorithm does not guarantee to find the optimal path considering route choice decisions from all other vehicles. As connected vehicles can collect traffic data and communicate with each other in real-time, there is a potential to find a more robust optimal path with better travel cost outcomes.

In this section, we introduce the CV navigation methodology that approximate the DUE traffic assignment in DTA models. The path-based DUE definition is presented in Section 2.2.3. Here we rewrite the DUE conditions from the individual vehicle perspective as follows

For each vehicle, starting from any instant of time, traveling between a specific origin-destination pair, the actual experienced route travel cost equals to the minimum possible travel cost on all feasible routes.

Casting into mathematical notations, if a vehicle v departs at time instant T_v, and all feasible routes for its O-D pair is $\mathbb{P}_v$, then it will select the path with the minimal travel cost. Let $\tau_P(T_v)$ denote the path travel time when departing at T_v, the minimum travel cost of vehicle v departing at T_v from its origin to the destination can be expressed as

$$\pi_v(T_v) \equiv \min_{P \in \mathbb{P}_v} \tau_P(T_v) \tag{5.6}$$

Then the vehicle level route choice decisions in the DUE state can be written as

$$\left(x_v^P = 1 \Longrightarrow \tau_P(T_v) = \pi_v(T_v)\right), \ \ \forall v, P \in \mathbb{P}_v \tag{5.7}$$

where x_v^P is the route choice decision variable of vehicle v defined the same as that in the last section.

The key task in DUE navigation methodology is to quantify the dynamic travel time of different paths. The path travel time can be expressed as the sum of its edge travel times at different time instants

$$\tau_P(T_v) = \sum_{E_P^i \in P} t_{E_P^i}\left(q_{E_P^i} \middle| T_v^{E_P^i}\right) \tag{5.8}$$

where $T_v^{E_P^i}$ denotes the time instant vehicle v arrives at the start node of edge E_P^i if choosing path P. In the dynamic traffic flow context, the times of arrival at each edge can be calculated as cumulative sums of edge travel times, which is

$$T_v^{E_P^i} = \begin{cases} T_v & \text{if } i = 1 \\ T_v^{E_P^{i-1}} + t_{E_P^{i-1}}\left(q_{E_P^{i-1}} \middle| T_v^{E_P^{i-1}}\right) & \text{if } i > 1 \end{cases} \tag{5.9}$$

where E_P^{i-1} is the proceeding edge of E_P^i in path P, if E_P^i is not the first edge of path P (in this case we have $i = 1$ and the vehicle arrives at the edge at T_v). Equation 5.9 explicitly represent the edge arrival time as the sum of the arrival time and travel time of the proceeding edge. Note that here we assume that the travel time vehicle v spent on an edge is determined at the time vehicle v arrives at the edge. This is an intuitive assumption as vehicles arriving later than v should not influence the travel time of v in the first-in-first-out (FIFO) car-following environment.

Based on the edge arrival times, the traffic flow propagation is satisfied in the sense that the time-variant edge traffic flow rate is calculated as the sum of individual vehicles traveling on the edge during a particular time period. Casting into mathematical notations yields

$$q_E|T = \frac{1}{\Delta T}\sum_v \sum_{P \ni E} x_v^P \cdot \mathbb{I}(T - \Delta T < T_v^E \leq T) \tag{5.10}$$

where

$$\mathbb{I}(T - \Delta T < T_v^E \leq T) = \begin{cases} 1 & \text{if } T - \Delta T < T_v^E \leq T \\ 0 & \text{otherwise} \end{cases}$$

Basically, the edge traffic flow at time instant T is defined as the average traffic flow rate during a short time period of $(T - \Delta T, T]$. The length of the analysis time period, ΔT, should be reasonably determined so that vehicles arriving during the period actually affect the edge travel time at T. In the local road network with intersections, ΔT should be in a similar scale as the signal cycle length to reflect the intersection traffic control operations.

Equations 5.6 - 5.10 express the vehicle-based DUE conditions as a mathematical problem. However, the mathematical problem is only of theoretical importance, and it is hard to solve the problem analytically due to discrete traffic volume and huge number of variables. This study focuses on simulation-based traffic analysis and proposes a coordinated vehicle navigation approach to solve the problem from the individual vehicle route choice perspective. Specifically, this study makes the following assumption:

The DUE conditions can be reached through all vehicles using time-dependent navigation that finds the user optimal path based on other vehicles' routing decisions.

It is important to note that the DUE state is an idealistic DTA condition since it requires the actual experienced travel cost of each vehicle to be minimal. From the individual vehicle perspective, however, it is impossible to obtain accurate evaluation of the path travel time at the start of the trip. Thus, this study proposes a DUE navigation methodology for connected vehicles from the following major aspects

- Future traffic flows and travel times are predicted based on real-time traffic data and route choice decisions of connected vehicles;

- Connected vehicles navigate following the optimal path based on the predicted future traffic flow and travel time; and
- Frequently update the travel time predictions using the most recent traffic data to ensure accuracy.

Note that as the optimal path is identified from the predicted travel time, the resultant traffic assignment is not strictly in a DUE state. Rather, the network traffic flow will approximate the DUE state in the sense that the actual travel time of each vehicle is approximately the minimum travel time on any feasible routes given the travel time predictions are sufficiently accurate.

The DUE vehicle navigation strategy can be represented by Equation 5.7, which aims to find the path with the minimum travel time. The only difference is that vehicles will use predicted travel times to identify the optimal path. Based on the edge arrival time equation (Equation 5.9), the downstream edge travel time depends on the predicted vehicle arrival time, which is expressed as the sum of the predicted arrival time and travel time of the upstream edge. This forms a dynamic programming framework in the sense that evaluating the edge travel time requires memorizing results of all preceding edges. This chapter focuses on discussing the vehicle navigation objectives to approximate the desired DTA conditions. More details about vehicle navigation algorithms will be presented in Chapter 6.

It is also necessary to mention that in the DUE state the optimal path is conditioned on route choice decisions of all other vehicles. Thus, collaboration is required among vehicles to avoid the optimal path being over-traveled. This study also proposes a coordinated vehicle navigation algorithm that utilizes the communication capability in the CVS. In the coordinated navigation system, each connected vehicle knows the route choice decision of all other CVs. The DUE navigation algorithm will find the optimal path conditioned on all other vehicles, and in the equilibrium state, no vehicle has the incentive to switch to a different path.

5.3.3 *Dynamic System Optimum Navigation*

The previous section introduces individual vehicles' navigation objectives to approximate the DUE state. From the system perspective, however, the DUE state is usually not optimal in terms of minimizing the total travel time for all vehicles. DSO represents the optimal traffic state from the system perspective, where the total travel time experienced by all vehicles is minimized. Although it is commonly accepted that DSO does not reflect real-world traffic conditions, it can still provide insights on the potential optimal network traffic condition. Additionally, DSO can be practically applied in some special cases. For example, a centralized authority controls a fleet of CVs and targets to minimize the total travel cost rather than the individual travel cost of each vehicle. In this section, we will discuss the vehicle navigation objectives towards the DSO state.

In macroscopic DTA research, the DSO state is usually formulated as an optimization problem that minimizes the sum of vehicle travel times in an integral over time (for continuous-time DTA models) or in a summation of all time intervals (for discrete-time DTA models). Such an optimization problem is not directly defined as individual vehicle navigation objectives. Studies found that the DSO conditions can be converted in to a path-based optimization in a similar form as the DUE problem (Qian et al., 2012; Shen et al., 2007). Define the path marginal cost as the change of total system travel time corresponding to a unit change of traffic flow on a particular path. From the individual vehicle perspective, the DSO conditions can be expressed as

For each vehicle, starting from any instant of time, traveling between a specific origin-destination pair, the actual induced system cost (path marginal cost) equals to the minimum possible system cost on all feasible routes.

On the individual vehicle level, the path marginal cost can be evaluated as the sum of system costs on the consisting edges. Additionally, the system cost associated with each edge can be evaluated given the link travel cost expressed as a function of traffic flow rate (Du et al., 2015b; Xiao and Lo, 2014). When a vehicle chooses edge E, the resultant system cost consists of two parts

- The private part: the edge travel time for the vehicle itself
- The external part: extra travel time imposed to other vehicles using the edge

The individual vehicle travel time can be estimated using the edge travel time function $t_E(q_E)$. For the existing edge traffic, the extra travel time caused by the route choice of an individual vehicle can be evaluated as the product of the existing traffic flow and the marginal increase of the edge travel time, which is $q_E \times \frac{d\, t_E(q_E)}{d\, q_E}$. Let $t_{s,E}$ denote the system cost associated with edge E, The edge system cost can be expressed as a function of the edge traffic flow rate

$$t_{s,E}(q_E) = t_E(q_E) + q_E \frac{d\, t_E(q_E)}{d\, q_E} \tag{5.11}$$

Expand the edge travel time function $t_E(q_E)$ (Equation 4.17) and we get

$$t_{s,E}(q_E) = t_0 + d_u + a\left(1 - b\frac{q_E}{n \times q_c}\right)^{-1} + \frac{ab}{n \times q_c} q_E \left(1 - b\frac{q_E}{n \times q_c}\right)^{-2} \tag{5.12}$$

Equation 5.12 gives the relationship between the system cost and the traffic flow rate for a particular edge. Given that the edge system cost can be evaluated, a potential way to achieve the DSO state is to allow individual CV to compute its own system optimum route using a DSO navigation algorithm. Compared with macroscopic DSO solutions, which usually requires a centralized authority to calculate the DSO traffic assignment and instruct drivers with proper navigation guidance, the individual vehicle navigation solution can distribute the computation load (for optimal path search) among individual vehicles without relying on a centralized authority.

However, this requires all vehicles having access to the real-time traffic information and collaborating on the optimal route choice decisions.

From the individual vehicle perspective, the DSO conditions can be mathematical expressed as

$$\left(x_v^P = 1 \Longrightarrow \tau_{s,P}(T_v) = \pi_{s,v}(T_v)\right), \quad \forall v, P \in \mathbb{P}_v \tag{5.13}$$

$$\pi_{s,v}(T_v) \equiv \min_{P \in \mathbb{P}_v} \tau_{s,P}(T_v)$$

$$\tau_{s,P}(T_v) = \textstyle\sum_{E_P^i \in P} t_{s,E_P^i}\left(q_{E_P^i}\middle|T_v^{E_P^i}\right)$$

$$T_v^{E_P^i} = \begin{cases} T_v & \text{if } i = 1 \\ T_v^{E_P^{i-1}} + t_{E_P^{i-1}}\left(q_{E_P^{i-1}}\middle|T_v^{E_P^{i-1}}\right) & \text{if } i > 1 \end{cases}$$

$$q_E|T = \frac{1}{\Delta T}\textstyle\sum_v \sum_{P \ni E} x_v^P \cdot \mathbb{I}(T - \Delta T < T_v^E \leq T)$$

$$x_v^P = \{0, 1\}$$

where the individual vehicle route choice variable (x_v^P), edge arrival time ($T_v^{E_P^i}$), and time-variant edge flow rate ($q_E|T$) are defined the same as that in the last section. $\tau_{s,P}(T_v)$ and $\pi_{s,v}(T_v)$ are path system cost and minimum system cost from all feasible paths, respectively, if vehicle v starts at T_v.

Similar to the DUE state, this study will solve the problem from the individual vehicle route choice perspective based on the following assumption:

The DSO conditions can be reached through all vehicles using time-dependent navigation that finds the system optimal path based on other vehicles' routing decisions.

Compared with the DUE problem, the optimal path in DSO context is the path with the minimum system cost. Thus, the edge travel time (t_E) in the DUE problem equations is here replaced by the edge system cost ($t_{s,E}$). Recall that $t_E(q_E)$ is the edge travel time function, which is a convex function when the traffic is uncongested. It is important to analyze the edge system cost function to see if the DSO problem can be solved in a similar way as that for the DUE problem.

Figure 5.1 shows the change of edge system cost of a typical edge against the average traffic flow rate and the degree of randomness in traffic. Note that here we consider stochastic vehicle arrivals, and the actual traffic flow is a mix of uniform and stochastic flows. According to the figure, the edge system cost is mainly determined by the traffic flow rate. With the increase of traffic flow randomness, the expected system cost also increases. Note that the effect of traffic flow randomness is relatively small compared to the effect of flow rate. When the traffic flow rate is small (e.g., less than 600 veh/h), the change of system cost caused by the degree of randomness is negligible. At all levels of traffic flow randomness, the edge system cost is a convex function of the traffic flow rate. This suggests that the vehicle navigation method used to approximate DUE conditions can be similarly applied to the DSO problem.

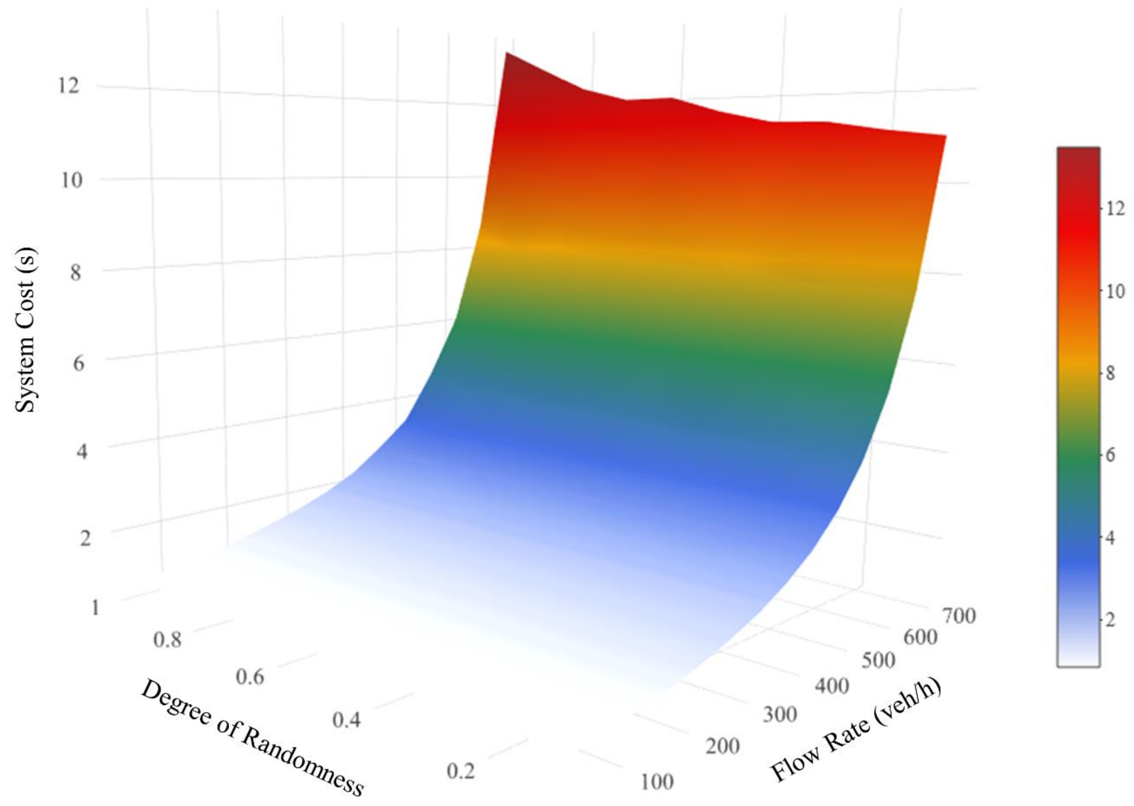


Figure 5.1. Change of system cost against traffic flow rate and degree of randomness.

From the individual vehicle perspective, the DSO navigation algorithm aims to find the path with the minimum system cost conditioned on the route choice decisions of all other vehicles. But the traffic flow propagation also requires evaluating the edge travel times to predict future traffic flows and travel times. Thus, the DSO navigation methodology will evaluate both the edge travel time and the edge system cost, in order to accurately predict the future traffic states and find the system optimal path.

It is important to note that the DSO navigation does not necessarily indicate that the network traffic flow is dynamically optimized at any instant of time. The term DSO is used here because the DSO navigation targets to achieve the DSO state as defined in DTA models. Given that CVs rely on predicted user and system costs to navigate and real-time traffic data are collected and updated at discrete time intervals, the DSO navigation will allocate CVs so that the system approximates the DSO state in the dynamic traffic environment.

Both DUE and DSO navigation algorithms rely on the link travel time function to evaluate the travel cost and system cost. However, this does not necessarily require accurate prediction of the edge travel time and system cost. It is also shown in Chapter 4 that real-world link travel time cannot be exactly expressed as a function due to the traffic flow randomness. In both DUE and DSO navigation algorithms, connected vehicles will communicate and find the optimal routes conditioned on the route choice decisions of all other vehicles. Thus, the link travel time function can actually be considered as a way to penalize vehicles choosing the same path. The general relationship between the link travel time and traffic flow rate is sufficient to help achieve a desired dynamic traffic assignment through the coordinate vehicle navigation methodology.

Chapter 6. VEHICLE NAVIGATION ALGORITHMS

The last chapter analyzed the navigation objectives from the individual vehicle perspective towards a desired DTA state (i.e., DUE or DSO). Following that, this chapter will introduce the specific vehicle navigation algorithms to achieve the objectives.

6.1 Static Navigation Algorithm

Non-CVs aim to find the shortest path based on static edge travel times. The navigation problem can be solved by classical shortest-path algorithms. This study specifically considers two types of shortest-path algorithms: Bellman-Ford algorithm and Dijkstra algorithm. Details of these two methods can be found in Section 2.2.2.

Based on the static navigation algorithms, non-CVs with the same O-D pair will find the same optimal route. If we have a significant amount of non-CVs starting at a short period, this may lead to non-recurrent congestions in the identified optimal route. In such scenarios, vehicle communication is critical to distribute the traffic more efficiently throughout the network.

6.2 Dynamic Navigation Algorithms

Two types of dynamic navigation methods (i.e., DUE and DSO navigation) have been proposed for CVs. Chapter 5 presented the vehicle-based mathematical representations of the DUE and DSO conditions. Under DUE conditions, the navigation problem for vehicle v (assuming it is a CV) starting at T_v can be expressed as

$$\min_{P \in \mathbb{P}_v} \tau_P(T_v) = \sum_{E_P^i \in P} t_{E_P^i}\left(q_{E_P^i} \middle| T_v^{E_P^i}\right) \tag{6.1}$$

where

$$T_v^{E_P^i} = \begin{cases} T_v & \text{if } i = 1 \\ T_v^{E_P^{i-1}} + t_{E_P^{i-1}}\left(q_{E_P^{i-1}}\middle|T_v^{E_P^{i-1}}\right) & \text{if } i > 1 \end{cases}$$

Under DSO conditions, the navigation problem for vehicle v can be expressed as

$$\min_{P\in\mathbb{P}_v} \tau_{s,P}(T_v) = \sum_{E_P^i\in P} t_{s,E_P^i}\left(q_{E_P^i}\middle|T_v^{E_P^i}\right) \tag{6.2}$$

where

$$T_v^{E_P^i} = \begin{cases} T_v & \text{if } i = 1 \\ T_v^{E_P^{i-1}} + t_{E_P^{i-1}}\left(q_{E_P^{i-1}}\middle|T_v^{E_P^{i-1}}\right) & \text{if } i > 1 \end{cases}$$

One of the key features in DUE and DSO navigation problem is that the edge travel time (or edge system cost) varies over time because of the time-variant traffic flow rate $(q_E|T)$. The evaluation of the time-dependent edge flow rate will be introduced in the following section. To solve the dynamic navigation problems, this study also developed a number of time-dependent shortest-path algorithms which will be illustrated in Section 6.2.2.

6.2.1 *Travel Time Prediction*

The dynamic navigation algorithms rely on predicting the future edge travel time (or edge system cost), which requires the estimation of time-dependent traffic flow rate $(q_E|T)$. In the CV-based navigation system design, the traffic flow prediction is conducted by RSUs, which can collect traffic data and predict future traffic states. Specifically, two pieces of traffic information, namely the historical and real-time traffic data, are used in the traffic flow prediction task. A Bayesian method is developed to utilize both historical traffic pattern and real-time traffic measurements. Here we define a “time period” as a length of time that periodically occurs with the similar traffic pattern (e.g., morning peak hours on weekdays). For a particular edge at a specific time period, the historical and real-time traffic data used in the travel time prediction task are defined as

- Historical edge traffic flow rates in the same time period;
- Real-time edge traffic flow measurements in the current or most recent time period.

Based on the two pieces of traffic information, the future traffic volume can be estimated considering both historical traffic patterns and real-time traffic information. The basic idea behind Bayesian theorem is that a rational estimate procedure is to update the prior belief with new information (Hoff, 2009). In the traffic volume prediction task, the historical traffic volume serves as the "prior" belief, the most recent real-time traffic volume is considered as the "new information". The underlying assumption of using the Bayesian method is that many traffic events are periodically recurrent and traffic flows are also temporally correlated. The traffic flow pattern in a certain time period is likely to be similar as that happened before in the same time period. Additionally, real-time traffic flow in the current of most recent time period might also suggest nonrecurrent traffic incidents that should be considered in the traffic flow prediction.

It is important to note that the equation for Bayesian method depends on the specific distribution of the data. Here we again consider a mixed traffic flow with uniform and stochastic vehicle arrivals, which is expressed as

$$q|T = q_u|T + q_r|T \tag{6.3}$$

where $q_u|T$ is the uniform traffic flow. The uniform flow rate is a constant equal to the expected flow rate

$$q_u|T \equiv E(q_u|T) \tag{6.4}$$

$q_r|T$ represent the random/stochastic traffic flow. Here we assume the stochastic vehicle arrivals can be described using a Poisson distribution expressed as

$$q_r|T \sim \mathrm{Poisson}(\lambda) \tag{6.5}$$

where $\lambda = E(q_r|T)$ is the expected stochastic traffic flow rate.

Assume we have observed n_1 historical traffic flow rates $(q_{11}, \dots, q_{1n_1})$ in the time period T and n_2 real-time traffic flow rates $(q_{21}, \dots, q_{2n_2})$ from the current or most recent time period. Each traffic flow observation can be broken down into uniform and stochastic flow components, which is

$$\begin{aligned} q_{1i} &= q_{u,1i} + q_{r,1i} \qquad 1 \le i \le n_1 \\ q_{2j} &= q_{u,2j} + q_{r,2j} \qquad 1 \le j \le n_2 \end{aligned} \tag{6.6}$$

On the uniform flow part, the observed n_1 historical traffic volumes and n_2 real-time traffic volumes all equal to the time conditioned constant traffic volume, which is expressed as

$$q_{1i} = q_{2j} \equiv E(q_u|T) \qquad 1 \le i \le n_1, 1 \le j \le n_2 \tag{6.7}$$

For the stochastic flow component, the joint probability of observing the historical traffic flow data is

$$p(q_{r,11}, \dots, q_{r,1n_1}|\,\lambda) = \prod_{i=1}^{n_1} e^{-\lambda} \frac{\lambda^{q_{r,1i}}}{q_{r,1i}!} \propto \lambda^{\Sigma q_{r,1i}} e^{-n_1\lambda} \tag{6.8}$$

Based on Bayesian theorem, the conditional probability of the Poisson distribution parameter is

$$p(\lambda|\, q_{r,11}, \dots, q_{r,1n_1}) = \frac{p(q_{r,11},\dots,q_{r,1n_1}\,|\,\lambda) \times p(\lambda)}{p(q_{r,11},\dots,q_{r,1n_1})} \propto p(\lambda) \times \lambda^{\Sigma q_{r,1i}} e^{-n_1\lambda} \tag{6.9}$$

Thus, the expected traffic flow rate, $\lambda = E(q_r|T)$, follows a distribution with the probability distribution function including terms like $\lambda^{c_1} e^{-c_2\lambda}$. The simplest probability distribution that includes such terms is the family of Gamma distributions, and the corresponding probability distribution function is

$$p(\lambda) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \tag{6.10}$$

where α, β are distribution parameters and $\Gamma(\cdot)$ is the Gamma function. The mean and variance of Gamma distribution is $E(\lambda) = \alpha/\beta$ and $Var(\lambda) = \alpha/\beta^2$, respectively. If only based on the historical traffic data, when we have observed n_1 traffic flow rates, $q_{r,11}, \dots, q_{r,1n_1}$, λ follows a Gamma distribution with $\alpha_1 = \Sigma q_{r,1i}$ and $\beta_1 = n_1$. The estimated average occupancy is

$$\lambda = E(q_r|T) = \frac{\Sigma q_{r,1i}}{n_1} \tag{6.11}$$

Use this as our prior belief of λ. In addition to the historical traffic data, if we further observe n_2 real-time traffic flow rates $q_{r,21}, \dots, q_{r,2n_2}$, then the posterior distribution of λ is

$$\begin{aligned} p(\lambda|\, q_{r,21}, \dots, q_{r,2n_2}) &\propto\ p(\lambda) \times \lambda^{\Sigma q_{r,2j}} e^{-n_2\lambda} \propto \lambda^{\alpha_1 - 1} e^{-\beta_1\lambda} \times \lambda^{\Sigma q_{r,2j}} e^{-n_2\lambda} \\ &\propto \lambda^{(\alpha_1 + \Sigma q_{r,2j})-1} e^{-(\beta_1+n_2)\lambda} \end{aligned} \tag{6.12}$$

This follows a Gamma$(\alpha_1 + \Sigma q_{r,2j}, \beta_1 + n_2)$ distribution, and the posterior parameter estimate is

$$\hat{\lambda} = E(q_r|T) = \frac{\alpha_1 + \Sigma q_{r,2j}}{\beta_1 + n_2} = \frac{\Sigma q_{r,1i} + \Sigma q_{r,2j}}{n_1 + n_2} \tag{6.13}$$

Based on the Bayesian theorem, the posterior estimation of the stochastic flow rate takes a form of weighted average of historical and real-time data. Combine Equations 6.3, 6.6, 6.7, and 6.13, the time-variant traffic flow rate can be estimated as

$$\begin{aligned} E(q|T) = E(q_u|T) + E(q_r|T) &= \frac{\Sigma q_{u,1i} + \Sigma q_{u,2j}}{n_1 + n_2} + \frac{\Sigma q_{r,1i} + \Sigma q_{r,2j}}{n_1 + n_2} \\ &= \frac{\Sigma q_{1i} + \Sigma q_{2j}}{n_1 + n_2} \end{aligned} \tag{6.14}$$

Equation 6.14 expresses the Bayesian based traffic flow prediction as a weighted average of historical and real-time flow measurements. Here n_1 and n_2 are the number of traffic flow rate observations from historical data and real-time data, but it can be generally interpreted as our "beliefs" in historical and real-time traffic. Thus, the number of historical and real-time measurements included in the prediction can be adjusted according to our belief of the data. For example, for a road with strong traffic patterns in the past, we might include more historical data and less real-time data to generate robust traffic flow estimations. For a road with more nonrecurrent traffic incidents, however, we might include more real-time traffic flow rates to ensure that real-time traffic changes can be sufficiently reflected in the future traffic flow prediction. Let $\gamma = n_2/n_1$ denote the Bayesian parameter, the traffic flow prediction can be further expressed as

$$E(q|T) = \frac{\bar{q}_{1i}+\gamma\bar{q}_{2j}}{1+\gamma} \tag{6.15}$$

Through adjusting the Bayesian parameter, we can change the belief of the real-time data in the traffic flow prediction task.

6.2.2 *Time-Dependent Shortest-Path Algorithms*

As the edge travel time (or edge system cost) is expressed as a time-dependent function, the individual vehicle navigation problems are essentially a time-dependent shortest-path problem. As the edge cost varies over time, the cost of each edge is evaluated at the time when the vehicle is expected to arrive at the edge. In this section, we extend the two classical shortest-path algorithms, namely the Bellman-Ford algorithm and the Dijkstra algorithm to solve the vehicle navigation problems under DUE and DSO conditions. The two algorithms were selected because of two major reasons: 1) both algorithms have been widely studied and practically applied; 2) the two algorithms represent two typical communication plans in the CVS. Comparison of the two algorithms and discussion about the communication system design will be presented in Section 6.2.3

For a roadway network represented as a directed graph $G(V, E)$, let V_i denote a specific vertex and E_{ij} denote the directed edge from V_i to V_j. T_i is the time of arrival at V_i and $t_{E_{ij}}|T$ is the time-dependent travel time associated with the edge E_{ij}. The Dijkstra based time-dependent shortest-path algorithm to find the fastest route (i.e., the route with the minimum total travel time) from the origin vertex V_O to the destination vertex V_D is illustrated as the following steps

1. Initialize the network, set the arrival time to the origin vertex as zero ($T_{V_O} = 0$) and the arrival times to all other vertices as infinity ($T_{V_i} = +\infty, \forall i \neq O$).
2. Initialize an empty parent hash map P to store the parent vertex of each vertex. For example, $P_{V_j} = i$ means the V_i is the parent vertex of V_j (i.e., V_i is the immediate upstream vertex of V_j in the selected path).
3. Initialize the "temporarily visited" group of vertices including the origin vertex, and the "unvisited" group of vertices including all other vertices. Create an empty group of "permanently visited" vertices.
4. Find the vertex V_i with the minimum arrival time in the "temporarily visited" group. For all its unvisited neighbor vertices V_j, calculate the conditional edge travel time $t_{E_{ij}}|T_{V_i}$.
5. Loop through all V_j, if we have $T_{V_i} + t_{E_{ij}}|T_{V_i} < T_{V_j}$, then update vertex V_j with the new arrival time $T_{V_j} = T_{V_i} + t_{E_{ij}}|T_{V_i}$ and update the parent hash map with $P_{V_j} = V_i$. Move V_j into the "temporarily labeled" group.
6. Move V_i into "permanently labeled" group.
7. Repeat steps 4-6 until the destination vertex V_D is in the "permanently visited" group. The shortest path can then be found through tracing the parent vertex from the destination all the way back to the origin.

Compared with the traditional shortest-path algorithm, the revised dynamic shortest-path algorithm has an additional step of updating the edge travel time conditioned on the cumulative travel time calculated from previous steps. The conditional edge travel time can be estimated using the Bayesian function derived in the previous section given that CVs can access both historical and real-time traffic data.

An alternative navigation solution is the Bellman-Ford based time-dependent shortest-path algorithm. When the optimal path is found, the system must satisfy the following set of equations

$$T_{V_i} = \min_{j \neq i} \left(T_{V_j} + t_{E_{ji}} | T_{V_j} \right), \quad i \neq O \tag{6.16}$$

$$T_{V_O} = 0$$

Basically the time of arrival to any vertex equals to the earliest time when the vertex can be reached from its neighboring vertices. Such a condition can be reached using a successive approximation approach illustrated as

1. Start with an initial sequence of arrival times $\left\{T_{V_i}^{(0)}\right\}$. Set the initial arrival time to the origin vertex as zero ($T_{V_O}^{(0)} = 0$) and initial arrival time to all other vertices as infinity ($T_{V_i}^{(0)} = +\infty, \forall i \neq O$).
2. In iteration k, evaluate the time-dependent travel time $t_{E_{ji}} | T_{V_j}^{(k-1)}$ for all edges.
3. Update the arrival times $T_{V_i}^{(k)} = \min_{j \neq i} \left(T_{V_j}^{(k-1)} + t_{E_{ji}} | T_{V_j}^{(k-1)} \right)$ and $T_{V_O}^{(k)} = 0$.
4. Repeat steps 2-3 until the result converges.

Note that the edge travel time is infinity if a direct edge does not exist from a pair of vertices. Compared with the static Bellman-Ford algorithm, the time-dependent algorithm has an extra step of edge travel time prediction (step 2). Additionally, the revised algorithm searches for the optimal path in the reverse order (i.e., starting from the origin) so that travel time can be updated following the travel direction.

The convergence of the algorithm is ensured by the FIFO assumption. Basically the earlier arrival at any vertex will guarantee earlier arrival to any of its neighbor vertices if traveling from the vertex. Mathematically, the FIFO assumption can be expressed as

$$\left(T_{V_j}^{(k)} < T_{V_j}^{(k-1)} \Rightarrow T_{V_j}^{(k)} + t_{E_{ji}}|T_{V_j}^{(k)} < T_{V_j}^{(k-1)} + t_{E_{ji}}|T_{V_j}^{(k-1)}\right), \quad \forall i, j, k$$

The system optimum dynamic navigation algorithm aims to find the route with the minimum system cost. Edge travel times, however, is still required to determine when the time-dependent edge system costs need to be evaluated. Thus, the time-dependent shortest-path algorithms need to calculate both travel time and system cost in the path searching process.

For the same network representation, let $T_{s,i}$ denote the cumulative system cost at V_i and $t_{s,E_{ij}}|T$ is the time-dependent system cost associated with E_{ij}. The Dijkstra based algorithm to find the shortest path from V_O to V_D for DSO navigation is illustrates as

1. Initialize the network, set the cumulative system cost to the origin vertex as zero ($T_{s,V_O} = 0$) and the cumulative system costs to all other vertices as infinity ($T_{s,V_i} = +\infty, \forall i \neq O$). Set the arrival time to the origin vertex as zero ($T_{V_O} = 0$)
2. Initialize an empty parent hash map P to store the parent vertex of each vertex.
3. Initialize the "temporarily visited" group of vertices including the origin vertex, and the "unvisited" group of vertices including all other vertices. Create an empty group of "permanently visited" vertices.
4. Find the vertex V_i with the minimum cumulative system cost in the "temporarily visited" group. For all its unvisited neighbor vertices V_j, calculate the conditional edge travel time $t_{E_{ij}}|T_{V_i}$ and the conditional edge system cost $t_{s,E_{ij}}|T_{V_i}$.

5. Loop through all V_j. If we have $T_{s,V_i} + t_{s,E_{ij}}|T_{V_i} < T_{s,V_j}$, then update vertex V_j with the new cumulative system cost $T_{s,V_j} = T_{s,V_i} + t_{s,E_{ij}}|T_{V_i}$ and new arrival time $T_{V_j} = T_{V_i} + t_{E_{ij}}|T_{V_i}$. Update the parent hash map with $P_{V_j} = V_i$. Move V_j into the "temporarily labeled" group.
6. Move V_i into "permanently labeled" group.
7. Repeat steps 4-6 until the destination vertex V_D is in the "permanently visited" group. The shortest path can then be found through tracing the parent vertex from the destination all the way back to the origin.

Following the same notations, the Bellman-Ford based time-dependent shortest-path algorithm to find the system optimal route in DSO conditions includes the following steps

1. Start with an initial sequence of cumulative system costs $\left\{T_{s,V_i}^{(0)}\right\}$. Set the cumulative system cost to the origin vertex as zero ($T_{s,V_O}^{(0)} = 0$) and the cumulative system costs to all other vertices as infinity ($T_{s,V_i}^{(0)} = +\infty, \forall i \neq O$). Set the arrival time to the origin vertex as zero ($T_{V_O}^{(0)} = 0$) and arrival times to all other vertices as infinity ($T_{V_i}^{(0)} = +\infty, \forall i \neq O$).
2. In iteration k, evaluate the time-dependent edge travel time $t_{E_{ji}}|T_{V_j}^{(k-1)}$ and system cost $t_{s,E_{ji}}|T_{V_j}^{(k-1)}$ for all edges.
3. Update the cumulative system costs $T_{s,V_i}^{(k)} = \min_{j \neq i}\left(T_{s,V_j}^{(k-1)} + t_{s,E_{ji}}|T_{V_j}^{(k-1)}\right)$. Let j^* denote the vertex index that gives the minimum cumulative system cost, also update the arrival time as $T_{V_i}^{(k)} = T_{V_{j^*}}^{(k-1)} + t_{E_{j^*i}}|T_{V_{j^*}}^{(k-1)}$. Set the arrival time and cumulative system cost to the origin vertex both as zero ($T_{V_O}^{(0)} = T_{s,V_O}^{(0)} = 0$).
4. Repeat steps 2-3 until the result converges.

In both the Dijkstra and Bellman-Ford based DSO navigation algorithms, the cumulative system cost (T_{s,V_i}) is used as the key variable used in the shortest-path searching. While the system cost is being updated, the arrival times to the corresponding vertex is updated accordingly to keep track of the vehicle trajectory.

6.2.3 *Communication System Design*

The last section introduces the extensions of two classical shortest-path algorithms (Dijkstra and Bellman-Ford algorithms) to solve the time-dependent vehicle navigation problem. Both algorithms are developed based on the information exchange capability of the CVS. Specifically, the dynamic navigation algorithms require network traffic states being collected and predicted by RSUs, and vehicles getting such information through V2I communication. Although this study focuses on testing the algorithms in a microsimulation environment rather than implementing the algorithms in practice. It is still important to understand practical requirements and limitations from the algorithm implementation perspective. This section will briefly discuss important concerns related to developing a supporting communication system for the two typical navigation algorithms.

From the computation perspective, Dijkstra's algorithm stores the vertices in a sorted/priority order so that the shortest-path searching is more efficient. In the contrast, the Bellman-Ford algorithm consider all possible paths between vertices in each iteration, and the number of iterations required before the result converges increases with the network complexity. Thus, Dijkstra's algorithm is commonly known to be faster than the Bellman-Ford algorithm.

From the application perspective, however, other concerns like communication load and system design need to be carefully evaluated. As Dijkstra's algorithm keeps a sorted/priority list of vertices for fast shortest-path searching, the computational efficiency is actually achieved at the cost of storing the entire network information at one place. This may lead to high data load in certain part the communication network. Figure 6.1 illustrates a typical communication system for the Dijkstra based navigation algorithm. To gather enough information for time-dependent shortest-path calculation, the CV needs to talk to all RSUs in the road network. This will result in high communication load on the CV side. Additionally, the navigation distance is limited by the range of V2I communication. Due to these practical reasons, such a communication system is not preferred for implementation. But it could be potentially improved by aggregating RSU data at certain level or utilizing V2V data sharing to reduce the amount of V2I communication needed.

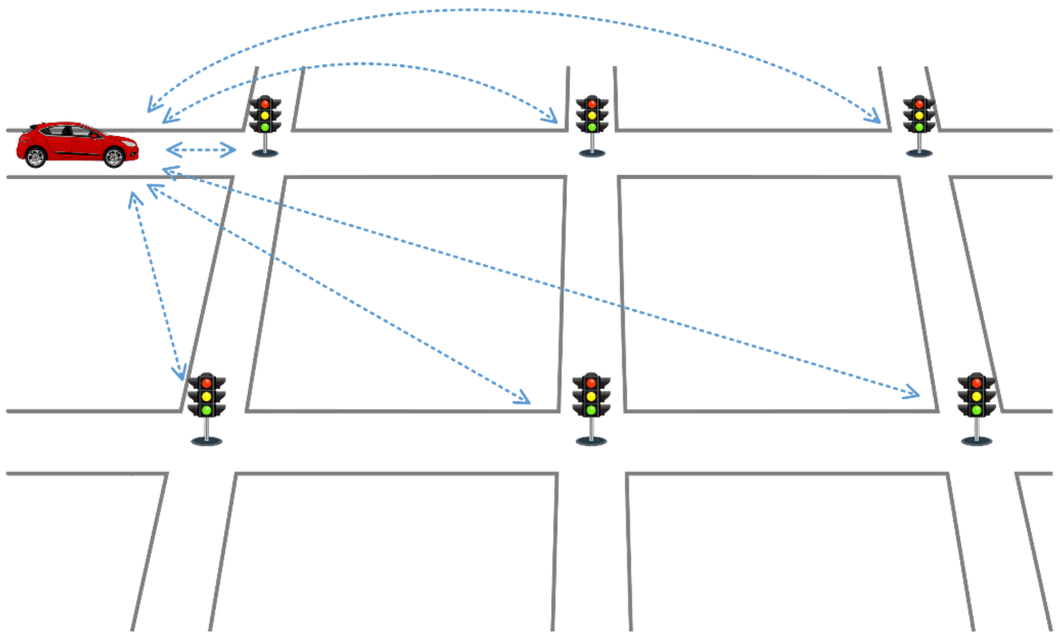


Figure 6.1. Communication system design for Dijkstra based navigation algorithm (without a cloud central database).

An alternative communication system to support the Dijkstra based navigation algorithm is illustrated in Figure 6.2, where a cloud central database is established and will communicate with both CVs and RSUs to build the data flow. In such a communication system, CVs do not directly talk with RSUs. Rather, the V2I communication is developed through the central database. The network traffic data collected from RSUs are aggregated by the central database and then sent to

CVs per each navigation request. Compared with the previous system design, this will certainly reduce the communication load on the CV side. The navigation distance is also not restricted by the communication range. Such a communication system is similar to that of the existing mobile-based map service applications, except that the traffic data are collected by RSUs rather than probe vehicles. Given the dynamic traffic states and frequent interaction between CVs and RSUs, however, the central database will become the focal point of data computation and communication. It is expected that extra cost will be generated in building and maintaining the central database, which could be a potential practical limitation of implementing such a system.

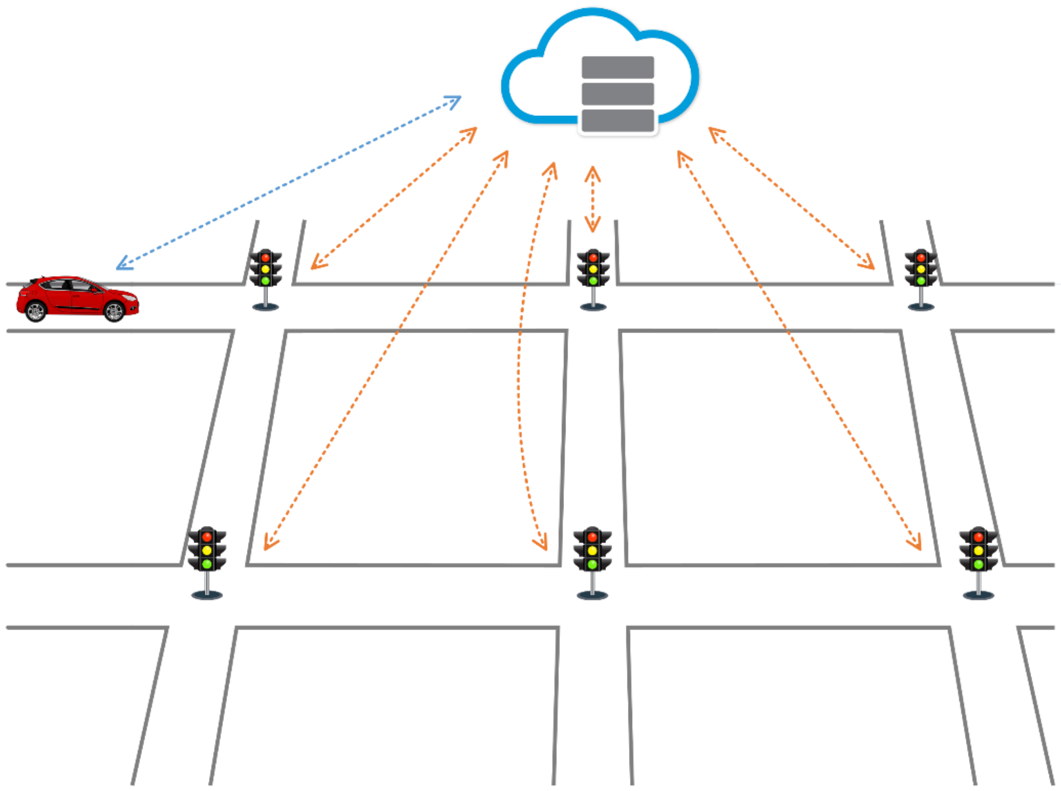


Figure 6.2. Communication system design for Dijkstra based navigation algorithm (with a cloud central database).

Figure 6.3 shows the typical communication system to support the Bellman-Ford algorithm. In this system design, each CV only needs to send the navigation request to its nearby RSU, and each RSU will communicate with the neighboring RSUs to disseminate the navigation request and update travel costs. Compared with the Dijkstra based algorithm, the Bellman-Ford based algorithm has the communication load more evenly distributed throughout the entire network. Recall that the Bellman-Ford algorithm requires several iterations to find the shortest-path. The total amount of communication requirement for the Bellman-Ford algorithm is not necessarily lower than that required by the Dijkstra based algorithm. To further improve such a system, communication between RSUs can potentially be integrated so that the same information can be shared with different CVs with similar navigation requests. Additionally, one may consider limiting the number of iterations to improve the computation speed at the cost of slightly reducing the result accuracy. Some other shortest-path searching techniques (e.g., map partitioning) may also be applied to improve the computational efficiency of the Bellman-Ford based algorithm.

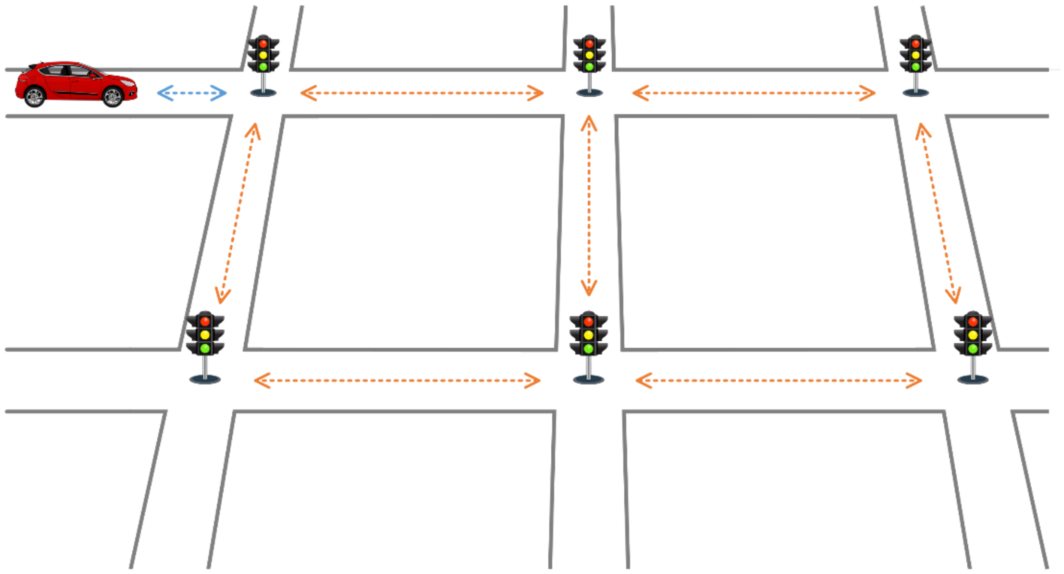


Figure 6.3. Communication system design for Bellman-Ford based navigation algorithm.

Note that in this study, two algorithms do not have significant performance difference because the studied network is relatively small with a finite number of road links. Thus, we only briefly mention the practical concerns in this section without further efforts to quantify the difference between algorithms. In the algorithm tests, both methods can give accurate shortest-path results and we will focus on analyzing the travel time impact of the dynamic navigation algorithms.

6.3 Coordinated Navigation Mechanism

According to the definition of DUE and DSO conditions, the optimal path is conditioned on route choice decisions of all other vehicles. In the time-dependent shortest-path searching, however, this can only be achieved if CVs can receive real-time route choice decisions from other vehicles. In this section, we will introduce a coordinated navigation mechanism to allow CVs share their real-time route choice decisions in the navigation process.

Instead of relying on V2V communication to share route choice decisions among CVs, this study will store CV route choices in RSUs so that it can be accessed by all CVs in real-time. Recall that in the dynamic navigation framework, an individual CV will first send the navigation request to and obtain network traffic states from RSUs. Thus, it is intuitively straightforward to update the network traffic states to include the CV once the optimal path is identified. Specifically, one unit of traffic flow will be added to the edges in the selected path following the expected CV trajectory. Following this method, the CV’s route choice decision will be incorporated in the time-dependent network traffic states, which will be utilized by all subsequent vehicles to find the conditional optimal path.

The coordinated navigation mechanism is implemented by a sequential updating approach in the sense that CVs will sequentially make their route choices. Specifically, the navigation mechanism can be illustrated in the following steps

1. Estimate the node and link travel time based on the real-time traffic conditions.
2. For one single CV, locate its current position and find its immediate downstream vertex V_i. Update its route as the time-dependent shortest-path from V_i to its destination V_D.

3. Update network traffic states based on the new CV’s route choice.
4. Repeat steps 1-3 and loop through all CVs until all vehicles do not change their optimal routes.

The key procedure in the coordinated navigation mechanism is that in each iteration only one CV will update its route choice. Compared to the case when all CVs change routes simultaneously, the proposed mechanism requires less computational resources. Additionally, each CV will identify its optimal route based on previous processed CVs as reflected by the updated network traffic conditions. The sequential updating approach will avoid all CVs making the same route choice, and hence avoid traffic fluctuations in the equilibrium state.

Following the coordinated navigation mechanism, in the equilibrium state every CV chooses the optimal route based on all other CVs’ route choice decisions. This is essentially the desired DTA conditions (i.e., DUE or DSO) from the network perspective. The convergence of the sequential updating navigation mechanism is proved in the static traffic assignment case (Du et al., 2015a). In this study, we assume the convergence still holds for the dynamic traffic assignment case. The complete assumption is stated as

If all vehicles sequentially navigate towards the time-dependent individual/system optimal route, the dynamic traffic assignment will converge to the DUE/DSO state.

Note that here the DTA equilibrium conditions are based on the real-time and predicted traffic states. To further utilize the future traffic data, CVs will also adaptively update their route choice decisions when new traffic data are collected. The coordinated navigation mechanism might need to loop through all CVs several rounds to finally reach an equilibrium dynamic traffic assignment. However, given the changing traffic states and traffic forecasting errors, the DTA equilibrium conditions will likely also vary with time. Thus, it is computationally inefficient to achieve the exact equilibrium state at every instant of time. A practical and potentially more efficient way is the adaptive coordinated navigation mechanism illustrated as follows

1. Sequentially update all CVs' route choice decisions using the coordinated navigation mechanism;
2. Once new traffic data are collected and processed, repeat step 1 and loop through all CVs just once.

The adaptive coordinated navigation mechanism assumes that traffic data are collected and processed at short time intervals, so that CVs can adaptively update their route choice decisions frequently. At each time interval when network traffic states are updated, we only loop through all CVs once. Although the resultant traffic assignment is not necessarily in the equilibrium state, through frequently updating CVs' route choice decisions based on real-time traffic data, the coordinate navigation results will approximate the equilibrium in a dynamic way.

Note that the coordinated navigation mechanism does not necessarily need extra implementation effort. In the real-world traffic environment, vehicles will start trips and request traffic information at different times. Thus, the route choices will naturally be sequentially determined and shared by different vehicles.

Chapter 7. VISSIM NAVIGATION MODULE

One of the major objectives of the study is to test the CV based navigation algorithm in a microscopic simulation environment. Although previous studies have developed simulation-based approaches to evaluate different DTA strategies, majority of the studies have been relying on numerical simulation models or analyzing at the mesoscopic scale. This study will build the vehicle navigation algorithms in a microscopic vehicular simulation platform. Specifically, this study developed a vehicle navigation module utilizing the application programming interface (API) in VISSIM, which is a widely used commercial simulation tool for microscopic traffic modeling and analysis.

7.1 INTRODUCTION TO VISSIM

VISSIM is a leading microscopic simulation software utilized by different users worldwide including research institutes, consulting companies, and public sectors (PTV, 2016a). It allows multimodal transportation modeling and operation analysis at detailed vehicular level, and thus have been deployed in numerous locations to answer various issues such as transportation planning, traffic control and management, and environmental impact.

7.1.1 *Simulation Model*

This study will utilize the VISSIM model of downtown Bellevue provided by the City of Bellevue. Detailed introduction of the VISSIM model can be found in Chapter 3. The network geometrics and traffic characteristics have been established and calibrated by the City of Bellevue and are thus used consistently without change in this study. Specifically, such information includes

- Road geometrics: link length, curvature, number of lanes, etc.

- Driving behaviors: designed speed, car-following model, accelerations/decelerations, etc.
- Intersection geometry: stop signs, conflict areas, priority rules, detector locations, etc.
- Traffic signals: signal control type, signal phasing/timing, etc.

In addition to the basic simulation model, hourly traffic count from midday off-peak period has also been collected and provided by the City of Bellevue. These data have been used as the background traffic in the simulation analysis.

7.1.2 *VISSIM COM API*

The development of the CV based navigation module is supported by the VISSIM Component Object Model (COM) API. COM provides data access to VISSIM objects, methods, and results, which can be adjusted or evaluated during or before/after simulation (PTV, 2016b). Additionally, COM supports a wide variety of languages (e.g., VBA, C, C++, MATLAB, and Python). Users are allowed to develop their own functions and applications which interact with COM objects to achieve different traffic control and operation purposes.

According to the COM API manual, VISSIM model objects that can be accessed in COM API are organized in a hierarchical structure, as shown in Figure 7.1. The highest-ranking object is IVissim, which must be called to access any sub-objects in the structure. INet is an important object including all sub-objects related to the model network (e.g., links, intersections, and vehicle input). VISSIM model parameters can also be accessed through the corresponding category (e.g., ISimulation contains simulation parameters that can be adjusted through COM API).

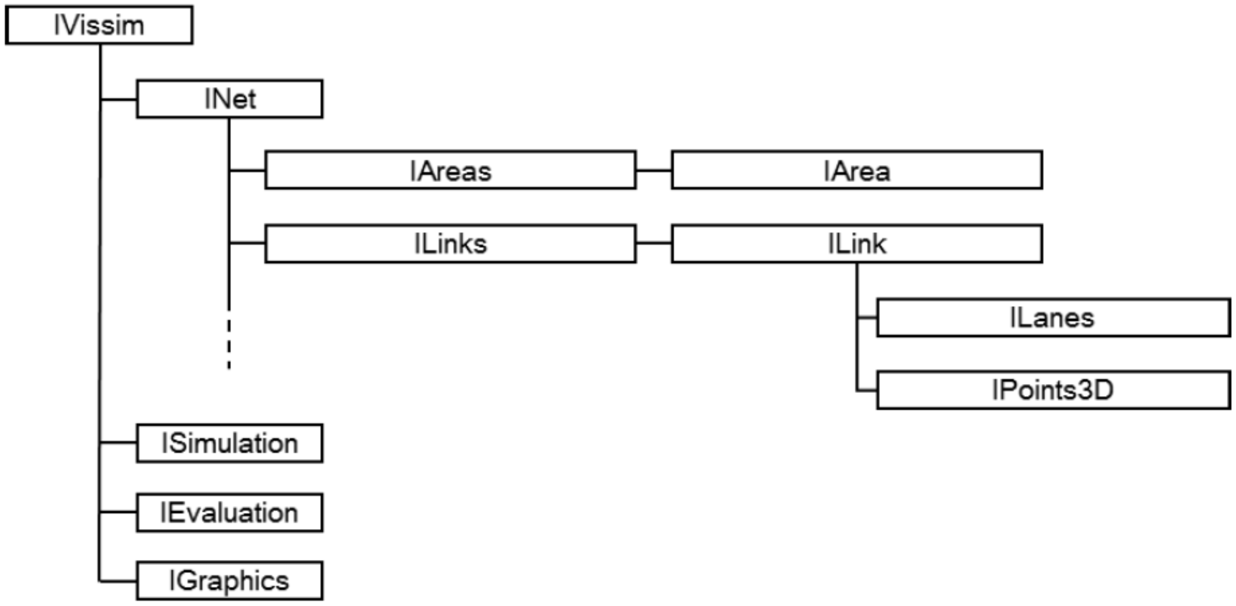


Figure 7.1. VISSIM COM object hierarchy.

This study developed a Python program to achieve the CV based navigation algorithms in the COM API. The program is integrated into the VISSIM model as a navigation module to dynamically navigate connected vehicles. During a simulation run, the navigation module will interact with a number of COM objects to achieve real-time data collection and coordinated navigation decisions.

7.2 VISSIM OBJECTS AND PARAMETERS

This section introduces major VISSIM objects and parameters that are used in the navigation module. Majority of the VISSIM network objects and model parameters have corresponding COM objects and attributes that can be accessed in the COM API.

7.2.1 *Vehicle Inputs*

Vehicle inputs are placed at the start points of road links and specify traffic flows that enter the studied network from different locations. Three groups of vehicle inputs are added to the simulation model, namely the background traffic, non-connected vehicles, and connected vehicles. To consider traffic flow randomness in the simulation experiment, all vehicle inputs assume the traffic volume type to be stochastic.

The background vehicle inputs are set up based on the hourly vehicle counts provided by the City of Bellevue. The background traffic also includes two percent of heavy goods vehicles (HGVs). Non-CV and CV vehicle inputs are added to the model at the corresponding route origins. Here we assume that all non-CVs and CVs are passenger cars. In the simulation experiments, different levels of traffic flow rate and CV penetration will be tested, and the non-CV and CV vehicle input values will be adjusted accordingly before simulation runs.

7.2.2 *Travel Time Measurement*

Real-time traffic volume and travel time collection is an important task in the CV navigation framework. In VISSIM, such data can be obtained using vehicle travel time measurements. Figure 7.2 shows an example vehicle travel time measurement in VISSIM. A vehicle time measurement consists of one start bar and one end bar. This study collects link-based travel time, and thus the start and end bars are placed at start points of different road link (note that the end bar of a travel time measurement is placed at the start of one downstream link). Every time a vehicle passes the start and end bar, the travel time in between is collected. In this fashion, the travel time vehicles spent at an intersection area will be included in the upstream link travel time. The average travel time and number of vehicles will then be aggregated by vehicle class and data collection interval defined in the evaluation parameters.

As illustrated in Figure 7.2, one road link typically needs three travel time measurements (assuming a four-way downstream intersection) to collect left-turn, through, and right-turn traffic, respectively. The collected vehicle counts and travel time data can be analyzed separately or aggregated to evaluate the overall link performance. In the studied network, majority of the traffic signals have a shared signal phase for both left-turn and through movements. Thus, the travel time and vehicle counts in different movements are aggregated at the link level for link performance modeling and real-time traffic data collection. By default, vehicle travel time measurements will collect the total vehicle count and average travel time for all vehicle classes. But the results can also be aggregated by vehicle class if the corresponding VISSIM evaluation parameter is properly specified.

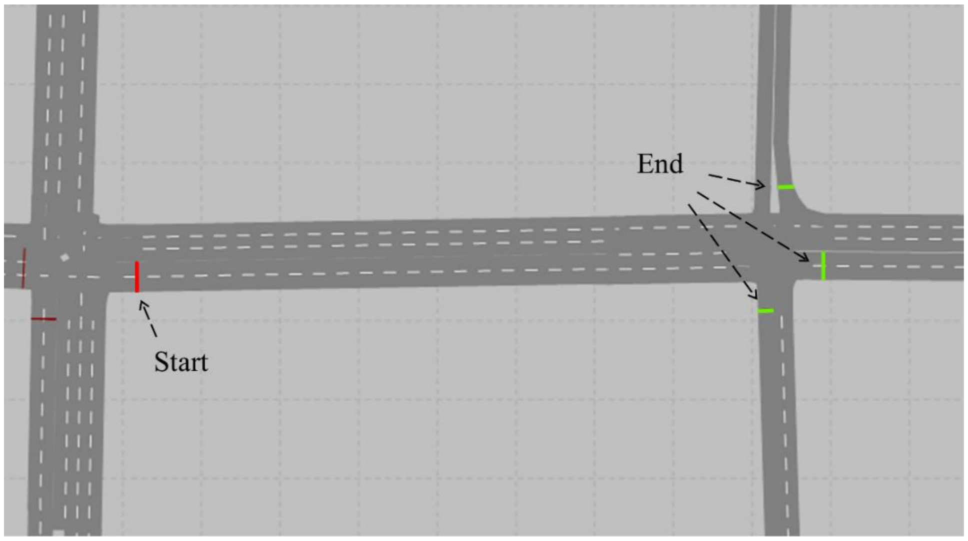


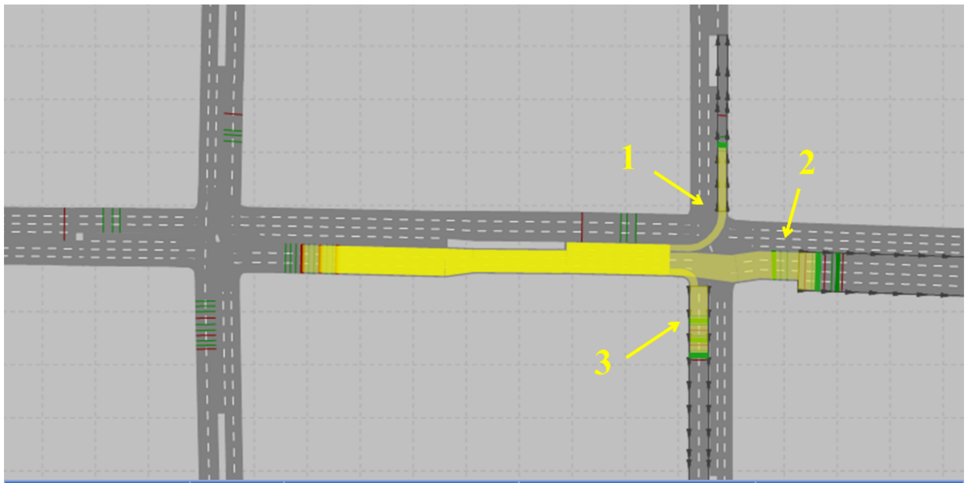
Figure 7.2. An example VISSIM vehicle travel time measurement.

Before testing the navigation algorithms, the vehicle travel time measurements have been used to collect traffic volume and average travel time at different links (and downstream intersections) under various levels of traffic input. The simulation results were then used to learn the relationship between link travel time and traffic flow rate (i.e., the link cost function). Details about the link cost function estimation can be found in Section 4.4.

7.2.3 *Vehicle Routing Decisions*

In VISSIM, vehicle movements are controlled by vehicle routing decisions. VISSIM provides two major types of vehicle routing decisions: static and dynamic vehicle routing decisions. Static vehicle routing decisions are placed at fixed locations and give the traffic flow distribution as relative flows for different movements. Dynamic vehicle routing decisions, in the contrast, has a built-in DTA model to distribute the network traffic flow to a number of alternative routes. The dynamic traffic assignment is determined using an iterative approach. The network is simulated multiple times and drivers are allowed to choose their paths based on the driving experiences in the preceding simulation. Such an approach is effective in calculating the theoretical optimal traffic assignment and provide insights for traffic planning and management. But the built-in DTA model is not a practical method as real-world drivers will not actually experience several different routes before selecting their path. Recall that this study developed a customized dynamic vehicle navigation methodology to approximate DTA equilibrium conditions through real-time communication in the CVS. Thus, instead of relying on the built-in DTA model, this study uses static vehicle routing decisions and dynamically change the relative flow distribution using COM API functions to achieve dynamic traffic assignment results.

Figure 7.3 shows an example static vehicle routing decision in the VISSIM model. One static vehicle routing decision typically consist of one start bar (i.e., the red bar) and several end bars (i.e., the green bars) corresponding to different routes, respectively. Once a vehicle passes the start bar, it will select one of the routes with probabilities derived from the relative flow rate. Note that as the route choice is decided for each discrete vehicle, the final traffic flow distribution will not be exactly equal to the relative flow rates. Instead, the relative flow values give an overall share of different route choices during the simulation run. As illustrated in the figure, the static vehicle routing decisions are placed at intersections, and each route corresponds to a particular movement (e.g., left-turn, through, or right-turn movement). Note that to ensure sufficient distance for vehicles to switch lanes, the start bar of the static vehicle routing decision needs to be placed near the start of the link.



Vehicle Routing Decision	Route No	Destination Link	Destination Position (ft)	Relative Flow (veh/h)
55	1	26: 110th Avenue NE	81.7	72.0
55	2	72: NE 8th Street	46.9	1192.0
55	3	97: 110th Avenue NE	43.0	95.0

Figure 7.3. An example VISSIM static vehicle routing decision.

The static vehicle routing decisions can also be set to only apply to a certain type of vehicles. Thus, three static vehicle routing decisions are placed at each approach for each intersection: one for background traffic, one for non-CVs, and one for CVs. For background traffic, the relative flows will be set equal to the vehicle counts of different movements at each intersection. For non-CVs, the relative flows will be set based on route choice decisions calculated from the static navigation algorithm. For CVs, the relative flows will be periodically updated based on dynamic navigation methods developed in COM API. Let $rf_{rd\_cv}(T)$ denote the relative flow for CV static vehicle routing decision rd_cv at time T, the dynamic relative flow can be calculated using the following equation

$$rf_{rd\_cv}(T) = \sum_{v\in\{CV\}} \sum_{P\ni rd} x_v^P \cdot \mathbb{I}\left(T \le T_v^{rd\_cv} < T + \Delta T\right) \tag{7.1}$$

$$\mathbb{I}\left(T \le T_v^{rd\_cv} < T + \Delta T\right) = \begin{cases} 1 & \text{if } T \le T_v^{rd\_cv} < T + \Delta T \\ 0 & \text{otherwise} \end{cases}$$

where $\{CV\}$ is the collection of all CVs. x_v^P is the vehicle route choice decision as previously defined. $T_v^{rd\_cv}$ is the expected arrival time at rd_cv (which equals to the corresponding edge arrival time). ΔT is the data collection and navigation updating interval.

7.2.4 *Basic Parameters*

In addition to network objects, COM API also allows access to basic VISSIM environmental parameters. Basic parameters that will be adjusted in the navigation module fall into two major categories: simulation and evaluation parameters. These parameters will be set as needed before simulation runs to apply different data collection plans and test the navigation algorithms in a variety of traffic conditions.

Table 7.1 lists basic VISSIM parameters that are adjusted in the navigation module using COM API functions. For simulation parameters, SimPeriod controls that each simulation run lasts for 2 hours (7200 seconds). Specifically, the first hour is generally considered as the “warm-up” period while the actual simulation results are collected during the second hour. The one-hour warm-up period is widely used by consulting companies and transportation agencies as an industrial practice. For the studied network, however, the network should be sufficiently warmed-up if the warm-up period is longer than fifteen minutes as the average travel time through the entire network is smaller than 10 minutes.

Random seed will be set before each simulation using the RandSeed COM attribute. For each model input, ten different random seeds will be used to obtain repeated results with variation. The NumRuns attribute can be used to repeatedly run the same simulation with different random seeds. In this study, as we need to dynamically adjust vehicle navigation decisions in each simulation, this function is not used and NumRuns is set as one. During a simulation, breakpoint can be set using the SimBreakAt attribute so that simulation will pause for collecting data and updating vehicle routing decisions. Note that when simulation stops at a breakpoint, network conditions (e.g., traffic signal states, vehicle locations and speed, etc.) will hold until the simulation resumes. In the simulation experiments, break points were set at 300 second intervals so that the vehicle navigation decisions are updated every five-minute.

Table 7.1. Basic VISSIM Parameters Adjusted in the Navigation Module

COM Attribute	**Parameter Type**	**Parameter Description**	**Parameter Value**
SimPeriod	Simulation	Total simulation seconds	7200
RandSeed	Simulation	Random seed	Vary
NumRuns	Simulation	Number of simulation runs	1
SimBreakAt	Simulation	Breakpoint simulation will stop at	300 intervals
VehClasses	Evaluation	Vehicle classes that travel time results will be aggregated by	Non-CV, CV, All
VehTravTmsCollectData	Evaluation	Whether to collect the vehicle travel time data	1 (true)
VehTravTmsFromTime	Evaluation	Start of the vehicle travel time data collection period	0 (from the start of the simulation)
VehTravTmsToTime	Evaluation	End of the vehicle travel time data collection period	99999 (to the end of the simulation)
VehTravTmsInterval	Evaluation	Time intervals that vehicle travel time results will be aggregated by	300

Evaluation parameters control how the simulation results are collected and organized. Specifically, the navigation module requires vehicle count and travel time collected by vehicle travel time measurements. The VehClass attribute can be specified as a list of vehicle classes the simulation results are aggregated by. As we will analyze detailed navigation effects on each vehicle class, the VehClasses attribute is set to aggregate results by non-CVs, CVs and all vehicles. The data collection period can be specified by the VehTravTmsFromTime and VehTravTmsToTime attributes. In this study, the traffic data are collected during the entire simulation, but only the second hour data will be used for result analysis. The vehicle count and travel time result can also be aggregate by time interval as specified by the VehTravTmsInterval attribute. As the navigation module will update vehicle routing decisions on five-minute basis, the traffic data will also be aggregated by 300-second intervals so that the most real-time data can be utilized.

In addition to the above-mentioned parameters, some other parameters (e.g., simulation speed, result format) are also specified to set up the basic simulation environment. In this section, we do not further introduce those parameters as they are not necessarily related to the navigation module framework.

7.3 NAVIGATION FUNCTIONS

This section will introduce navigation functions that compute time-dependent shortest-paths for individual vehicles.

7.3.1 *Data Structure*

As illustrated in Chapter 6, the shortest-path searching is performed on a directed graph representing the roadway network. In the navigation module, the directed graph is defined using the Python class structure. Upon initialization, a graph class has three basic instance objects

- Vertices – a list of vertices in the graph
- Edges – a list of edges in the graph
- Neighbors – a hash map providing the list of neighbor vertices for each vertex

While Vertices can be simply stored using the node number, Edges in a directed graph have a number of attributes to support the navigation calculation, such as the start/end node, link length, traffic flow, and link user/system cost. Note that Neighbors can be derived from Vertices and Edges. Here we pre-process and store Neighbors upon initialization of the graph to facilitate the computation speed of navigation methods.

Before navigation, vehicles will obtain network traffic information (e.g., from travel time measurements) and build a directed graph of the downtown Bellevue street network. In addition to the instance objects, the directed graph also has multiple navigation methods to be used by different classes of vehicles.

7.3.2 *Navigation Methods*

This study specifically considers three types of navigation algorithms: static navigation, DUE navigation and DSO navigation. Each algorithm corresponds to a navigation method in the graph class. This section will introduce the navigation methods using pseudo code which basically implements the algorithms introduced in Chapter 6.

Non-CVs do not have access to the real-time traffic data. The optimal path with the minimum total travel time is computed at the start of the trip based on the historical traffic data (i.e., background traffic in this study). Thus, all non-connected vehicles will choose the same path as they use the same set of historical data. The static navigation method used by non-CVs can be implemented following classical shortest-path algorithms. Specifically, this study applies Dijkstra’s algorithm and the Bellman-Ford algorithm for non-CV navigation. Dijkstra’s algorithm can be implemented using the following pseudo code

```
Initialization:
   Set the arrival time to the origin vertex as zero, i.e., $T_{V_O} = 0$
   Set arrival times to other vertices as infinity, i.e., $T_{V_i} = +\infty, i \neq O$
   Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
   Create a “temporarily visited” list of vertices, i.e., $L_{temp} = \{V_O\}$
   Create a “permanently visited” list of vertices, i.e., $L_{perm} = \{\}$
   Create an “unvisited” list of vertices, i.e., $L_{un} = \{V_i, i \neq O\}$

Shortest path searching:
   While $V_D$ not in $L_{perm}$
      Find $V_i$ in $L_{temp}$ where $T_{V_i}$ is the smallest
      For $V_j$ in neighbors of $V_i$ and $V_j$ not in $L_{perm}$
         If $T_{V_i} + t_{E_{ij}} < T_{V_j}$
            Update $T_{V_j} = T_{V_i} + t_{E_{ij}}$
            Set the parent vertex of $V_j$, i.e., $P_{V_j} = V_i$
            Remove $V_j$ from $L_{un}$
            Add $V_j$ to $L_{temp}$
         End if
      End for
      Remove $V_i$ from $L_{temp}$
      Add $V_i$ to $L_{perm}$

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
      Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

As an alternative method, the Bellman-Ford algorithm can be implemented using the following pseudo code

```
Initialization:
   Start with initial arrival times $\left\{T_{V_i}^{(0)}\right\}$. Set $T_{V_O}^{(0)} = 0$ and $T_{V_i}^{(0)} = +\infty, i \neq O$
   Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
   Initialize iteration counter $k = 0$

Shortest path searching:
   Start loop
      $k = k + 1$
      For $V_i$ in all vertices $\{V_i\}$
         Set $T_{V_i}^{(k)} = T_{V_i}^{(k-1)}$
         For $V_j$ in neighbors of $V_i$
            If $T_{V_j}^{(k-1)} + t_{E_{ji}} < T_{V_i}^{(k)}$
               Update $T_{V_i}^{(k)} = T_{V_j}^{(k-1)} + t_{E_{ji}}$
               Set the parent vertex of $V_i$, i.e., $P_{V_i} = V_j$
         End if
       End for
     Set $T_{V_O}^{(k)} = 0$
     If $\sum_i \left(T_{V_i}^{(k)} - T_{V_i}^{(k-1)}\right) \leq$ tolerance
         Break loop
   End loop

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
     Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

CVs have access to both historical and real-time traffic data and uses dynamic navigation methods based on time-dependent shortest-path searching. The DUE navigation method targets to find the route with the minimum user travel time. This study also provides two alternative DUE navigation methods by extending the classical Dijkstra's algorithm and the Bellman-Ford algorithm. The Dijkstra based DUE navigation algorithm can be implemented using the following pseudo code

```
Initialization:
  Set the arrival time to the origin vertex as zero, i.e., $T_{V_O} = 0$
  Set arrival times to other vertices as infinity, i.e., $T_{V_i} = +\infty, i \neq O$
  Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
  Create a "temporarily visited" list of vertices, i.e., $L_{temp} = \{V_O\}$
  Create a "permanently visited" list of vertices, i.e., $L_{perm} = \{\}$
  Create an "unvisited" list of vertices, i.e., $L_{un} = \{V_i, i \neq O\}$

Shortest path searching:
  While $V_D$ not in $L_{perm}$
    Find $V_i$ in $L_{temp}$ where $T_{V_i}$ is the smallest
    For $V_j$ in neighbors of $V_i$ and $V_j$ not in $L_{perm}$
      Estimate the time-dependent edge cost $t_{E_{ij}}|T_{V_i}$
      If $T_{V_i} + t_{E_{ij}}|T_{V_i} < T_{V_j}$
        Update $T_{V_j} = T_{V_i} + t_{E_{ij}}|T_{V_i}$
        Set the parent vertex of $V_j$, i.e., $P_{V_j} = V_i$
        Remove $V_j$ from $L_{un}$
        Add $V_j$ to $L_{temp}$
      End if
    End for
```

```
      Remove $V_i$ from $L_{temp}$
      Add $V_i$ to $L_{perm}$

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
      Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

As an alternative method, the Bellman-Ford based DUE navigation algorithm can be implemented using the following pseudo code

```
Initialization:
   Start with initial arrival times $\left\{T_{V_i}^{(0)}\right\}$. Set $T_{V_O}^{(0)} = 0$ and $T_{V_i}^{(0)} = +\infty, i \neq O$
   Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
   Initialize iteration counter $k = 0$

Shortest path searching:
   Start loop
      $k = k + 1$
      For $V_i$ in all vertices $\{V_i\}$
         Set $T_{V_i}^{(k)} = T_{V_i}^{(k-1)}$
         For $V_j$ in neighbors of $V_i$
            Estimate the time-dependent edge cost $t_{E_{ji}}|T_{V_j}^{(k-1)}$
            If $T_{V_j}^{(k-1)} + t_{E_{ji}}|T_{V_j}^{(k-1)} < T_{V_i}^{(k)}$
               Update $T_{V_i}^{(k)} = T_{V_j}^{(k-1)} + t_{E_{ji}}|T_{V_j}^{(k-1)}$
               Set the parent vertex of $V_i$, i.e., $P_{V_i} = V_j$
         End if
         End for
      Set $T_{V_O}^{(k)} = 0$
      If $\sum_i \left(T_{V_i}^{(k)} - T_{V_i}^{(k-1)}\right) \leq$ tolerance
         Break loop
   End loop

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
      Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

CVs that target to minimize the total system cost will utilize the DSO navigation algorithm to find the optimal path. Here the optimal path is defined as the path with the minimum system cost (or marginal cost). Similar to the DUE navigation, two DSO navigation methods are provided by extending classical shortest-path algorithms. The Dijkstra based DUE navigation algorithm can be implemented using the following pseudo code

```
Initialization:
   Set the cumulative system cost to the origin vertex, i.e., $T_{s,V_O} = 0$
   Set cumulative system costs to other vertices, i.e., $T_{s,V_i} = +\infty, i \neq O$
   Set the arrival time to the origin vertex, i.e., $T_{V_O} = 0$
   Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
   Create a “temporarily visited” list of vertices, i.e., $L_{temp} = \{V_O\}$
   Create a “permanently visited” list of vertices, i.e., $L_{perm} = \{\}$
   Create an “unvisited” list of vertices, i.e., $L_{un} = \{V_i, i \neq O\}$

Shortest path searching:
   While $V_D$ not in $L_{perm}$
      Find $V_i$ in $L_{temp}$ where $T_{V_i}$ is the smallest
      For $V_j$ in neighbors of $V_i$ and $V_j$ not in $L_{perm}$
         Estimate the time-dependent edge cost $t_{E_{ij}}|T_{V_i}$
         Estimate the time-dependent edge system cost $t_{s,E_{ij}}|T_{V_i}$
         If $T_{s,V_i} + t_{s,E_{ij}}|T_{V_i} < T_{s,V_j}$
            Update $T_{s,V_j} = T_{s,V_i} + t_{s,E_{ij}}|T_{V_i}$
            Update $T_{V_j} = T_{V_i} + t_{E_{ij}}|T_{V_i}$
            Set the parent vertex of $V_j$, i.e., $P_{V_j} = V_i$
            Remove $V_j$ from $L_{un}$
            Add $V_j$ to $L_{temp}$
         End if
   End for
      Remove $V_i$ from $L_{temp}$
      Add $V_i$ to $L_{perm}$

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
      Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

As an alternative method, the Bellman-Ford based DSO navigation algorithm can be implemented using the following pseudo code

```
Initialization:
   Start with initial cumulative system costs $\left\{T_{s,V_i}^{(0)}\right\}$
   Set $T_{s,V_O}^{(0)} = 0$ and $T_{s,V_i}^{(0)} = +\infty, i \neq O$
   Start with initial arrival times $\left\{T_{V_i}^{(0)}\right\}$. Set $T_{V_O}^{(0)} = 0$ and $T_{V_i}^{(0)} = +\infty, i \neq O$
   Create an empty parent hash map, i.e., $P_{V_i} = \text{none}$
   Initialize iteration counter $k = 0$

Shortest path searching:
   Start loop
      $k = k + 1$
      For $V_i$ in all vertices $\{V_i\}$
         Set $T_{s,V_i}^{(k)} = T_{s,V_i}^{(k-1)}$
         For $V_j$ in neighbors of $V_i$
            Estimate the time-dependent edge system cost $t_{s,E_{ji}} | T_{V_j}^{(k-1)}$
            If $T_{s,V_j}^{(k-1)} + t_{s,E_{ji}} | T_{V_j}^{(k-1)} < T_{s,V_i}^{(k)}$
               Update $T_{s,V_i}^{(k)} = T_{s,V_j}^{(k-1)} + t_{s,E_{ji}} | T_{V_j}^{(k-1)}$
               Set the parent vertex of $V_i$, i.e., $P_{V_i} = V_j$
               Estimate the time-dependent edge cost $t_{E_{ji}} | T_{V_j}^{(k-1)}$
               Update $T_{V_i}^{(k)} = T_{V_j}^{(k-1)} + t_{E_{ji}} | T_{V_j}^{(k-1)}$
         End if
         End for
      Set $T_{V_O}^{(k)} = 0$
      Set $T_{s,V_O}^{(k)} = 0$
      If $\sum_i \left(T_{V_i}^{(k)} - T_{V_i}^{(k-1)}\right) \leq$ tolerance
         Break loop
   End loop

Output the shortest path:
   Initialize the list of path vertices $L_{path} = \{V_D\}$
   While the leftmost vertex of $L_{path}$, $V_i$, is not $V_O$
      Append $P_{V_i}$ to the left of $L_{path}$
   $L_{path}$ contains all vertices in the shortest path in order
```

The output of the navigation method is a list of vertices in the identified optimal path. Such information will be used to update vehicle routing decisions in the VISSIM model. Additionally, CVs will send their route choices to RSUs which can update the predicted edge travel time and system cost. Thus, following CVs will anticipate the flow change and choose their optimal path conditional on previously processed CVs.

For non-CVs, the static navigation method will be called only once at the start of the trip, and the navigation decision will not change during the trip. For CVs, the dynamic navigation methods will also be called periodically (e.g., every five-minute) so that the optimal path can change adaptively with the real-time traffic conditions. Recall that in VISSIM, the vehicle movement is collaboratively determined by static vehicle routing decisions. Although the navigation method computes the optimal path for each CV, the route choice results are aggregated as relative movement flows to update the vehicle routing decisions at each intersection. From the macroscopic perspective, the network traffic flow distribution is ensured to be consistent with CV navigation results. But on the individual vehicle level, each CV does not necessarily follow the optimal path it identified. By using VISSIM vehicle routing decisions to update CV navigation results, we assume that CVs are interchangeable if they arrive at the same intersection during the same short period of time. Such an assumption will certainly constrain the model’s usefulness in vehicle-level analysis but will not influence the network-level analysis results (e.g., network traffic assignment, average/total travel time and system cost).

7.4 NAVIGATION MODULE STRUCTURE

The navigation module is built in a Python program which connect to the VISSIM COM API through the PyWin32 extension. Figure 7.4 shows the overall structure of the VISSIM navigation model. In each step of the framework, the navigation model will utilize corresponding COM objects to complete the task (e.g., set parameters, collect data, update routing decisions, etc.). As non-CVs use historical data to navigate, their routing decisions will be set before the start of the simulation. During the simulation, traffic data collection and CV navigation will be conducted in parallel at five-minute intervals. Note that CVs will sequentially navigate and communicate route choice decisions following the coordinated navigation mechanism introduced in Section 6.3. The CV routing decision objects in the VISSIM model will then be updated based on the combined CV navigation results.

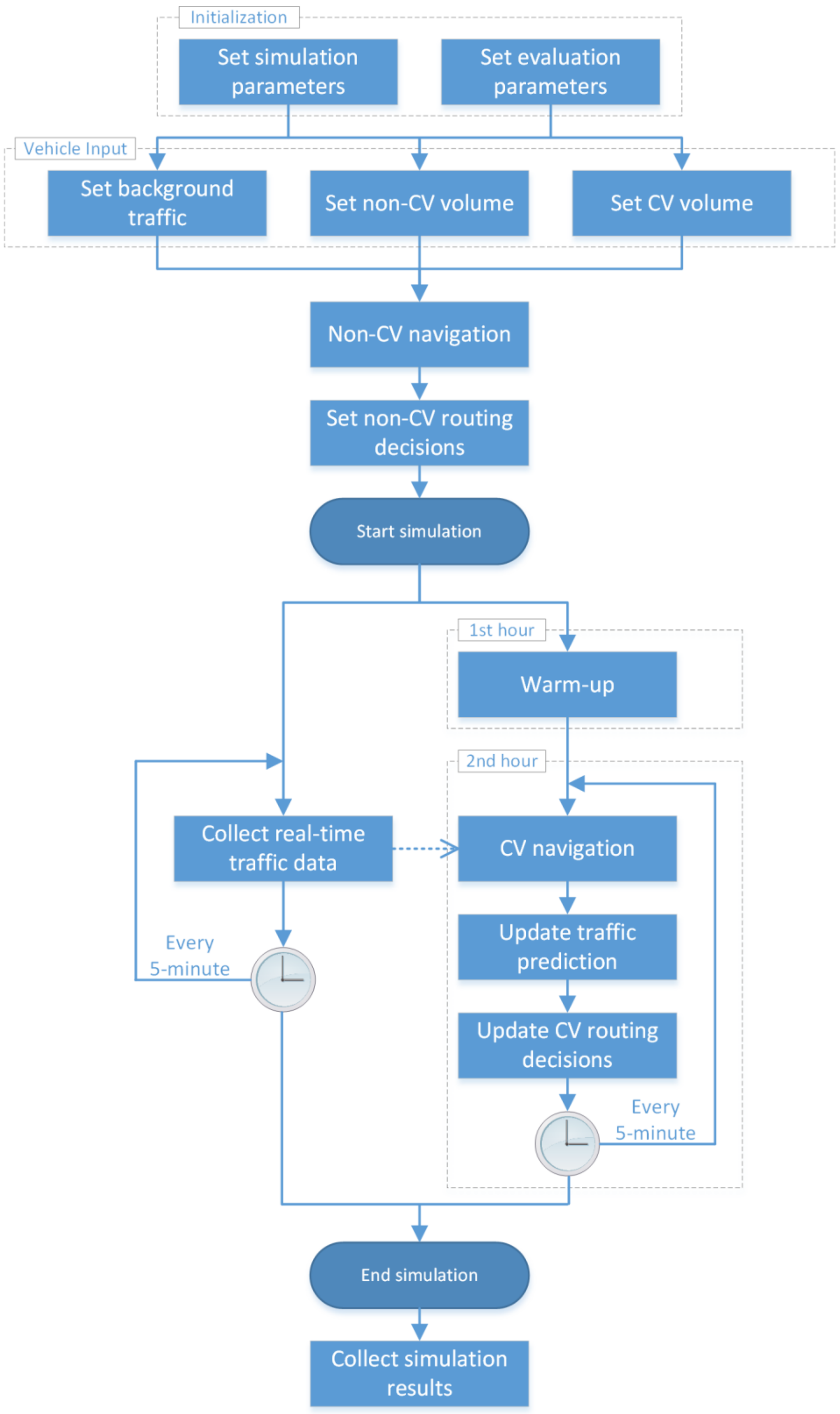


Figure 7.4. Structure of the VISSIM navigation module.

Although the vehicle navigation functions can be executed in real-time, the execution speed of the navigation module is mainly constrained by COM objects. Due to the hierarchical structure of VISSIM COM objects, all objects are sub-objects of the top-level object (i.e., IVissim). Thus, COM objects must be accessed or updated one at a time. For example, it is impossible to directly read the network traffic data as a table. Instead, the vehicle count and average travel time for each travel time measurement object must be read separately. The same procedure also applies when updating the vehicle routing decisions. Thus, simulation breakpoints at five-minute intervals are also established to ensure sufficient time for collecting data and updating vehicle routing decisions.

In the simulation experiments, five-minute (300 simulation seconds) is selected as the data collection and navigation updating interval. One of the key reasons is to balance the navigation frequency and execution speed. Due to the COM API structure, collecting real-time traffic data and updating vehicle routing decisions will typically take a couple minutes for the studied network size. Additionally, aggregate traffic data (i.e., vehicle count and travel time) by five-minute interval will also help reduce the impact of stochastic traffic fluctuations. If the data collection interval is too short, for example, the traffic data will have greater variance. If the data collection interval is too long (e.g., 30-minute), in contrast, real-time traffic changes may be over-smoothed, and CVs cannot change routes adaptively. Thus, five-minute is a preferred interval length which is long enough to smooth traffic fluctuations but can still reflect real-time traffic changes.

Chapter 8. ALGORITHM TESTS

The developed CV navigation algorithms are tested in both numerical and simulation experiments. Specifically, the goals of the algorithm tests include: 1) implement the developed CV navigation algorithms in numerical calculation and microscopic simulation platform; 2) quantify the navigation impacts in various traffic conditions and different CV penetration rates; 3) compare the results of DUE and DSO navigation algorithms; and 4) provide insights and recommendations for practical implementation.

8.1 EXPERIMENT DESIGN

The urban road network of downtown Bellevue will be used to test the CV navigation algorithms. To evaluate the navigation results in different scenarios, we designed a number of different experiment configurations that considers various real-world traffic cases (e.g., events, non-recurrent congestions, and different CV penetrations). Additionally, we also evaluate the navigation results in detail under different navigation objectives and algorithm parameters.

8.1.1 *Traffic Input*

Three types of vehicle input are added to the road network: background traffic, non-CVs and CVs. The background traffic is set using the actual hourly traffic count during the midday non-peak period provided by the City of Bellevue. In the experiments, it is assumed that vehicles in the background traffic do not change their routes. This part of traffic can be generally considered as familiar drivers with preferred routes (and thus do not use navigation applications in the area) or commercial vehicles that have planned routes. The VISSIM model was first simulated ten times with only the background traffic to develop link cost functions and summarize the average link traffic volumes that serve as historical traffic data in the CV navigation algorithms.

Non-CVs and CVs are added to the road network on top of the background traffic. Here we assume this extra traffic flow starts from the top-left corner and route towards the bottom-right corner of the network, so that all road links can potentially be traveled in a candidate path. This experiment design can be considered as a nonrecurrent event which generates an extra traffic flow in the studied network. Non-CV and CV navigation algorithms will be applied to corresponding vehicle classes to compute the optimal path for each vehicle, and we will evaluate the user cost as well as the system impact associated with the extra traffic flow. Additionally, we will also test different flow levels and CV penetration rates to evaluate the navigation results in various scenarios.

8.1.2 *Model Parameters*

An important part of the navigation algorithm is predicting the future travel time, which utilizes a Bayesian model that considers both historical and real-time traffic data. Theoretically, historical traffic data summarize typical traffic patterns and the travel time predictions will have smaller variance if only relying on historical data. But the travel time predictions could potentially be biased as historical traffic data do not reflect real-time traffic changes. If only relying on real-time traffic data, in contrast, the travel time predictions are expected to be unbiased but with greater variance. In the practical application, it is important to wisely determine the Bayesian parameter to balance the effect of historical and real-time traffic data.

In the simulation experiments, real-time traffic data are collected and aggregated by five-minute intervals. Here we use the traffic data from three most recent time intervals (i.e., the last 15-minute) as real-time traffic information in the travel time prediction task. As the historical data are aggregated on an hourly basis, we use 0.25 as the Bayesian parameter to reflect the approximate traffic amount ratio. In further experiments, we will also test different Bayesian parameters and discuss the impacts in some special cases (e.g., non-recurrent events).

8.2 EXPERIMENT RESULTS

This section presents the navigation results from both numerical and simulation experiments. Specifically, the actual path traveled by each vehicle is recorded to understand the network traffic distribution. The network travel time is also collected and summarized by vehicle type by link, in order to evaluate the navigation results in terms of user and system costs.

It is important to first verify that the simulation results agree with numerical calculations when applying the CV navigation algorithms. Figure 8.2 shows the comparison of the general results from numerical and simulation experiments. Here we change the extra flow rate from 100 to 800 veh/h and assume all vehicles in the extra traffic flow are CVs. The trends of total user cost (i.e., total CV travel time) from numerical calculation are presented by curves, while results from simulation experiments are presented by scatter points. Note that we also considered two cases when CVs are using DUE and DSO navigation algorithms, respectively, and show the results in different colors and line types (or point shapes). In general, the simulation results agree with numerical calculation in the sense that points are closely distributed near the curves. The relationship between the CV flow rate and the total CV travel time is convex because the average travel time per vehicle will increase with the CV flow rate. Variance of simulation results increase with the CV flow rate, indicating higher traffic randomness when the network is more congested. Note that the differences between DUE and DSO navigation algorithms are minimal from numerical results and are unobservable from simulation results. Detailed comparisons between DUE and DSO navigation algorithms in various scenarios will be presented in following sections and thus are not further discussed here.

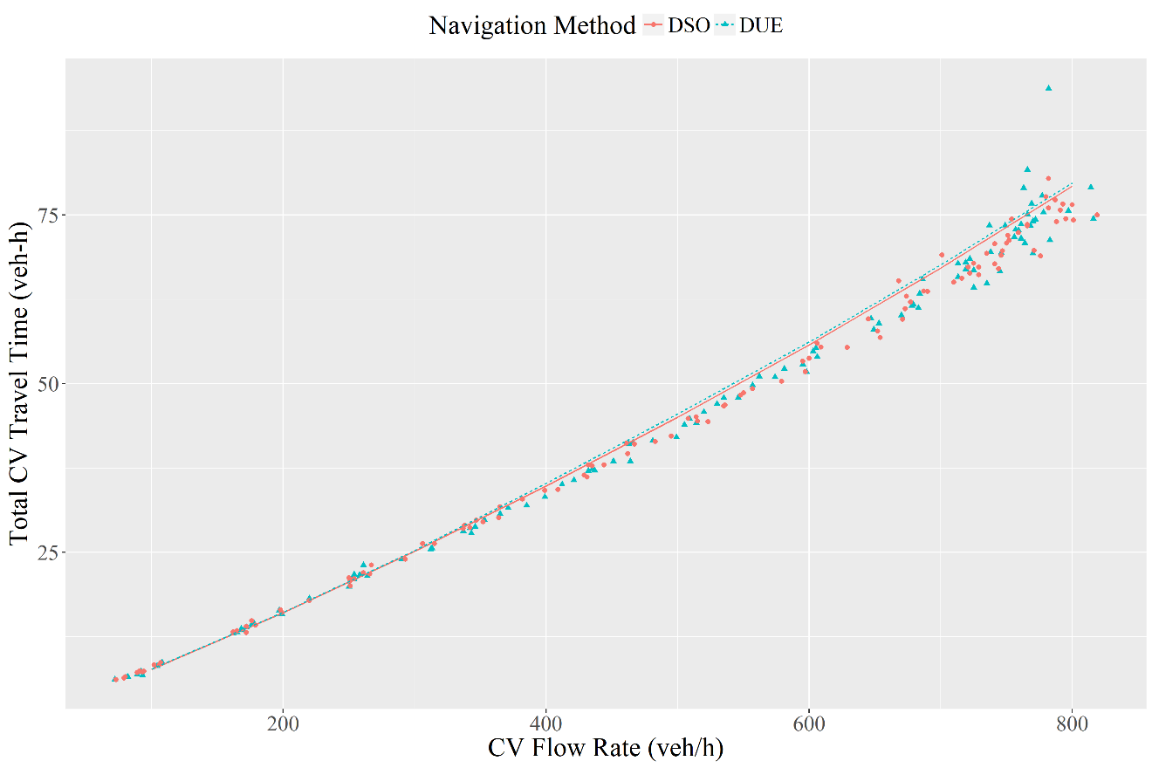


Figure 8.1. Summary of navigation results from numerical and simulation experiments.

It is important to note that the amount of traffic that can be added into the network is constrained by the intersection capacity at the entry node. In the studied network, the maximum extra traffic flow is around 800 veh/h, which is approximately equivalent to 30% of the background traffic. No more extra traffic can be added to the network without reducing the background traffic. Additionally, when the extra traffic flow is 800 veh/h, a number of traffic signals are already operated close to the capacity. Thus, there is no need to further increase the extra traffic flow as this study focuses on navigate CVs when the traffic is uncongested.

8.2.1 *Network Traffic Distribution*

We first look at the network traffic distributions under different navigation algorithms. Figure 8.2 shows the distribution of the extra traffic on all road links when the extra traffic flow rate is

100 veh/h. The first subplot presents the distribution of the extra traffic when all vehicles use the static navigation algorithm (i.e., all vehicles are non-CVs). In this case all vehicles choose the same path based on the background traffic (i.e., historical traffic data). In this scenario, the extra traffic flow is equivalent to around 3% of the total background traffic. Thus, the network traffic conditions will not be greatly influenced even if all extra vehicles choose the same path. However, this clearly is not the optimal traffic distribution as there exists multiple alternative paths from the origin to the destination. The second and third subplots present the network traffic distributions based on two CV navigation algorithms (i.e., the DUE and DSO navigation algorithms), respectively, assuming all extra vehicles are CVs. According to the graph, the DUE navigation algorithm mainly utilized two fastest paths, while the DSO navigation algorithm utilized more road links and achieves a more balanced network traffic distribution especially near the start and the end of the trip.

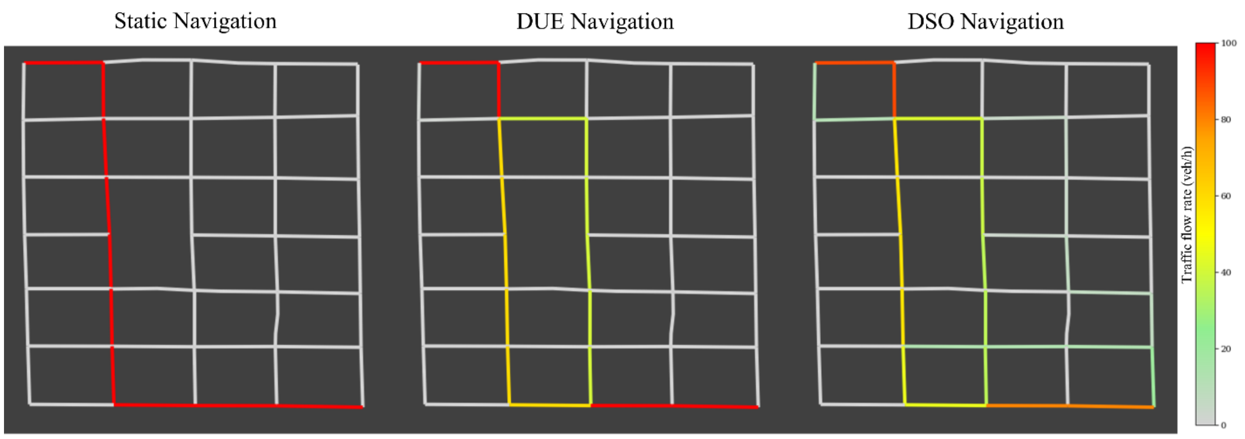


Figure 8.2. Traffic flow distribution for different navigation algorithms (100 veh/h).

Figure 8.3 - Figure 8.5 show the network traffic distributions if further increasing the extra traffic flow to 300, 500, and 800 veh/h, respectively. The static navigation results are the same as all non-CVs will still choose the same path identified by historical traffic data. Dynamic navigation algorithms will utilize more roads with the increase of CV flow rate. When the CV flow rate is

800 veh/h, for example, approximately 80% of the roads will be traveled by some CVs, and majority of the utilized roads will receive around 20% - 60% of the total extra traffic. Comparing DUE and DSO navigation results, the overall traffic distributions under DUE and DSO navigation algorithms become more similar with the increase of CV flow rate. The difference between DUE and DSO navigation results is not very obvious from the network traffic distribution graph. One observable difference between the two subplots is near the destination. DSO navigation can slightly alleviate traffic demand on major roads (i.e., the bottom horizontal street) near the destination, which gives a more balanced traffic distribution in the network.

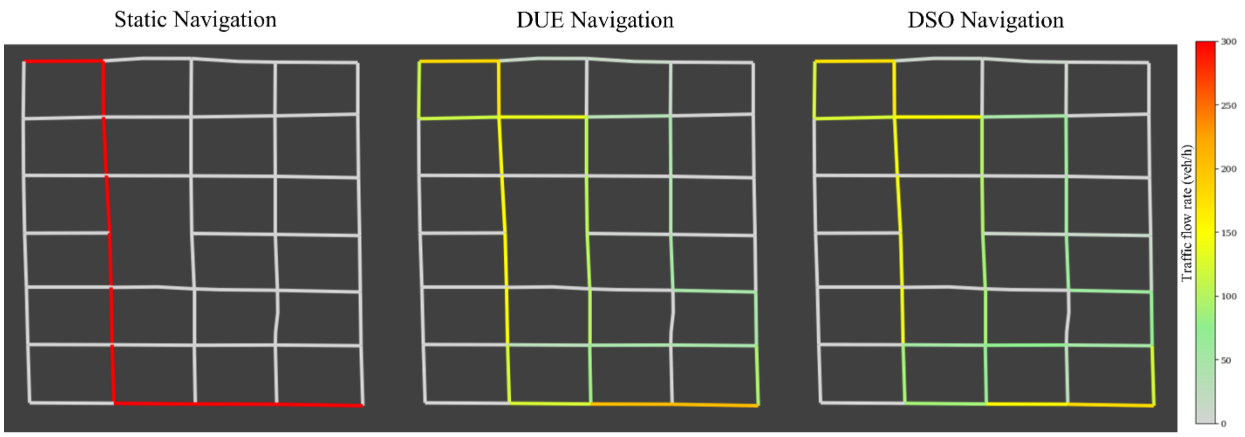


Figure 8.3. Traffic flow distribution for different navigation algorithms (300 veh/h).

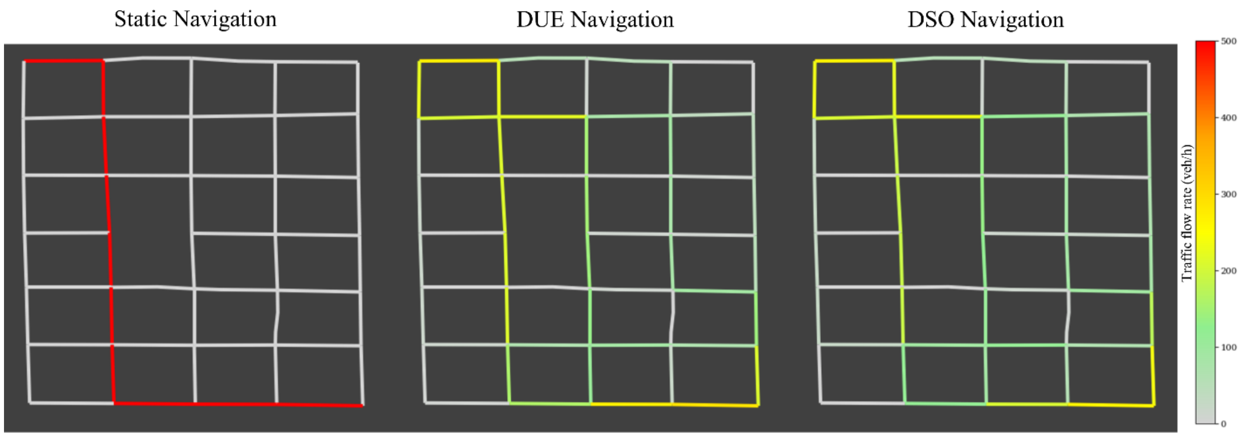


Figure 8.4. Traffic flow distribution for different navigation algorithms (500 veh/h).

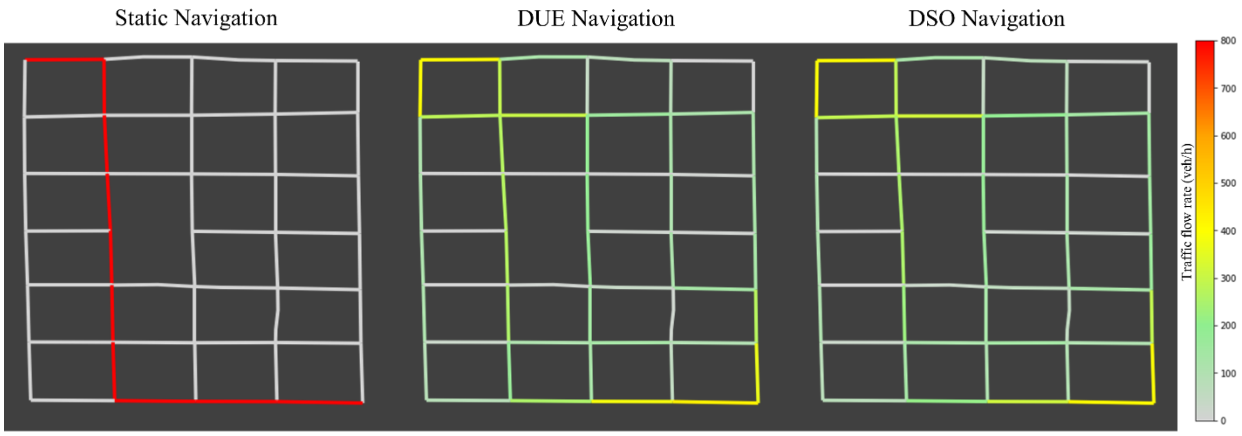


Figure 8.5. Traffic flow distribution for different navigation algorithms (800 veh/h).

According to the experiment design, the extra traffic flow specifically considers the case of a non-recurrent traffic flow increase. In this scenario, non-CVs will choose the same path as the traffic increase is not observed in historical traffic data. If the extra traffic is added as a long-term flow increase, then some of the travel time impact can be reflected from historical traffic data. As a result, non-CVs can potentially utilize more than one path and the network traffic distribution for static navigation could actually be somewhere between the static and dynamic navigation results in the above figures.

To quantify the differences between network traffic distributions, we define the path difference rate between two navigation algorithms as

$$PathDiff(1,2) = \frac{\sum_E |q_{1,E} - q_{2,E}|}{2\sum_E q_{1,E}} \tag{8.1}$$

where $q_{1,E}$ and $q_{2,E}$ are the link traffic volume under algorithms 1 and 2, respectively. The range of the difference rate is between 0 (same) and 100% (completely different). Note that the path difference rate will not change if we switch the places of the two algorithms (i.e., $PathDiff(1,2) = PathDiff(2,1)$). This is because the total traffic volume is the same when comparing the two algorithms in the same traffic scenario (i.e., $\sum_E q_{1,E} = \sum_E q_{2,E}$, which both equal to the extra traffic volume).

Table 8.1 summarizes path difference rates under different levels of extra traffic flow. According to the table, differences between the static navigation and dynamic navigations will increase with the traffic flow rate. This is intuitive as more roads will be traveled when more CVs are added to the network. Recall that static navigation algorithm will assign all traffic to the fastest path identified by historical data. The historical fastest path still play an important role in dynamic navigation algorithms in the sense that it is traveled by around 35% CVs when the CV flow rate is 800 veh/h. The path difference rate between two dynamic navigation algorithms (i.e., DUE and DSO navigation algorithms) has a decreasing trend with the CV flow rate. When the CV flow rate is 800 veh/h, for example, the path difference rate between DUE and DSO navigation results is 5.2%, which is hard to observe in the network traffic distribution graph (Figure 8.5)

Table 8.1. Path Difference Rate between Navigation Algorithms

Extra traffic flow (veh/h)	Path difference rate (%)		
	Static vs. DUE	Static vs. DSO	DUE vs. DSO
100	24.2	34.0	11.7
200	38.7	47.7	11.2
300	46.2	52.7	8.7
400	50.9	57.4	9.2
500	55.5	61.1	8.0
600	58.6	63.1	6.7
700	60.8	64.6	5.9
800	62.5	65.7	5.2

It is important to note that the network traffic distributions and path difference rates are calculated from numerical experiments, which do not sufficiently consider traffic randomness in a dynamic environment. In simulation experiments, the dynamic traffic assignment will change periodically with the real-time traffic data collection (e.g., every five-minute). Thus, the actual network traffic distribution may be slightly different from the numerical results at a specific time instant. In this section, we analyze numerical experiment results to help understand the average traffic distribution and expected traffic distributions. Further analysis of simulation results and discussion about result variations will be presented in the following sections.

8.2.2 *Impact of CV Penetration*

In the previous section, we show that dynamic navigation algorithms can distribute CVs on many alternative paths. However, 100% CV penetration is not realistic in the recent future. Thus, it is important to test the navigation algorithms under different levels of CV penetration. In this section, we further consider the extra traffic as a mixed flow of non-CVs and CVs. While non-CVs continue to choose the optimal path based on the historical knowledge of traffic, CVs will dynamically calculate the optimal path using DUE or DSO navigation algorithms. Vehicle input with different CV penetrations have been added to the VISSIM model, and the travel time results were collected and analyzed to quantify the impact.

Figure 8.6 shows the relationship between average path travel time travel time and CV penetration at different levels of extra traffic flow. Note that the VISSIM model was simulated ten times with different random seeds for each specific scenario (i.e., as defined by traffic flow level, CV penetration, and dynamic navigation algorithm). Due to the stochastic vehicle input, the actual percentage of CVs that actually entered the network in each simulation will be slightly different unless the CV penetration is 0% or 100%. In the graph, the simulation results are plotted based on the detected CV penetration. In general, the average path travel time decreases with the increase of CV penetration in all scenarios, and the benefit of CVs will increase with more traffic being added to the network. Under low or moderate traffic flow levels (i.e., 100 or 300 veh/h), the travel time impact of CV penetration is relatively small (but still observable) compared to the result variations. If the vehicles are fully connected (i.e., 100% CV penetration), the average path travel time under 500 veh/h traffic flow can be reduced to the similar level as that under the minimum traffic flow (i.e., 100 veh/h). This indicates that dynamic navigation algorithms can efficiently assign traffic into different paths and reduce user cost to the minimum level.

When the CV penetration rate is higher than 60%, the system will not benefit from further increasing the CV penetration in the sense that the marginal reduction of user cost is minimal. This is consistent with the numerical experiment results that around 35% of CVs will still choose the historical fastest path even if the extra traffic flow is high. If changing these 35% CVs to non-CVs, their route choice decisions will still be the same, and the network traffic distribution will not be greatly influenced. That said, the mechanisms behind the similar network traffic distributions are different. When the traffic is not fully connected, for example, CVs must rely on traffic detectors to collect non-CV traffic data and then calculate the conditional optimal path. If all vehicles are CVs, they can share route choice decision in real time without relying on traffic detectors, and the dynamic traffic assignment can be achieved faster.

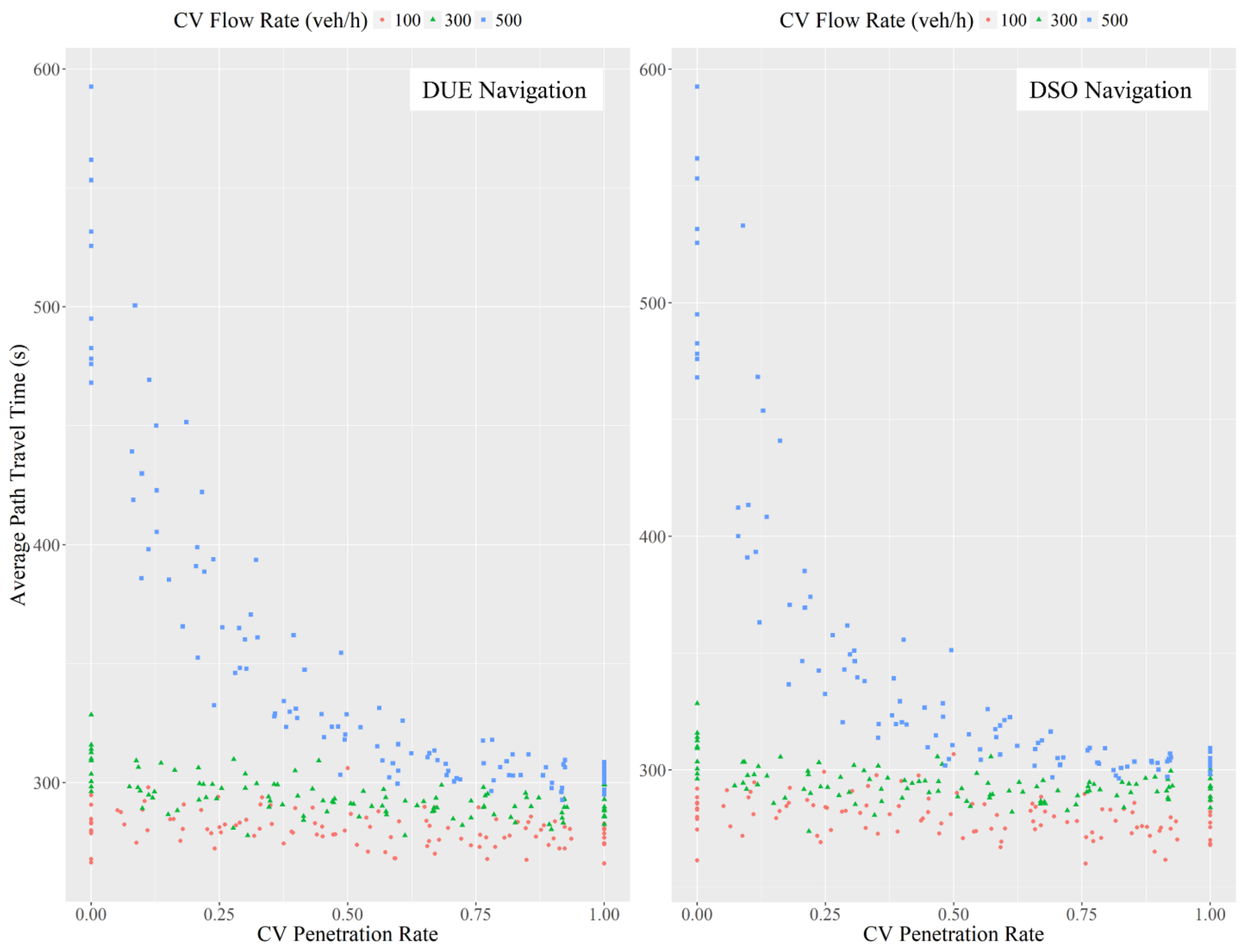


Figure 8.6. Change of system and extra travel cost with CV penetration rate.

According to Figure 8.6, two dynamic navigation algorithms have similar travel time impacts, which is consistent with the previous finding that DUE and DSO navigation algorithms will lead to similar network traffic distributions. In the following section, we will further compare these two algorithms to analyze the benefit of DSO navigation in terms of the system impact.

8.2.3 *Benefit of DSO Navigation*

Figure 8.7 shows the change of total user cost (left subplot) and total system cost (right sub plot) with CV penetration based on DSO and DUE navigation algorithms from simulation experiments. To reduce the impact of travel time variations, results under 500 veh/h extra traffic flow rate are presented as the data trends are more obvious. According to the figure, the total user and system costs will decrease with higher CV penetration under both DUE and DSO navigation algorithms. The maximum benefit is achieved when the CV penetration reaches around 60%.

Figure 1.1Figure 8.8 shows the average trends of navigation results. When the CV penetration is lower than 50%, DSO navigation results are better than DUE in the sense that both user and system costs drop faster. This indicates that implementing the DSO navigation algorithm will help reduce user and system costs when CVs only make up a small portion of the traffic. When the CV penetration is close to 100%, the difference between DUE and DSO navigation results are relatively small. When considering individual simulation runs, however, the benefit of the DSO navigation algorithm is not reliable due to variations in travel time results. In a dynamic traffic environment, multiple sources of randomness (e.g., stochastic vehicle arrivals, actuated signal controls, dynamic route choice decisions, etc.) will cause variations in vehicle travel time results. Based on the simulation results, the variation of hourly vehicle travel time is around 5% of the result.

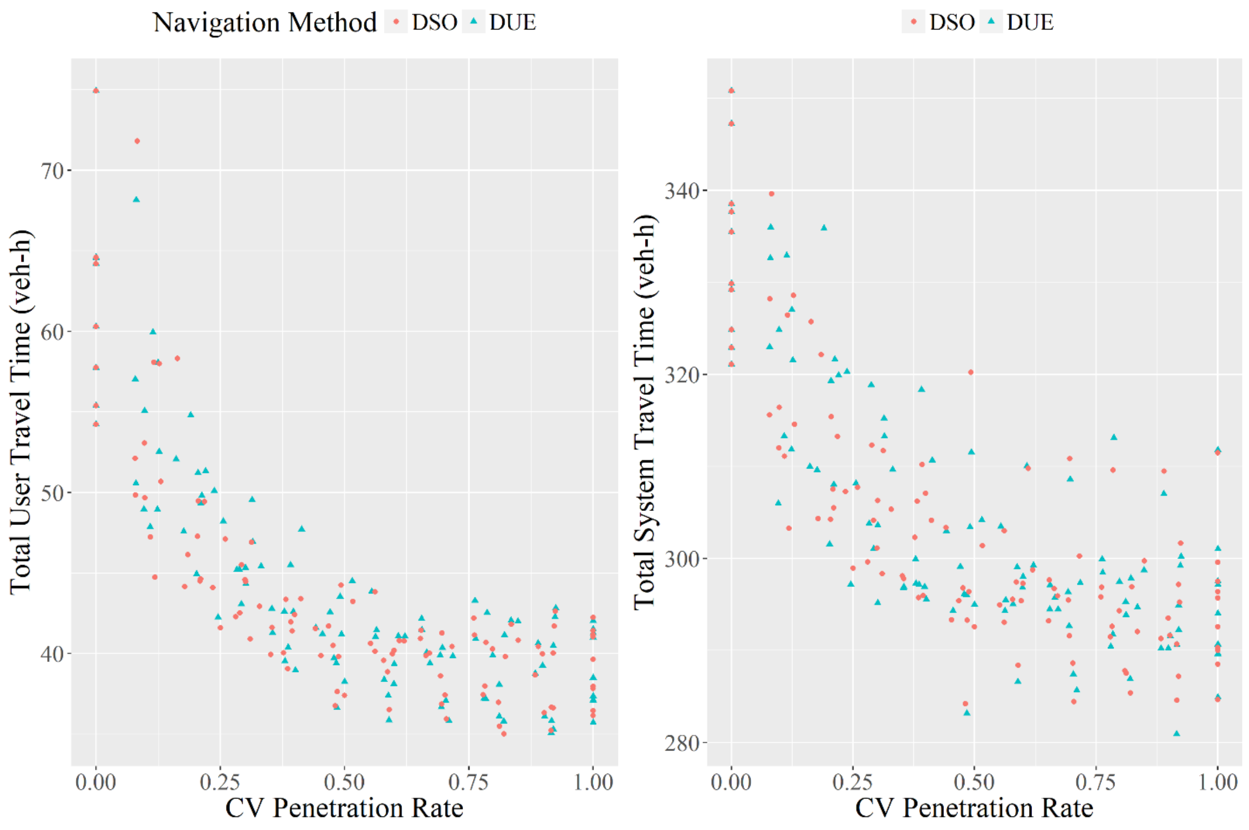


Figure 8.7. Comparison of DUE and DSO navigation results.

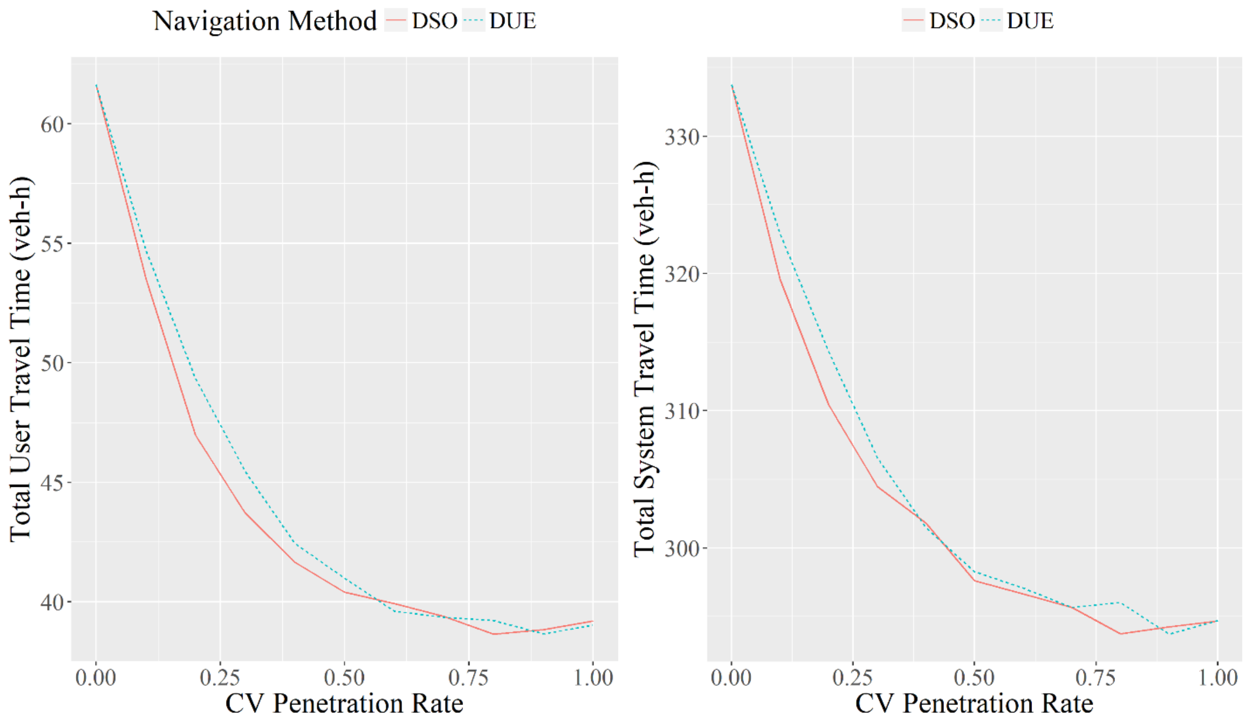


Figure 8.8. Comparison of DUE and DSO navigation results (trends).

To further understand the potential benefit of the DSO navigation algorithm, Table 8.2 summarizes the comparison of user and marginal system costs between DUE and DSO navigations based on numerical experiments. In all traffic scenarios, the marginal system cost is equivalent to approximate 120-140% of the total user cost, and the ratio slightly increases with the traffic flow rate. Comparing DUE and DSO results, the DSO navigation algorithm will benefit the system more than users in the sense that the system benefit is around twice of the user benefit. This is intuitive as the DSO navigation algorithm targets to minimize the total system cost. The relative benefit of the DSO navigation algorithm is small (e.g., less than or around 1% in all scenarios). Such a small benefit may not be practically achievable considering real-world traffic flow randomness.

Table 8.2. Comparison of DUE and DSO Navigation Results from Numerical Experiments

Traffic Flow (veh/h)	User Cost				System Cost (Marginal)			
	Total Cost under DUE (veh-h)	Total Cost under DSO (veh-h)	Benefit of DSO (veh-h)	Benefit of DSO (%)	Total Cost under DUE (veh-h)	Total Cost under DSO (veh-h)	Benefit of DSO (veh-h)	Benefit of DSO (%)
100	8.08	8.05	0.03	0.37	10.90	10.86	0.04	0.37
200	16.96	16.88	0.08	0.47	23.05	22.89	0.16	0.69
300	26.59	26.48	0.11	0.41	36.33	36.04	0.29	0.80
400	37.05	36.69	0.36	0.97	50.88	50.32	0.56	1.10
500	47.88	47.41	0.47	0.98	66.36	65.65	0.71	1.07
600	59.16	58.70	0.46	0.78	82.78	82.01	0.77	0.93
700	71.18	70.68	0.50	0.70	100.36	99.45	0.91	0.91
800	83.95	83.44	0.51	0.61	119.11	118.11	1.00	0.84

The small benefit of the DSO navigation algorithm is because of the fact that DUE and DSO navigation algorithms give very similar network traffic distributions in the studied network. Recall that the path difference rate between DUE and DSO traffic assignments is around 5%-10% in all scenarios. With the increase of CV flow rate, the DUE and DSO traffic assignment becomes even more similar. As a result, the advantage of DSO over DUE navigation is negligible in the studied network after considering real-world traffic flow randomness.

8.2.4 *Balancing Historical and Real-Time Data*

The last part of result analysis involves the discussion about navigation algorithm parameters, as well as the robustness to traffic fluctuations. In order to navigate CVs adaptively under real-time traffic conditions, dynamic navigation algorithms need to utilize real-time traffic data collected from traffic detectors. Future traffic flow is predicted as a weighted average that combines historical and real-time traffic data. Specifically, the Bayesian parameter, γ, (see Equation 6.15) controls the "weight" of real-time traffic data as compared to the historical traffic data.

In the simulation model, historical traffic is defined as the average hourly traffic flow and travel time results under only the background traffic input. The real-time traffic is defined as the average traffic flow and travel time results from most recent three data collection intervals (i.e., the latest 15-minute) under both background traffic and extra traffic flow input. In previous simulation experiments, $\gamma = 0.25$ was used so that the weight of real-time traffic data is consistent with the length of the data collection period. Additionally, as the background traffic is the same as the that used to simulate historical data, real-time traffic will also follow historical traffic patterns. Thus, adjusting the Bayesian parameter will not significantly affect the navigation results.

When the real-time traffic does not follow historical traffic patterns, for example, when accidents and non-recurrent events cause traffic changes, the effectiveness of dynamic navigation algorithms will be influenced by the specific Bayesian parameter used in the travel time prediction model. To test the impact of the Bayesian parameter, we increase the background traffic to 120%

(note that the historical data are still based on 100% background traffic). This scenario can be generally regarded as incidents or events that create non-recurrent traffic demands higher than historical traffic patterns. The extra traffic flow is set as 500 veh/h with 100% CVs. Multiple simulation experiments have been conducted with different Bayesian parameter values to test the robustness of dynamic navigation algorithms under non-recurrent traffic changes.

Figure 8.9 presents the relationship between the average path travel time and the log-transformed Bayesian parameter. Two subplots show the travel time results under DUE and DSO navigation algorithms, respectively. The scatter points are slightly jittered in the horizontal direction to avoid overlapping. For the DUE navigation results, the average path travel time first decreases and then increases with the increase of the Bayesian parameter. There exist an optimal range of the Bayesian parameter (e.g., $-1 < \log(\gamma) < 1$) that approximately gives the minimum average path travel time. In terms of the result variation, the variance of the average path travel time has an increasing trend with the Bayesian parameter, indicating that over-weighting real-time traffic data may result in higher variations in navigation results. As for the DSO navigation results, we cannot clearly observe an optimal range of the Bayesian parameter, possibly because the DSO traffic assignment are more consistent in regular and non-recurrent traffic conditions. But the variance of the average path travel time will also increase with the Bayesian parameter (i.e., when the traffic prediction relies more on real-time data)

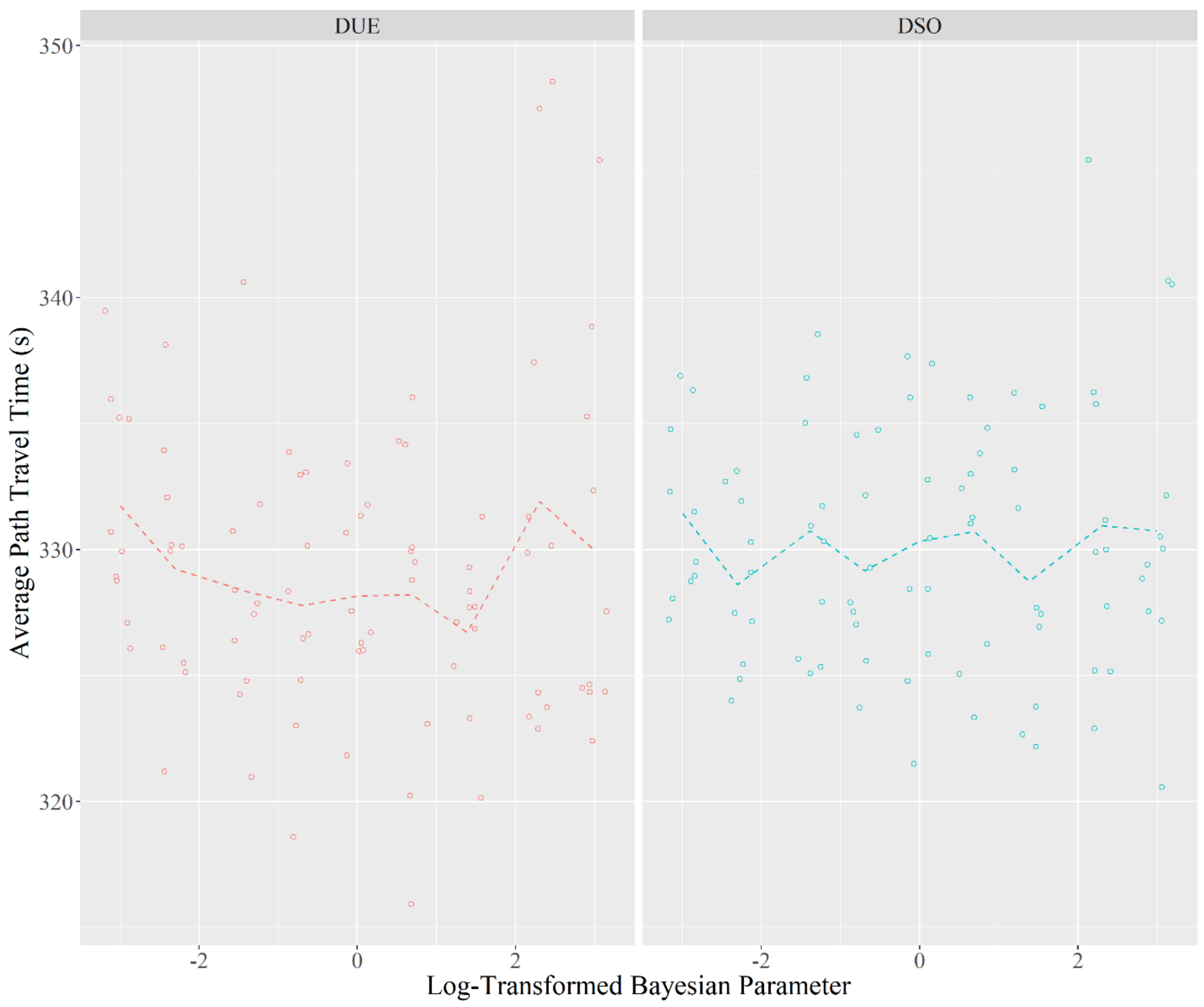


Figure 8.9. Impact of Bayesian parameter on navigation results.

Note that the simulation results are consistent with the bias-variance tradeoff in travel time prediction. Based on previous discussions, traffic predictions based on historical data should have less variance but could be potentially biased. While predictions based on real-time data will be unbiased but have greater variance. In the practical application, the Bayesian parameter should be appropriately determined based on the local traffic patterns in order to balance historical and real-time traffic data in travel time prediction tasks.

Chapter 9. CONCLUSIONS AND RECOMMENDATIONS

9.1 SUMMARY OF RESULTS

This study developed a coordinated navigation system that can adaptively navigate CVs towards DUE and DSO traffic assignment conditions. Different from macroscopic DTA models, the developed navigation algorithms are specifically suitable for urban street networks with intersections. An empirical intersection delay function has been developed based on classical probabilistic intersection delay models. The delay function parameters can be empirically learned from actual traffic data collected by detectors at each intersection. Based on the intersection delay function, the future traffic states can be predicted combining historical and real-time traffic data. Then CVs will coordinately select different alternative paths through real-time sharing of their routing decisions.

The developed navigation algorithms have been implemented in numerical and simulation experiments to quantify the impacts. Results show that compared to static navigation algorithm used by non-CVs, CV navigation algorithms can effectively utilize more roads and achieve a more balanced network traffic distribution. As a result, both user and system costs will be greatly reduced. The developed CV navigation algorithms are helpful in mitigate traffic impact caused by non-recurrent events. The benefit of CV navigation algorithms will also increase (i.e., the user and system costs will decrease) with the increase of CV penetration. In all experiments, the user and system costs cannot be further reduced when the CV penetration reaches around 60%. This indicates that the traffic does not need to be fully connected to achieve the maximum benefits from CV navigation algorithms. Note that the maximum beneficial CV penetration threshold is similar to the path difference rate between static and dynamic navigation results. This indicates that if we know the difference between the current and designed network traffic assignments, we can introduce the exact amount of CVs to fill this gap and lead the system to the desired state. The efficiency of the CV impact is guaranteed by the real-time traffic data collection and V2I communication.

An important finding from the algorithm tests is that the benefit of using the DSO navigation algorithm is relatively small (e.g., around 1%) compared to the DUE navigation results. Given the real-world traffic randomness in the urban street network, such a small improvement may not be practically achievable. Additionally, DSO is not an equilibrium state consider the user optimality. Extra efforts such as incentives and policies are required to encourage the use of DSO navigation. The benefit of DSO navigation is more significant when CV penetration is low or when the network is more congested. This indicates that encouraging DSO navigation may help faster achieve the CV benefit in the early phase of CV deployment.

It is important to note that DUE and DSO states have very similar network traffic distributions in the studied network. This is possibly because the studied area has a grid network consisting of road sections with relatively homogeneous roadway characteristics (e.g., link length, design speed, traffic control, etc.). As a result, there exists many alternative paths with similar features for each vehicle. With the increase of the CV flow rate, both DUE and DSO navigation algorithms will lead to balanced traffic distributions in the network. Thus, the marginal benefit of DSO navigation over DUE navigation is negligible after considering the traffic flow randomness. It is expected that the benefit of DSO navigation will be greater in a network with more diverse road sections. Similar analysis could be applied to quantify the navigation impacts and provide insights in the practical implementation of the developed CV navigation system.

9.2 DISCUSSIONS AND FUTURE DIRECTIONS

This study developed a navigation module that can dynamically control CV movements in the VISSIM simulation model. Specifically, the developed DUE and DSO navigation algorithms are implemented to test the travel time impact. Note that the use of the navigation module is not restricted to testing the proposed CV navigation algorithms. Other dynamic navigation algorithms can also be implemented as a graph method in the navigation module program. Additionally, extra real-time data, if needed by a different navigation algorithm, can be collected from the simulation model by deploying additional sensors. But the vehicle routing decisions can be dynamically updated in the same fashion.

To ensure the robustness of dynamic navigation algorithms in non-recurrent congestions, this study applied a Bayesian model for the traffic forecasting task. Specifically, future traffic is predicted using a combination of historical and real-time traffic information. This study uses the Bayesian model mainly because the link cost is written as a function of traffic flow, and stochastic vehicle arrivals can be assumed to follow typical probabilistic distributions (e.g., Poisson distribution). The Bayesian model can also be replaced by other traffic forecasting algorithms such as time-series and machine learning models to improve the traffic prediction accuracy. As this study mainly relies on the simulated traffic network to test proposed navigation algorithms, traffic forecasting is not a major focus on this study. The Bayesian model is sufficient to provide accurate traffic predictions and ensure the effectiveness of CV navigation algorithms in various conditions.

According to the fundamental diagram of traffic flow theory, the relationship between travel time and traffic flow has two branches, the uncongested and congested branches. This study focuses on the uncongested branch so that the link travel time is a convex function of the traffic flow rate. As a result, the coordinate CV navigation algorithms only work when the traffic is uncongested. This is intuitive as when the traffic is already congested, there is no much benefit of dynamically navigating CVs. Note that the “uncongested prerequisite” only applies to non-CV traffic, as the route choice of CVs will be coordinately determined to avoid road sections with near-capacity traffic flows. Thus, as long as there exist some paths where non-CV traffic flows do not exceed the capacity, and the capacity of alternative paths can serve the CV traffic demand, the coordinate CV navigation algorithms can efficiently distribute CVs into different paths to approximate network traffic distribution towards different DTA states.

Although the CV navigation algorithms are designed to provide routing guidance to each vehicle at any instant of time, implementation of the algorithms are constrained by the limitation of the simulation platform. For example, the vehicle navigation updating frequency is constrained by the data reading and writing speed of VISSIM COM API. Additionally, in VISSIM vehicle movements are collaboratively controlled by vehicle routing decisions. Thus, the CV routing decisions must be aggregated to update the relative flows at different vehicle routing decisions in the network. Such an approximation will not greatly influence the network-level travel time results but will lose specific control of each individual vehicle. Future research may focus on developing vehicular simulation module that can specifically control the movement of each vehicle. In VISSIM, however, this could be achieved at the cost of reducing the navigation updating speed or simplifying the roadway network to ensure data can be collected and processed in real-time.

In term of the navigation objectives, this study uses travel time as the measurement of travel cost, and the link travel time is thus expressed as a function of link traffic flow rate. In practice, the actual travel cost may include other components such as social costs. For example, commercial vehicles may want to avoid local roads in residential areas to reduce the safety and environmental impact. Depending on the nature, social cost can be considered as independent of or a function of the traffic flow. The general travel cost can then be represented as a combination of travel time and social cost. The CV navigation algorithms can be adjusted accordingly towards user of system optimality as defined by the general travel cost.

The dynamic navigation algorithms rely on CV communications (i.e., V2I and V2V communications) to adaptively identify the optimal path based on real-time traffic information and other vehicles' route choice decisions. This study did not specifically consider the quality of CV communications. In practical implementations, however, communication quality such as network latency and transmission quality are critical to ensure the reliability and effectiveness of CV navigation algorithms. An interesting research direction is to quantify the communication load at different parts of the network and optimize the communication system design, in order to ensure the quality of CV communications.

BIBLIOGRAPHY

Ban, X. (Jeff), Liu, H.X., Ferris, M.C., Ran, B., 2008. A link-node complementarity model and solution algorithm for dynamic user equilibria with exact flow propagations. Transportation Research Part B: Methodological 42, 823–842. https://doi.org/10.1016/j.trb.2008.01.006

Beckmann, M., McGuire, C.B., Winsten, C.B., 1956. Studies in the economics of transportation. Yale University Press, New Haven.

Bellman, R., 1958. On a routing problem 4.

Bliemer, M.C.J., Bovy, P.H.L., 2003. Quasi-variational inequality formulation of the multiclass dynamic traffic assignment problem. Transportation Research Part B: Methodological 37, 501–519. https://doi.org/10.1016/S0191-2615(02)00025-5

Chen, C.L.P., Zhou, J., Zhao, W., 2012. A Real-Time Vehicle Navigation Algorithm in Sensor Network Environments. IEEE Transactions on Intelligent Transportation Systems 13, 1657–1666. https://doi.org/10.1109/TITS.2012.2201478

Chen, P.-A., Kempe, D., 2008. Altruism, selfishness, and spite in traffic routing, in: Proceedings of the 9th ACM Conference on Electronic Commerce - EC '08. Presented at the the 9th ACM conference, ACM Press, Chicago, Il, USA, p. 140. https://doi.org/10.1145/1386790.1386816

Çolak, S., Lima, A., González, M.C., 2016. Understanding congested travel in urban areas. Nature Communications 7, 10793. https://doi.org/10.1038/ncomms10793

Dantzig, G.B., Ramser, J.H., 1959. The Truck Dispatching Problem. Management Science 6, 80–91. https://doi.org/10.1287/mnsc.6.1.80

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271. https://doi.org/10.1007/BF01386390

Dion, F., Rakha, H., Kang, Y.-S., 2004. Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections. Transportation Research Part B: Methodological 38, 99–122. https://doi.org/10.1016/S0191-2615(03)00003-1

Du, L., Chen, S., Han, L., 2015a. Coordinated Online In-Vehicle Navigation Guidance Based on Routing Game Theory. Transportation Research Record: Journal of the Transportation Research Board 2497, 106–116. https://doi.org/10.3141/2497-11

Du, L., Han, L., Chen, S., 2015b. Coordinated online in-vehicle routing balancing user optimality and system optimality through information perturbation. Transportation Research Part B: Methodological 79, 121–133. https://doi.org/10.1016/j.trb.2015.05.020

Eklund, P.W., Kirkby, S., Pollitt, S., 1996. A dynamic multi-source Dijkstra's algorithm for vehicle routing, in: 1996 Australian New Zealand Conference on Intelligent Information Systems. Proceedings. ANZIIS 96. Presented at the 1996 Australian New Zealand Conference on Intelligent Information Systems. Proceedings. ANZIIS 96, pp. 329–333. https://doi.org/10.1109/ANZIIS.1996.573976

FCC, 1999. FCC Allocates Spectrum 5.9 GHz Range for Intelligent Transportation Systems Uses [WWW Document]. Federal Communications Commission. URL https://transition.fcc.gov/Bureaus/Engineering_Technology/News_Releases/1999/nret9006.html (accessed 1.6.19).

Ford, L.R., 1956. Network flow theory. RAND Corporation P-923.

Frank, H., 1969. Shortest Paths in Probabilistic Graphs. Operations Research 17, 583–599. https://doi.org/10.1287/opre.17.4.583

Friesz, T.L., Bernstein, D., Smith, T.E., Tobin, R.L., Wie, B.W., 1993. A Variational Inequality Formulation of the Dynamic Network User Equilibrium Problem. Operations Research 41, 179–191.

Fu, L., 2001. An adaptive routing algorithm for in-vehicle route guidance systems with real-time information. Transportation Research Part B: Methodological 35, 749–765. https://doi.org/10.1016/S0191-2615(00)00019-9

Gendreau, M., Guertin, F., Potvin, J.-Y., Taillard, É., 1999. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. Transportation Science 33, 381–390. https://doi.org/10.1287/trsc.33.4.381

Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R., 2003. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research 151, 1–11. https://doi.org/10.1016/S0377-2217(02)00915-3

Groot, N., De Schutter, B., Hellendoorn, H., 2015. Toward System-Optimal Routing in Traffic Networks: A Reverse Stackelberg Game Approach. IEEE Transactions on Intelligent Transportation Systems 16, 29–40. https://doi.org/10.1109/TITS.2014.2322312

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics 4, 100–107. https://doi.org/10.1109/TSSC.1968.300136

Hoff, P.D., 2009. A first course in Bayesian statistical methods, Springer texts in statistics. Springer, London ; New York.

Ichoua, S., Gendreau, M., Potvin, J.-Y., 2006. Exploiting Knowledge About Future Demands for Real-Time Vehicle Dispatching. Transportation Science 40, 211–225. https://doi.org/10.1287/trsc.1050.0114

Kammoun, H.M., Kallel, I., Casillas, J., Abraham, A., Alimi, A.M., 2014. Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model. Transportation Research Part C: Emerging Technologies 42, 147–167. https://doi.org/10.1016/j.trc.2014.03.003

Koonce, P., Rodegerdts, L., 2008. Traffic signal timing manual. Federal Highway Administration, United States.

Luby, M., Ragde, P., 1989. A bidirectional shortest-path algorithm with good average-case behavior. Algorithmica 4, 551–567. https://doi.org/10.1007/BF01553908

Ma, R., Ban, X. (Jeff), Pang, J.-S., 2014. Continuous-time dynamic system optimum for single-destination traffic networks with queue spillbacks. Transportation Research Part B: Methodological 68, 98–122. https://doi.org/10.1016/j.trb.2014.06.003

Mahmassani, H.S., 2016. 50th Anniversary Invited Article—Autonomous Vehicles and Connected Vehicle Systems: Flow and Operations Considerations. Transportation Science 50, 1140–1162. https://doi.org/10.1287/trsc.2016.0712

Mahmassani, H.S., 2001. Dynamic network traffic assignment and simulation methodology for advanced system management applications. Networks and Spatial Economics 1, 267–292. https://doi.org/10.1023/A:1012831808926

Mannering, F.L., Washburn, S.S., 2013. Principles of Highway Engineering and Traffic Analysis, 5th ed. John Wiley & Sons, Inc.

Merchant, D.K., Nemhauser, G.L., 1978. A Model and an Algorithm for the Dynamic Traffic Assignment Problems. Transportation Science 12, 183–199.

Miller, A.J., 1963. Settings for Fixed-Cycle Traffic Signals. Operations Research 14, 373–386. https://doi.org/10.2307/3006800

Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T., 2007. Partitioning graphs to speedup Dijkstra's algorithm. Journal of Experimental Algorithmics 11, 2.8. https://doi.org/10.1145/1187436.1216585

Muñoz, J.C., Laval, J.A., 2006. System optimum dynamic traffic assignment graphical solution method for a congested freeway and one destination. Transportation Research Part B: Methodological 40, 1–15. https://doi.org/10.1016/j.trb.2005.01.001

Newell, G.F., 1960. Queues for a fixed-cycle traffic light. Annals of Mathematical Statistics 31, 589–597.

NOCoE, 2018. SPaT Challenge Overview [WWW Document]. National Operations Center of Excellence. URL https://transportationops.org/spatchallenge (accessed 1.3.19).

NTOC, 2012. National Traffic Signal Report Card: Technical Report. National Transportation Operations Coalition.

Peeta, S., Ziliaskopoulos, A.K., 2001. Foundations of dynamic traffic assignment: The past, the present and the future. Networks and Spatial Economics 1, 233–265. https://doi.org/10.1023/A:1012827724856

Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L., 2013. A review of dynamic vehicle routing problems. European Journal of Operational Research 225, 1–11. https://doi.org/10.1016/j.ejor.2012.08.015

Powell, W.B., 1996. A Stochastic Formulation of the Dynamic Assignment Problem, with an Application to Truckload Motor Carriers. Transportation Science 30, 195–219.

PTV, A., 2016a. Vissim 9 User Manual. PTV GROUP, Karlsruhe, Germany.

PTV, A., 2016b. Vissim 9 Introduction to the COM API. PTV GROUP, Karlsruhe, Germany.

Qian, Z. (Sean), Shen, W., Zhang, H.M., 2012. System-optimal dynamic traffic assignment with and without queue spillback: Its path-based formulation and solution via approximate path marginal cost. Transportation Research Part B: Methodological 46, 874–893. https://doi.org/10.1016/j.trb.2012.02.008

Ran, B., Lo, K., Boyce, D.E., 1996. A formulation and solution algorithm for a multi-class dynamic traffic assignment problem. Presented at the Transportation and traffic theory, Lyon, France.

Rosenthal, R.W., 1973. A class of games possessing pure-strategy Nash equilibria. International Journal of Game Theory 2, 65–67.

Ryan, D., 2018. Bellevue prepares for autonomous vehicle transit [WWW Document]. Seattle Transit Blog. URL https://seattletransitblog.com/2018/06/27/bellevue-commutepool/ (accessed 5.15.19).

SAE, 2006. Dedicated Short Range Communications (DSRC) Message Set Dictionary^{TM}. Society of Automotive Engineers. https://doi.org/10.4271/J2735_200612

Schnabel, R.B., 1985. A Modular System of Algorithms for Unconstrained Minimization. ACM Transactions on Mathematical Software 11, 22.

Secomandi, N., 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers & Operations Research 27, 1201–1225. https://doi.org/10.1016/S0305-0548(99)00146-X

Shen, W., Nie, Y., Zhang, H.M., 2007. On Path Marginal Cost Analysis and its Relation to Dynamic System-Optimal Traffic Assignment. Presented at the Transportation and Traffic Theory 2007. Papers Selected for Presentation at ISTTT17, London , England.

Shen, W., Zhang, H.M., 2014. System optimal dynamic traffic assignment: Properties and solution procedures in the case of a many-to-one network. Transportation Research Part B: Methodological 65, 1–17. https://doi.org/10.1016/j.trb.2014.02.002

TRB, 2010. Highway Capacity Manual 2010. Transportation Research Board.

Turpen, A., 2019. Ford steps away from DSRC, embraces 5G cellular for connected vehicles [WWW Document]. New Atlas. URL https://newatlas.com/ford-5g-dsrc-connected-vehicles/57914/ (accessed 5.15.19).

USDOT, 2018. Connected Vehicle Pilot Deployment Program [WWW Document]. United States Department of Transportation. URL https://www.its.dot.gov/pilots/index.htm (accessed 2.26.18).

USDOT, 2016. Connected Vehicle Basics [WWW Document]. United States Department of Transportation. URL https://www.its.dot.gov/cv_basics/cv_basics_20qs.htm (accessed 2.26.18).

USDOT, 2015. CV Pilot Deployment Program - Connected Vehicle Applications [WWW Document]. United States Department of Transportation. URL https://www.its.dot.gov/pilots/cv_pilot_apps.htm (accessed 1.6.19).

Wagner, D., Willhalm, T., 2007. Speed-Up Techniques for Shortest-Path Computations, in: In Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (Stacs'07. Springer, pp. 23–36.

Wardrop, J.G., 1952. Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil Engineers 1, 325–362. https://doi.org/10.1680/ipeds.1952.11259

Webster, F.V., 1958. Traffic signal settings. Road Research Lab Tech Papers /UK/.

Xiao, L., Lo, H.K., 2014. Adaptive Vehicle Navigation With En Route Stochastic Traffic Information. IEEE Transactions on Intelligent Transportation Systems 15, 1900–1912. https://doi.org/10.1109/TITS.2014.2303491

Yang, Q., Koutsopoulos, H.N., 1996. A microscopic traffic simulator for evaluation of dynamic traffic management systems. Transportation Research Part C: Emerging Technologies 4, 113–129.

Zhu, W., Li, Z., Ash, J., Wang, Y., Hua, X., 2017. Capacity Modeling and Control Optimization for a Two-Lane Highway Lane-Closure Work Zone. Journal of Transportation Engineering, Part A: Systems 143, 04017059. https://doi.org/10.1061/JTEPBS.0000078

Ziliaskopoulos, A.K., 2000. A Linear Programming Model for the Single Destination System Optimum Dynamic Traffic Assignment Problem. Transportation Science 34, 37–49. https://doi.org/10.1287/trsc.34.1.37.12281

APPENDIX A

In this appendix we list the link cost function fitting results for all road sections in the simulation network. Figure A.1 shows the numbering of road links (in orange color) in the studied network. The detailed link cost function fitting results are presented in Figure A.2 - Figure A.11. In general, the empirical link cost function can accurately describe the trend of link travel time with traffic volume.

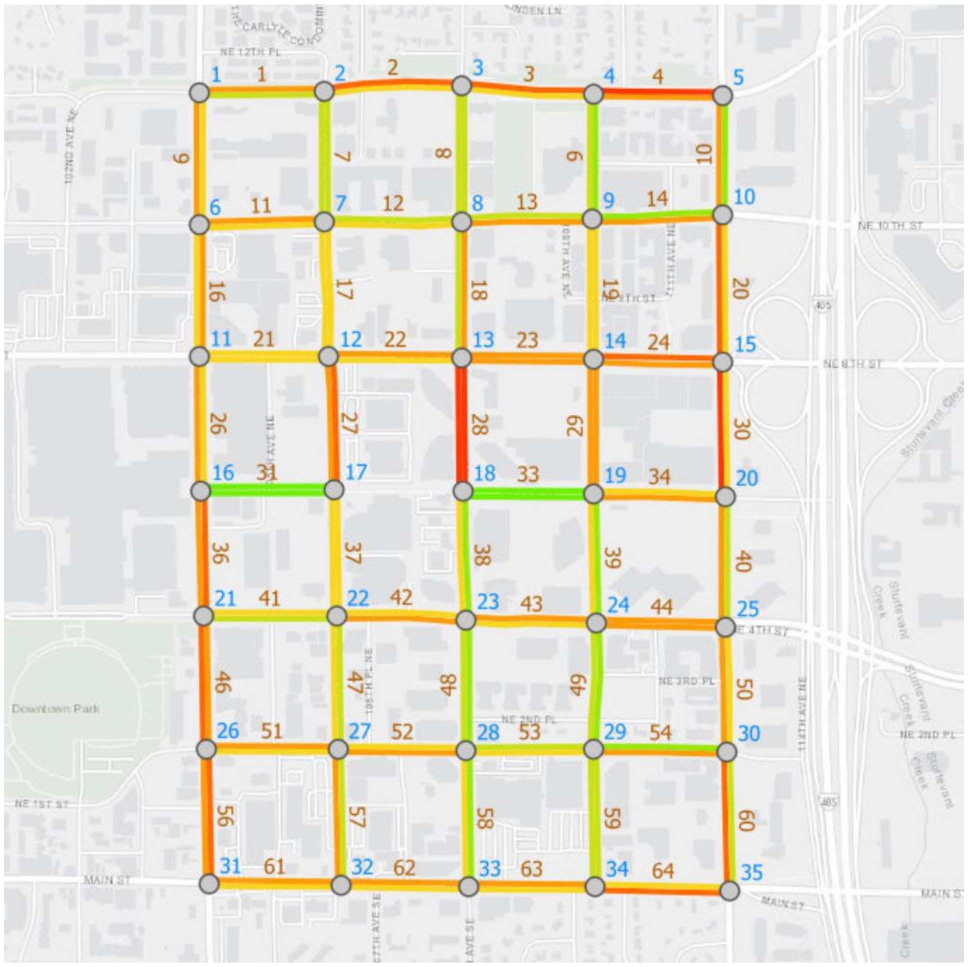


Figure A.1. Numbering of nodes and links in the studied network.

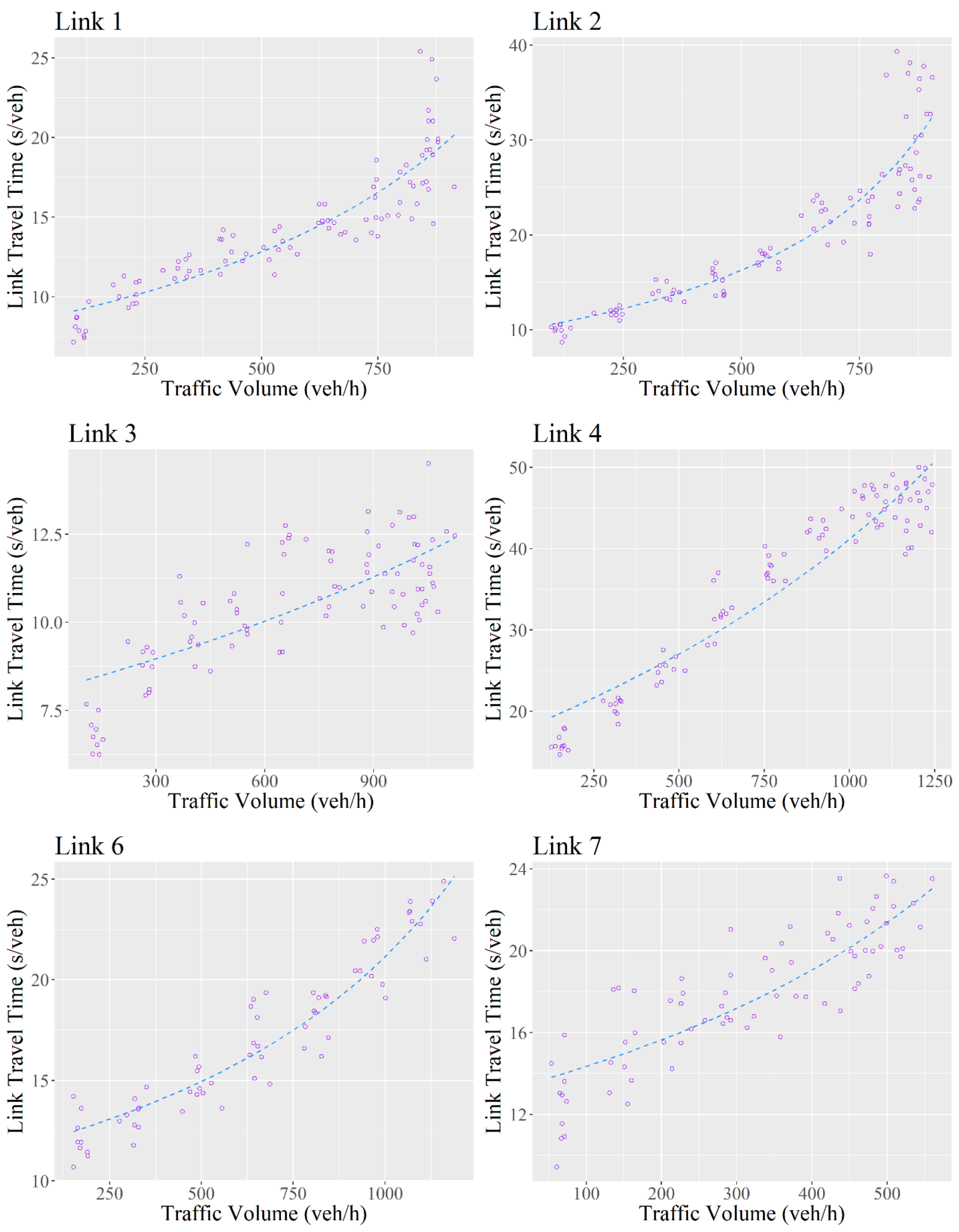


Figure A.2. Link cost function fitting results (link 1-7).

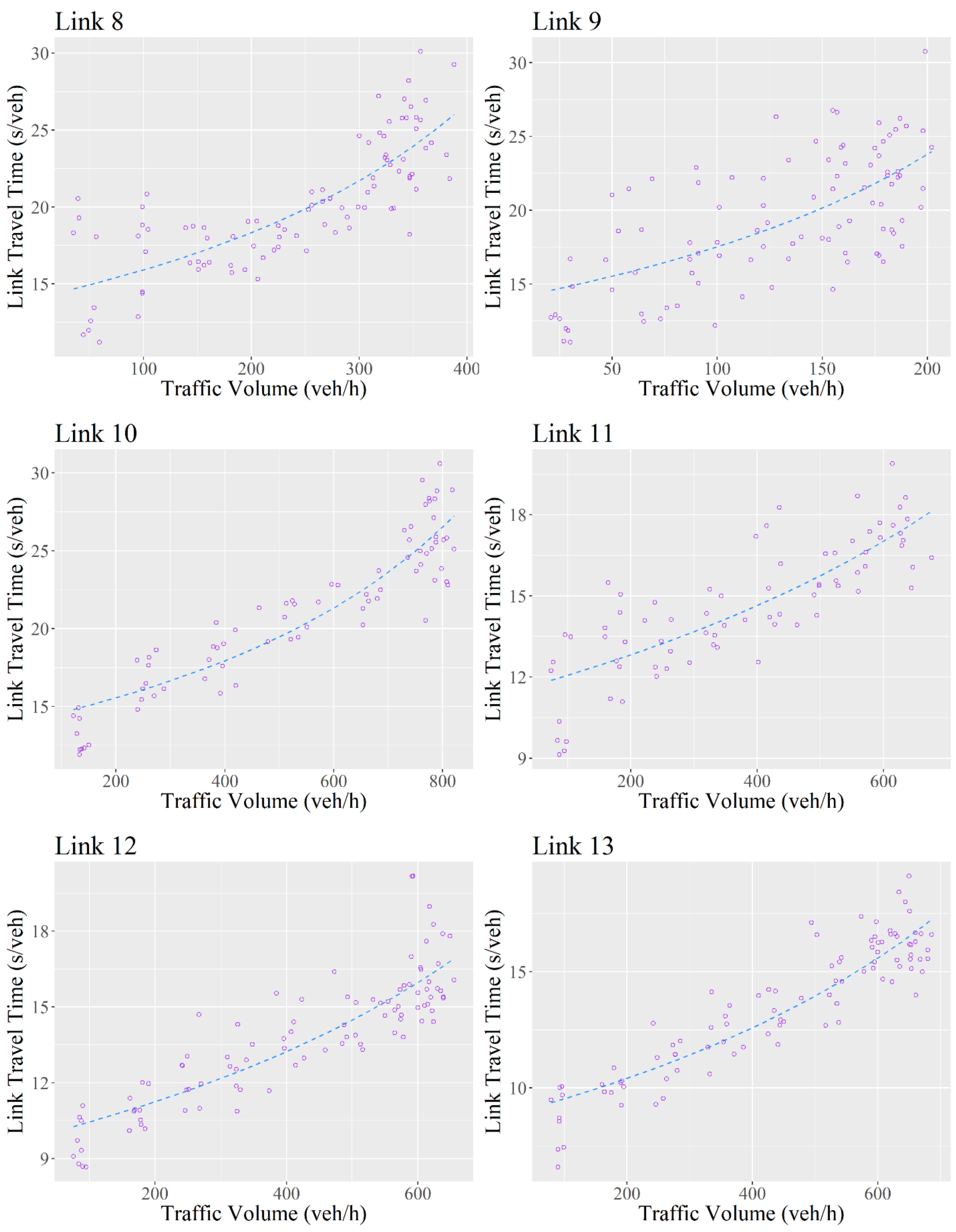


Figure A.3. Link cost function fitting results (link 8-13).

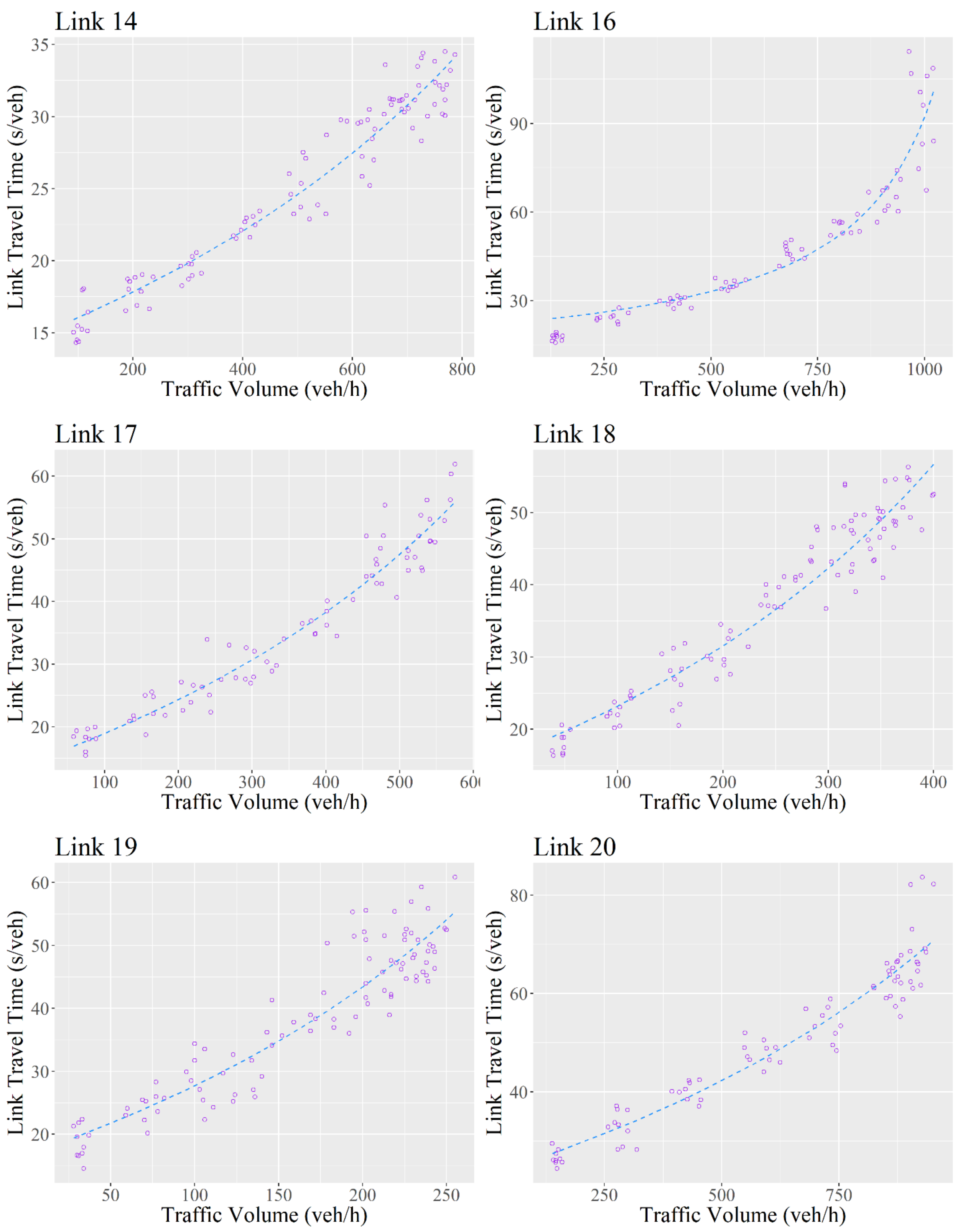
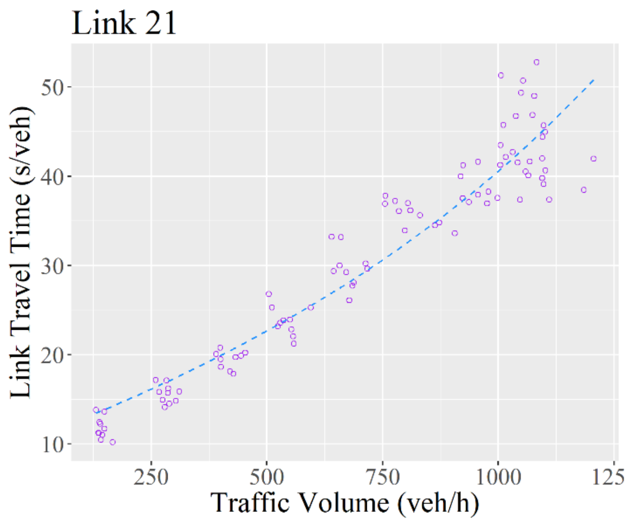
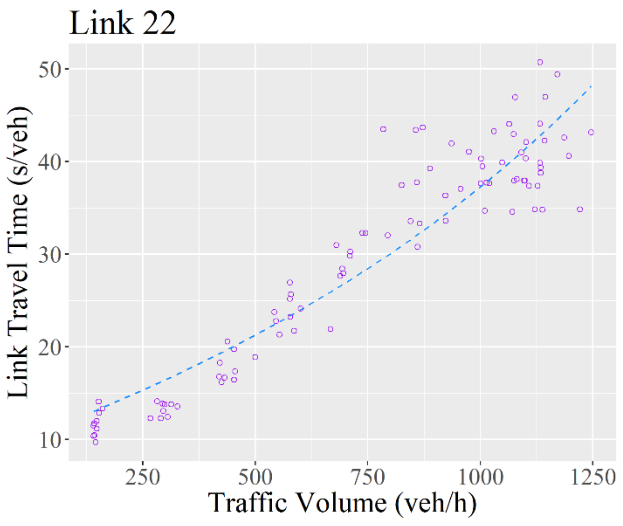
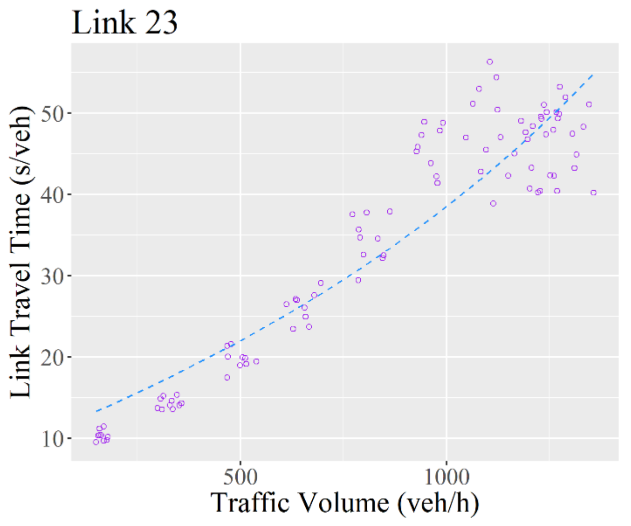
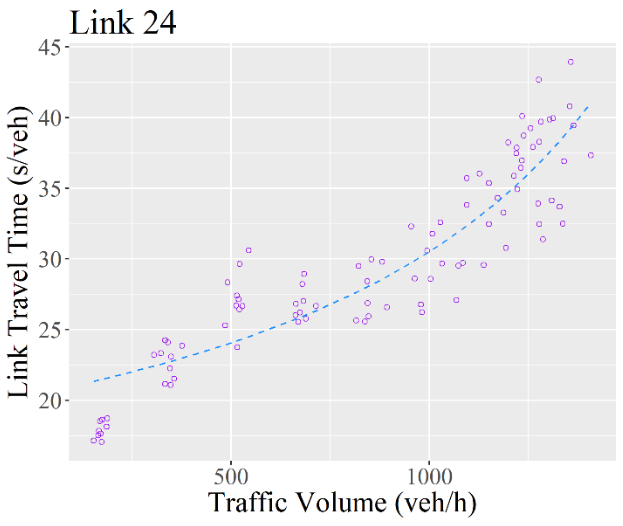


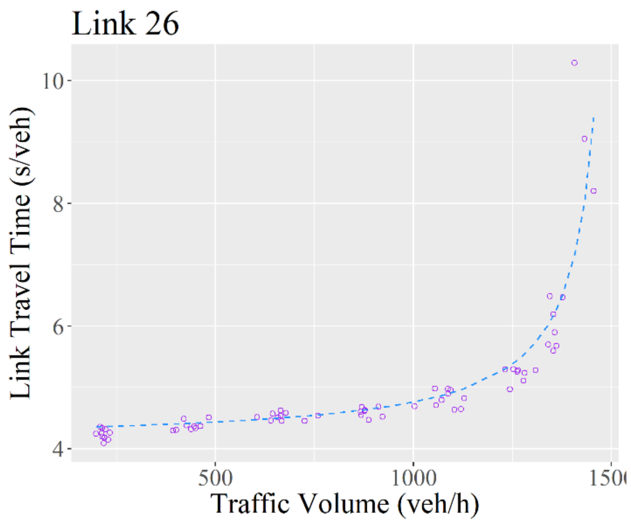
Figure A.4. Link cost function fitting results (link 14-20).











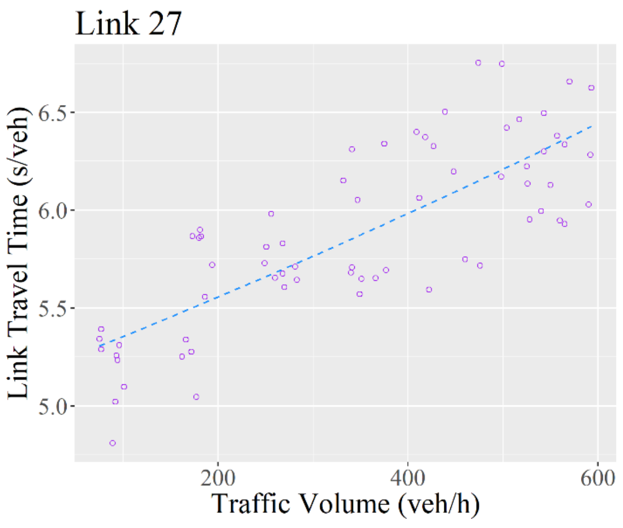


Figure A.5. Link cost function fitting results (link 21-27).

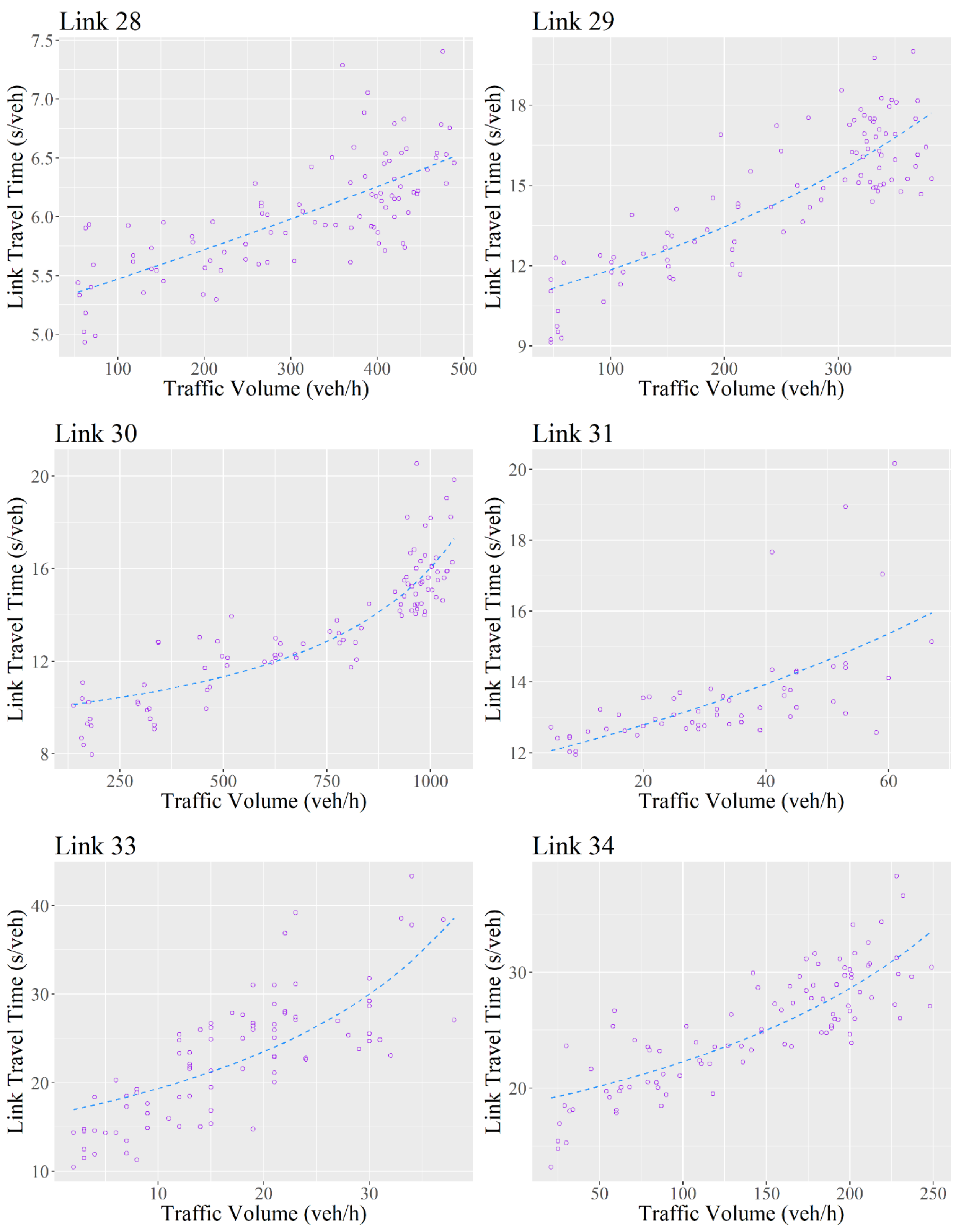


Figure A.6. Link cost function fitting results (link 28-34).

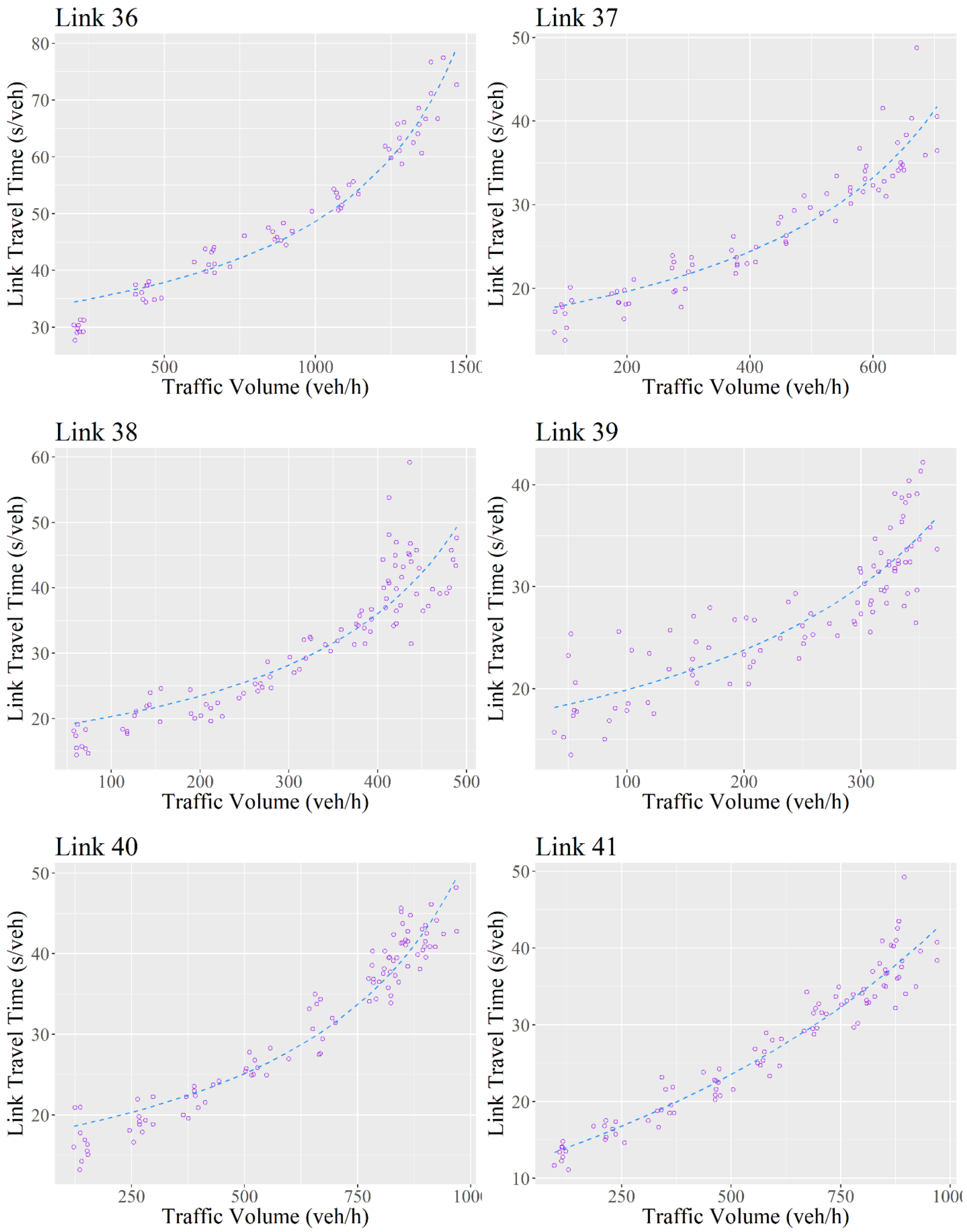


Figure A.7. Link cost function fitting results (link 36-41).

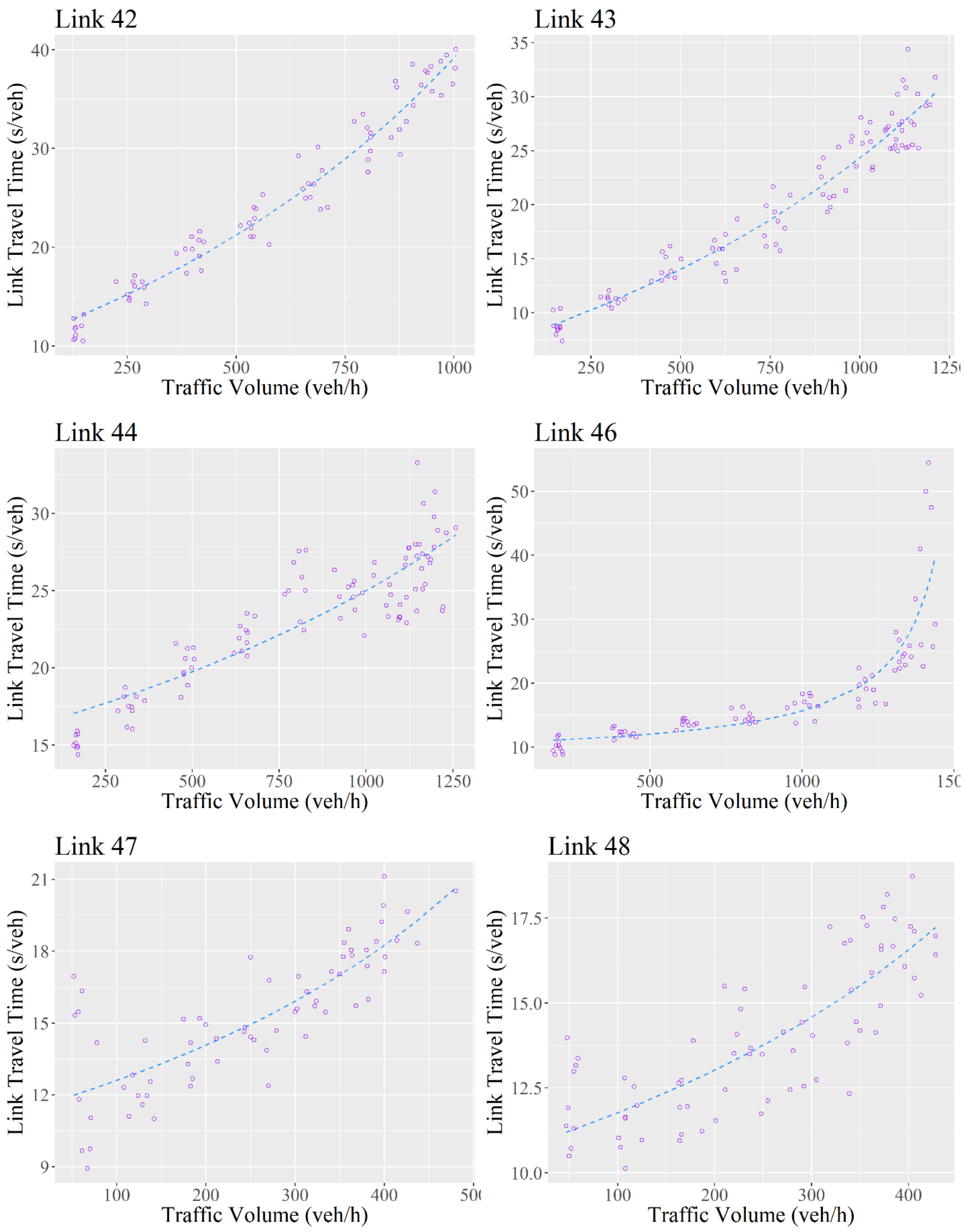


Figure A.8. Link cost function fitting results (link 42-48).

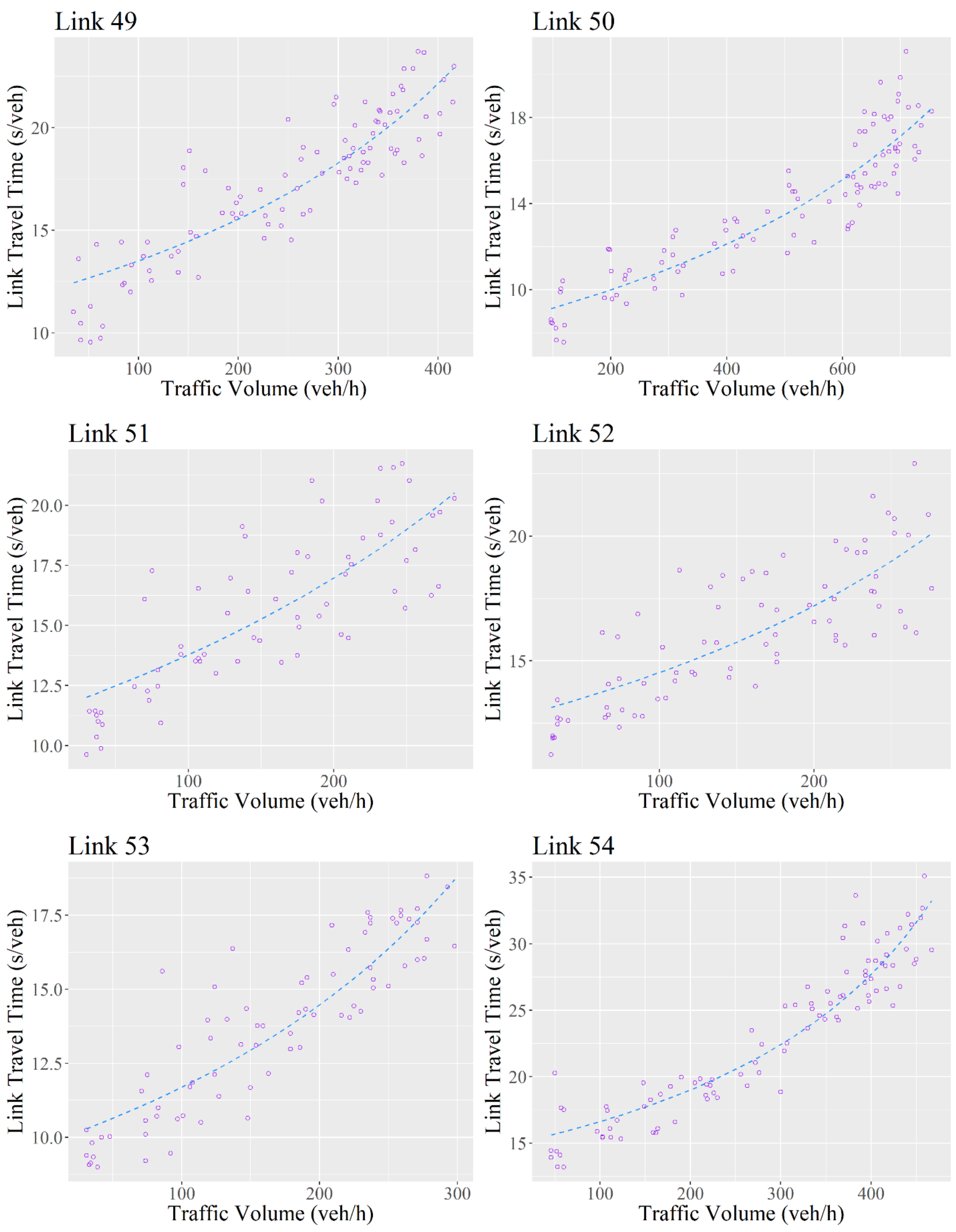


Figure A.9. Link cost function fitting results (link 49-54).

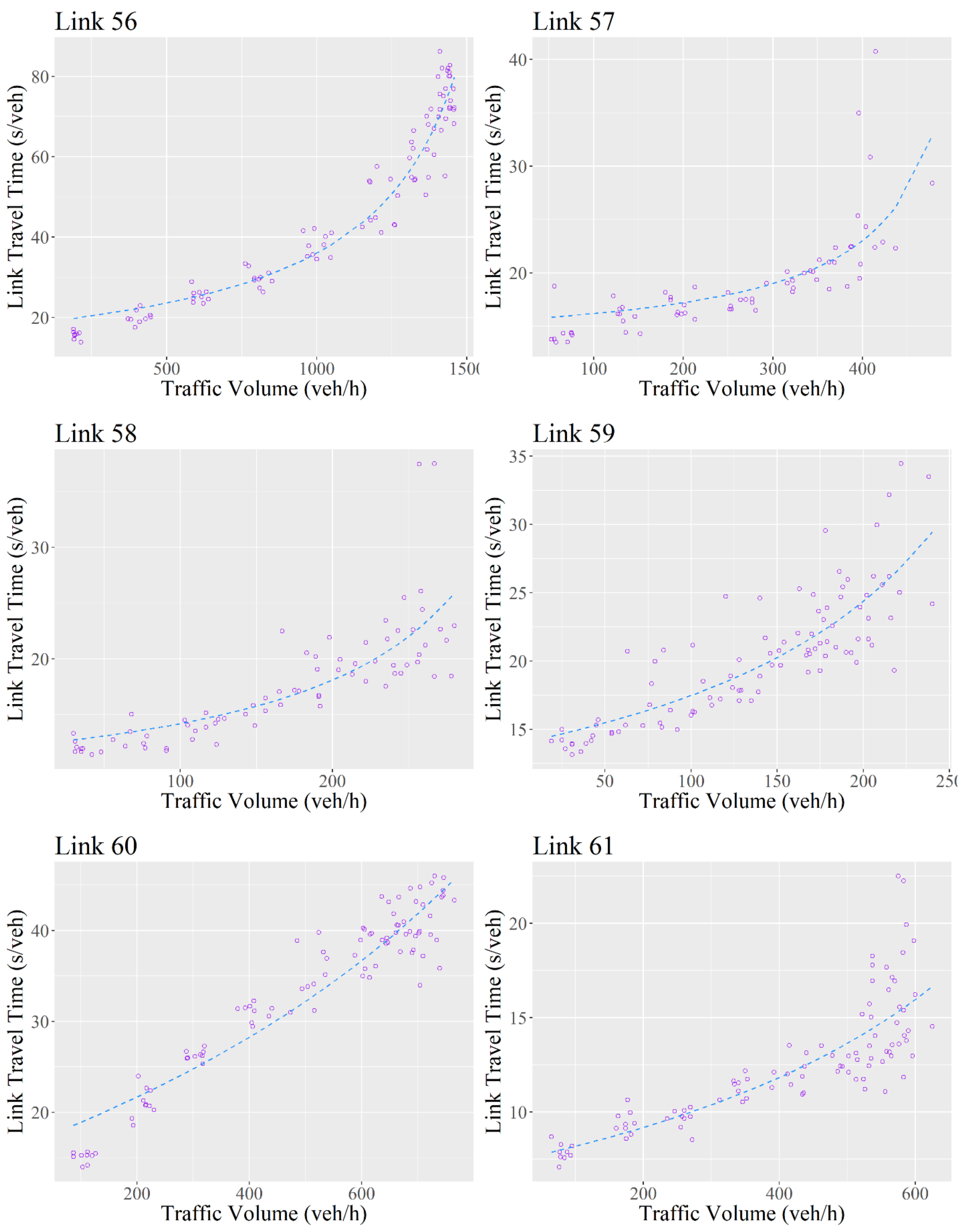


Figure A.10. Link cost function fitting results (link 56-61).

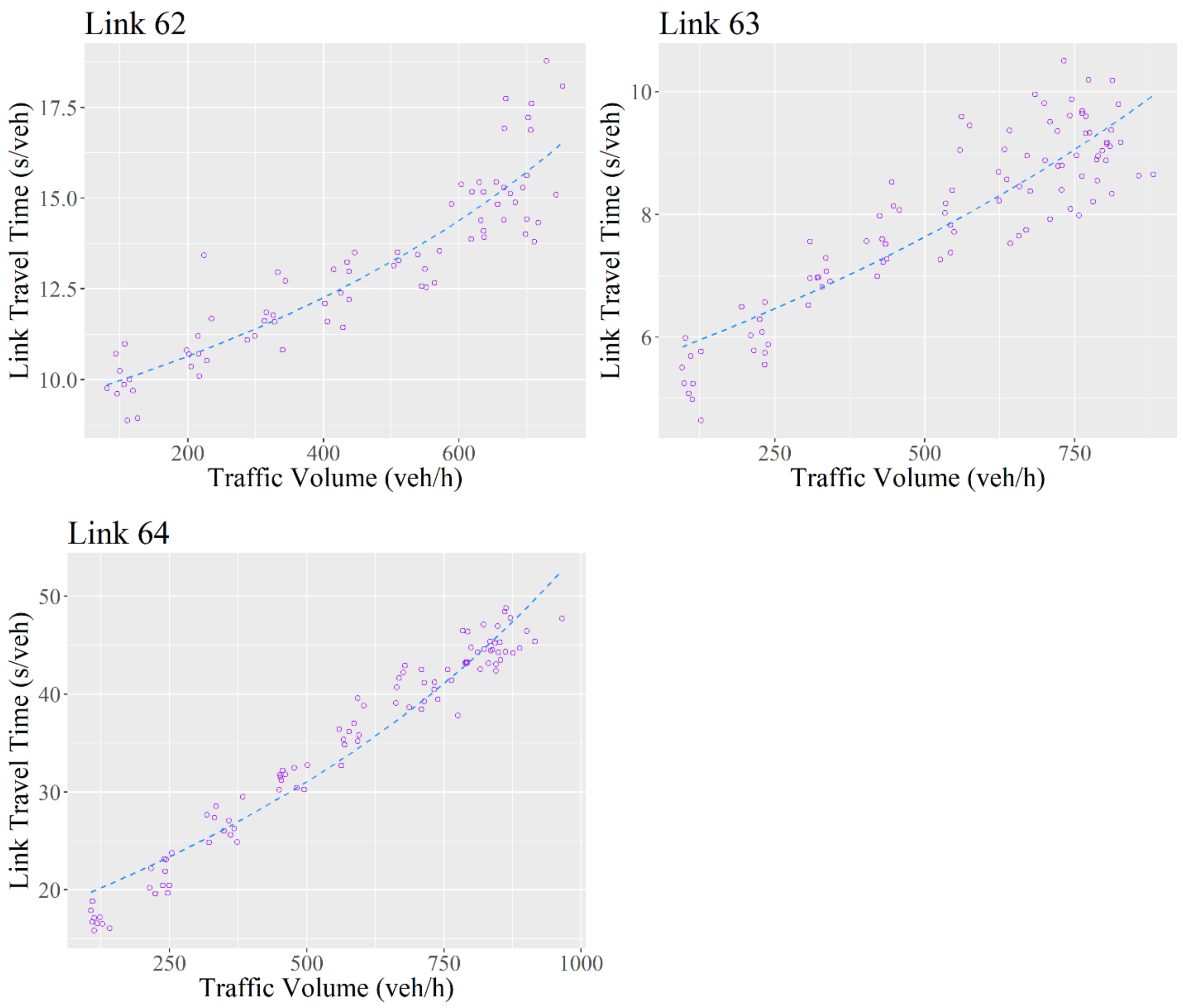


Figure A.11. Link cost function fitting results (link 62-64).

VITA

Wenbo Zhu

EDUCATION

Ph.D. in Civil and Environmental Engineering	04/2015 – present	University of Washington
M.S. in Statistics	09/2017 – 03/2019	University of Washington
M.S. in Civil and Environmental Engineering	09/2013 – 03/2015	University of Washington
B.S. in Civil Engineering	08/2009 – 07/2013	Tsinghua University
B.S. in Economics	08/2010 – 07/2013	Tsinghua University

EXPERIENCE

Graduate Research Assistant	09/2013 – 03/2019	University of Washington
Applied Science Intern	06/2018 – 09/2018	Zillow
Predoctoral Instructor	01/2017 – 03/2017	University of Washington

PUBLICATIONS

Zhu, W., Li, Z., Ash, J., Wang, Y., & Hua, X. (2017). “Capacity modeling and control optimization for two-lane highway lane closure work zones”. ASCE Journal of Transportation Engineering, Part A: Systems, 143(12), 04017059.

Zhu, W., Ash, J., Li, Z., Wang, Y., & Lowry, M. (2015). Applying semi-supervised learning method for cellphone-based travel mode classification. In Smart Cities Conference (ISC2), 2015 IEEE First International (pp. 1-6). IEEE.

Zhou, Z., Zhang, K., **Zhu, W.**, & Wang, Y. (2019). Modeling Lane-Choice Behavior to Optimize Pricing Strategy for HOT Lanes: A Support Vector Regression Approach. Journal of Transportation Engineering, Part A: Systems, 145(4), 04019004.

Zeng, Z., **Zhu, W.**, Ke, R., Ash, J., Wang, Y., Xu, J., & Xu, X. (2017). A generalized nonlinear model-based mixed multinomial logit approach for crash data analysis. Accident Analysis & Prevention, 99, 51-65.

Pu, Z., Li, Z., Ash, J., **Zhu, W.**, & Wang, Y. (2017). Evaluation of spatial heterogeneity in the sensitivity of on-street parking occupancy to price change. Transportation Research Part C: Emerging Technologies, 77, 67-79.

Chen, X., Li, Z., Wang, Y., Tang, J., **Zhu, W.**, Shi, C., & Wu, H. (2018). Anomaly Detection and Cleaning of Highway Elevation Data from Google Earth Using Ensemble Empirical Mode Decomposition. Journal of Transportation Engineering, Part A: Systems, 144(5), 04018015.

SELECTED PRESENTATIONS

Zhu, W., Zeng, Z., Wang, Y., & Pu, Z. (2017). Predicting Incident Duration based on Spatiotemporal Heterogeneous Pattern Recognition (No. 17-06787). Transportation Research Board 96th Annual Meeting. Washington, D.C.

Zhu, W., Wright, B., Li, Z., Wang, Y., & Pu, Z. (2016). Analyzing the impact of grade on fuel consumption for the national interstate highway system (No. 16-6999). Transportation Research Board 95th Annual Meeting. Washington, D.C.

Zhu, W., Li, Z., Ash, J., & Wang, Y. (2016). "Capacity modeling and control optimization for two-lane highway lane closure work zones". International Symposium on Enhancing Highway Performance (ISEHP). Berlin, Germany.

AWARDS

Traffic Bowl Champion	Oregon Institute of Transportation Engineers	11/2015
ITE Student Night Award	Washington State Institute of Transportation Engineers	05/2014
National Scholarship of China	Tsinghua University	11/2012

PROJECTS

Developing a Statistically Valid and Practical Method to Compute Bus and Truck Occupancy Data. Federal Highway Administration. 04/2018 – present.

Reliability Data Guide – Procedures for Using Data in Travel Time Reliability Analyses. Federal Highway Administration. 09/2016 – 03/2018.

Highway Grade Characterization and Operating Efficiency Methods, Tools, and Data Development. Federal Highway Administration. 04/2014 – 06/2016.

Work Zone Capacity Methods for the Highway Capacity Manual. Transportation Research Board. 04/2012 – 09/2014.

Pilot Testing of SHRP 2 Reliability Data and Analytical Products. Transportation Research Board. 02/2013 – 04/2014.

SYNERGISTIC ACTIVITIES

President	University of Washington ITE Student Chapter	10/2015 – 10/2016
President	Chinese Students & Scholars Association at UW	06/2015 – 06/2016
Member	Chinese Overseas Transportation Association	12/2013 – present
Reviewer	ASCE Journal of Transportation Engineering	
Reviewer	Journal of Intelligent Transportation Systems	
Reviewer	TRB Freeway Operations Committee (AHB20)	
Reviewer	IEEE Smart Cities Conference (ISC2)	