DATA-DRIVEN SOLUTIONS TO INTELLIGENT TRANSPORTATION
SYSTEM PROBLEMS CONCERNING TRAFFIC SAFETY AND MOBILITY

BY

SHAN JIANG

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Industrial and Systems Engineering

Written under the direction of

Professor Mohsen A. Jafari

And approved by

_________________________________________________________________

_________________________________________________________________

_________________________________________________________________

_________________________________________________________________

_________________________________________________________________

New Brunswick, New Jersey

May 2021
ABSTRACT OF THE DISSERTATION

Data-Driven Solutions to Intelligent Transportation System Problems Concerning Traffic Safety and Mobility

By Shan Jiang

Dissertation Director:
Mohsen Jafari

Issues surrounding traffic safety and mobility have been at the center of transportation research for many years. Much effort has been made to identify the problematic road features and develop countermeasures to mitigate the crash risk in crash-prone locations. With the emerging connected vehicle technology and smart roadways, the need for smart adaptive traffic signal control and Urban Navigation is more than ever. This dissertation starts with a preliminary study to show how traffic signal timing and drivers’ behavior impact mobility and safety measures at a network of two intersections and the connecting roadway segments. This study integrates the mobility and safety measures into a single two-dimensional Key Performance Indicator (KPI) for intersections. A SUMO simulation model of a real-world roadway network in China is built and run for many scenarios based on a design of experiment aiming at identifying a statistical correlation between the combined measures, the timing of the traffic signals, and drivers’ behaviors. In the world of smart roadways and vehicles, the findings from this work can be used to alert drivers adequately and to optimize signal timings at intersections through closer monitoring of
roadway traffic and drivers’ behavior. This model is also appropriate for road safety audits (RSAs) to prioritize improvements with respect to the combined key performance indicator. The work continues by proposing a novel approach, the Safe Route Mapping (SRM) model that integrates crash-based estimates with conflict risks computed from driver-based data to score the safety of roadways. An advanced Safety Performance Function (SPF) estimates the number of crashes, and a driver-based model computes dynamic conflict risk measures from driver and traffic data. In real-life implementations of the proposed methodology, the driver-based data and traffic data can be collected from vehicles or infrastructure-based data sources, including smartphones. The methodology uses real historical crash data and simulated driver-based data obtained from VISSIM and SSAM to show safety risk heat maps of a roadway and illustrate how these maps change with driver types and traffic volumes. The proposed methodology fills the existing gaps in near real-time dynamic data to designate safe corridors, dispatch law enforcement, and plan safety projects. Drivers can also use road heat maps for situational awareness and trip planning. Followed by the safety study, an Accumulated Exponentially Weighted Waiting Time-based Adaptive Traffic Signal Control (AEWWT-ATSC) model is proposed to calculate roadways' priorities for signal scheduling. As the size of the traffic network grows, it adds significant complexity and challenges to computational efficiencies. A novel Distributed Multi-agent Reinforcement Learning (DMARL) with a graph decomposition approach for large-scale ATSC problems is proposed to solve this issue. The method clusters intersections by the level of connectivity (LoC), defined by the average residual capacities (ARC) between connected intersections, making it possible to train subgraphs instead of the entire network in a synchronized way. The problem is formulated as a Markov Decision
Process (MDP), and the Double Dueling Deep Q Network with Prioritized Experience Replay is utilized to solve it. Under the optimal policy, the agents can select the optimal signal time to minimize the waiting time and queue size. The evaluation shows that the superiority of the AEWWT-ATSC based RL methods in different densities demonstrates the DMARL with graph decomposition approach on a large graph in Manhattan, NYC. The approach is generic and can be extended to various types of use cases.

In the last part of the work, a data-driven optimization approach for dynamic shortest path problems (DDSP) considering traffic safety for urban navigations is developed. The dynamic risk scores and travel times on different road facilities at different times and locations are estimated by the Safe Route Mapping (SRM) methodology and Long Short Term Memory (LSTM) with Autoencoder, respectively, where possible variations in the future are considered. The DDSP is formulated as a mixed-integer linear programming problem under risk constraints to minimize the total travel cost, defined as the weighted sum of distance and travel time. An improved Double Search Algorithm (DSA) with alternative initial-solution algorithms is designed to accommodate various problem scales and improve the efficiency of the DDSP. Moreover, the subgraph and self-adaptive insertion are adopted as acceleration strategies to improve computational efficiency further. Numerical experiments investigate the algorithm's computational performance and compete with the CPLEX solver, a label-setting algorithm, a state-of-the-art algorithm, and commercial navigation software. The result shows a satisfactory trade-off between optimality and computational efficiency with the proposed acceleration strategies. The real life implementation shows that the algorithm can provide the same quality of routing
decisions on the shortest and fastest path as the Google Map, which is promising for Urban Navigation.
ACKNOWLEDGEMENTS

The completion of this dissertation would not have been possible without the support of my family, professors, and friends. Thanks so much to everyone who has helped me along my long journey. There are far too many to name, but here are some people whom I would like to mention especially. First and foremost, I would like to express my appreciation and gratitude to my advisor Prof. Mohsen Jafari, who has been a tremendous mentor. I want to thank him for encouraging my research and allowing me to grow as a research scientist. His advice on my research and career has been invaluable. I would also like to sincerely thank my Ph.D. committee, Prof. Gursoy, Prof. Myong K. Jeong, and Prof. Jin, for taking the time to review my dissertation and provide constructive suggestions. I would also like to thank Dr. Nasim Arbazadeh for the support and advice during my internship at Johnson & Johnson. I gratefully acknowledge partial financial support from Qatar National Research Fund (QNRF) under Grant no. NPRP8-910-2-387. Finally, special thanks to my father Zhibin Jiang, mother Xiaodi Diao, my wife Jing Yang, and my daughter Olivia Jiang. My parents are extremely educated and high achievers in their professional and personal lives, inspiring me to overcome hard times in my studies. My father always encouraged me with constructive suggestions when I faced challenges. My mother gave up her relaxed life in Shanghai, China, to help me take care of my baby daughter. My wife Jing Yang accompanied me during my Ph.D. study, enduring loneliness and hardships with me. The birth of my lovely daughter Olivia motivates me to be a good father and a responsible person in my career and life. Words cannot express how grateful I am to my family for their encouragement, patience, and unconditional support. This journey would not have been possible without your support.
# TABLE OF CONTENTS

ABSTRACT OF THE DISSERTATION ......................................................................................... ii

ACKNOWLEDGEMENTS ......................................................................................................... vi

TABLE OF CONTENTS ......................................................................................................... vii

1. INTRODUCTION ............................................................................................................... 1
   1.1 The Objective of the Work .......................................................................................... 1
   1.2 Synopsis of Contribution .......................................................................................... 3
      1.2.1 Developing a Two-Dimensional Key Performance Indicator of Safety and Mobility for Intersections ................................................................. 3
      1.2.2 Safe Route Mapping of Roadways Using Multiple Sourced Data .............. 4
      1.2.3 A Distributed Multi-Agent Reinforcement Learning with Graph Decomposition Approach for Large-scale Adaptive Traffic Signal Control ............ 5
      1.2.4 Data-driven optimization for dynamic shortest path problem with time-varying travel time and traffic safety for urban navigation ........................................ 6

2. DEVELOPING A TWO-DIMENSIONAL KEY PERFORMANCE INDICATOR OF SAFETY AND MOBILITY FOR INTERSECTIONS ................................................. 8
   2.1 Introduction .................................................................................................................. 8
   2.2 Literature Review ...................................................................................................... 9
   2.3 Modeling and Problem Formulation ......................................................................... 10
   2.4 Numerical Example .................................................................................................. 13
   2.5 Conclusion ............................................................................................................... 18

3. SAFE ROUTE MAPPING OF ROADWAYS USING MULTIPLE SOURCED DATA ................................................................. 20
3.1 Introduction ...................................................................................................................... 20
3.2 Literature Review ............................................................................................................. 21
3.3 Description of SRM Methodology .................................................................................. 24
  3.3.1 Crash Prediction Model ............................................................................................... 26
  3.3.2 Driver-based Model .................................................................................................... 28
  3.3.3 Integrated Risk Model ............................................................................................... 30
3.4 Illustrative Example ........................................................................................................ 33
  3.4.1 Prediction of Crash Count ......................................................................................... 33
  3.4.2 Risk Profile of Individual Drivers ........................................................................... 36
  3.4.3 Integration of Crash Data and Conflict Data .......................................................... 42
3.5 Conclusion ..................................................................................................................... 45
3.6 Appendix ....................................................................................................................... 46

4. A DISTRIBUTED MULTI-AGENT REINFORCEMENT LEARNING WITH
GRAPH DECOMPOSITION APPROACH FOR LARGE-SCALE ADAPTIVE
TRAFFIC SIGNAL CONTROL ......................................................................................... 49
  4.1 Introduction .................................................................................................................... 49
  4.2 Literature Review .......................................................................................................... 52
  4.3 Problem Formulation and Modeling ........................................................................... 55
    4.3.1 AEWWT-ATSC Model ........................................................................................... 56
    4.3.2 Formulation of Signal Scheduling ......................................................................... 59
    4.3.3 DQN Models and Algorithm ............................................................................... 62
    4.3.4 Distributed Multi-agent Reinforcement Learning with Graph Decomposition Approach .......................................................... 66
4.4 Simulation Experiments .................................................................................. 71

4.4.1 Comparison of Different Control Policies for A Single Intersection ........ 71

4.4.2 Comparison of Competitive and Cooperative MARL for Two Intersections ........................................................................................................ 75

4.4.3 Case Study of Using DMARL for A Large Real Network ................. 76

4.5 Conclusion ........................................................................................................ 78

5. DATA-DRIVEN OPTIMIZATION FOR DYNAMIC SHORTEST PATH PROBLEM WITH TIME-VARYING TRAVEL TIME AND TRAFFIC SAFETY FOR URBAN NAVIGATION ........................................................................................................ 80

5.1 Introduction ...................................................................................................... 80

5.2 Literature Review .............................................................................................. 83

5.3 Methodology and Problem Statement ............................................................ 87

5.3.1 Risk Estimation .......................................................................................... 88

5.3.2 Travel Time Prediction ................................................................................. 88

5.3.3 Shortest Path Formulation .......................................................................... 89

5.4 DDSP Solution approach .................................................................................. 92

5.4.1 Initial Solution ............................................................................................ 93

5.4.2 Penalized Objective Function ..................................................................... 96

5.4.3 Neighborhood structures ............................................................................ 97

5.4.4 Tabu List and Aspiration criterion .............................................................. 97

5.4.5 Acceleration Strategies ............................................................................... 97

5.5 Numerical Study ............................................................................................... 99

5.5.1 Experimental Data ...................................................................................... 99
5.5.2 Comparison among Initial-Solution algorithms................................. 100
5.5.3 Comparison with other solution approaches.................................... 103
5.5.4 Verification of the Subgraph................................................................ 107
5.5.5 Real Network Demonstration ............................................................. 108

5.6 Conclusion........................................................................................................ 119

Reference ............................................................................................................... 120
1. INTRODUCTION

1.1 The Objective of the Work

This dissertation intends to develop data-driven solutions for traffic safety and mobility applications. Integration of data from multiple sources, Operations Research, and Machine Learning approaches are the foundations of the methodologies presented in this Ph.D. dissertation. This dissertation starts with a discussion and numerical experimentation relating to safety and mobility. The objective is to emphasize the strong correlation between safety and mobility issues. This study follows traffic safety with a novel methodology that takes advantage of historical crash data along with driver data from real-time onboard or infrastructure devices. The study is interested in driver behavior that can potentially lead to a near-miss condition. Unlike crashes, near-miss events are in abundance and observable through appropriate instrumentation. Such a hybrid model provides safety performance measures that are far powerful than those merely obtained from rare crash events. These measures are also the basis for calculating risk profiles for drivers used while driving or can be aggregated to network-level measures to be used by law enforcement or network operators. Similar ideas with much less rigorous mathematics have already been in use by auto insurance companies. These risk measures’ dynamic nature is very appealing; a roadway segment can be assigned different risk scores depending on the time of the day, weather conditions, or even the type of drivers and vehicles. The term Safe Route Mapping (SRM) is tossed, and risk maps are built available to drivers and authorities with different resolution levels. Risk heat maps can help drivers plan their trips with traffic safety in mind (safety-based navigation) and help network
owners and authorities plan safety solutions and dispatching law enforcement resources in more real-time and proactively.

Unsafe traffic conditions, including accidents and near misses, are like random shocks that temporarily degrade mobility, with more permanent fatigues resulting from frequent recurrence of such events. Economically speaking, traffic delays caused by accidents and conflicts are penalties (value of lost time) that are imposed on drivers and network owners alike. In general terms, one can define a penalty function with two basic terms: (i) penalty due to accidents and traffic conflicts, (ii) penalty due to suboptimal congestion management techniques. Since these two terms are correlated, there could be double counting in the overall penalty calculation unless one can fully separate the consequential delays due to accidents/conflicts from those caused by traffic controls. This is quite possible in a simulation environment and also in a real-world where roadways and vehicles are instrumented and connected.

The primary focus on congestion management will be geared toward traffic signal controls and urban navigations. A novel Distributed Multi-agent Reinforcement Learning (DMARL) with a graph decomposition approach for large-scale ATSC problems is developed. The priority of a direction that is going to receive the next green signal is determined by an Accumulated Exponentially Weighted Waiting Time-based Adaptive Traffic Signal Control (AEWWT-ATSC) model. This dissertation also attempts to tackle the dynamic shortest path problem (DDSP) considering traffic safety and mobility for urban navigations. The dynamic risk scores and travel times on different road facilities at different times and locations are estimated by the Safe Route Mapping (SRM) methodology and Long Short Term Memory (LSTM) with Autoencoder, respectively,
where possible variations in the future are considered. The DDSP is formulated as a mixed-integer linear programming problem under risk constraints to minimize the total travel cost, which is defined as the weighted sum of distance and travel time. An improved Double Search Algorithm (DSA) with alternative initial-solution algorithms is designed to accommodate various problem scales and improve the efficiency of the DDSP. Moreover, the subgraph and self-adaptive insertion are adopted as acceleration strategies to improve computational efficiency further.

1.2 Synopsis of Contribution

1.2.1 Developing a Two-Dimensional Key Performance Indicator of Safety and Mobility for Intersections

This preliminary study develops an understanding of how traffic signal timing and drivers’ behavior impact mobility and safety measures at a network of two intersections and the connecting roadway segments. The mobility and safety measures are integrated into a single two-dimensional Key Performance Indicator (KPI) for intersections. A SUMO simulation model of a real-world roadway network in China is built and run for many scenarios based on a design of experiment, aiming to identify a statistical correlation between the combined measures, namely, the timing of the traffic signals and drivers’ behaviors. In the world of smart roadways and vehicles, the findings of this work can be used to alert drivers adequately and to optimize signal timings at intersections through closer monitoring of roadway traffic and drivers’ behavior. This model is also appropriate for road safety audits (RSAs) to prioritize improvements concerning the combined key performance indicator.
1.2.2 Safe Route Mapping of Roadways Using Multiple Sourced Data

The proposed Safe Route Mapping (SRM) model integrates crash-based estimates with conflict risks computed from driver-based data to roadways' risk score safety. For real-life implementations, the driver and road data are obtained from vehicles or infrastructure-based data sources, including smartphones. Here, the methodology using real historical crash data and driver-based data obtained from VISSIM / SSAM and real sources, including Naturalistic Driver Behavior Study conducted by the US-DoT, are demonstrated. This work is partially supported by China's project and a joint project with Qatar University funded by the Qatar National Research Foundation (QNRF). An APP is developed and is now being used by the Qatar University team to collect real data for model validation.

The Safety Performance Function (SPF) model estimates crash counts for a given roadway segment or facility using Negative Binomial Regression on historical crash data. The driver-based model predicts individual drivers' conflict probabilities by considering their behavioral characteristics and roadway and traffic conditions. A Neural Network (NN) model is used to estimate risks from driver-based data. A fuzzy logic-based integration method is also employed to fuse the data from SPF and Driver-based models to obtain risk scores and generate risk heatmaps. This work's contributions are several folds: (i) An advanced SPF to estimate the number of crashes. The proposed SPF uses many independent variables and relies less on correction factors. It accounts for over-dispersion and hence assumes a Negative Binomial model of crash counts. (ii) A methodology to create risk profiles of individual drivers with location and time attributes. A risk profile is a collection of risk measures that are sampled from a driver-based Machine Learning (ML) model. The model takes into account key driving safety behavioral attributes (e.g.,
aggressiveness). (iii) Aggregation of risk profiles into statistical risk measures for a roadway segment and a given period of time. These statistical measures change with driver mix and traffic flow. (iv) Risk heat mapping of roadway segment or facilities using a hybrid measure composed of SPF estimates and aggregated statistical measures from drivers. A fuzzy logic-based approach that translates the number of crashes and the probability of conflicts into fuzzy measures is built to determine a common denominator.

1.2.3 A Distributed Multi-Agent Reinforcement Learning with Graph Decomposition Approach for Large-scale Adaptive Traffic Signal Control

With the emerging connected vehicle technology and smart roadways, the need for intelligent adaptive traffic signal controls is more than ever before. There are already many adaptive solutions to the traffic signal scheduling problem in the literature. The purpose is not to incrementally add to an already rich body of literature on adaptive signal controls. Methodologically speaking, the Accumulated Exponentially Weighted Waiting Time-based Adaptive Traffic Signal Control (AEWWT-ATSC) model takes the exponentially weighted waiting time of each vehicle at the intersection to generate priorities as the control references for signal sequencing and timing. In a dynamic traffic study where signal times are not fixed, each intersection controller is assumed to be an intelligent agent seeking to minimize the traffic delay and the number of vehicles in the queue by selecting a proper green signal duration in each cycle. The control problem is formulated as a Markov Decision Process (MDP), and Double Dueling DQN with Prioritized Experience Replay is utilized to find the optimal solution. A well-trained agent adopts optimal control policies for signal times, according to the current traffic patterns. Under the optimal policy, the traffic delay and the queue size are minimized. The AEWWT-ATSC model's performance
is compared with other control policies in different traffic densities to show the superiority of the method. It is necessary for a network with more than one intersection to adopt Multi-agent Reinforcement Learning (MARL). In a network with two intersections, competitive MARL and fully observable cooperative MARL are compared. The results indicate that fully observable cooperative MARL outperforms the competitive MARL. For large graphs, it is essential to apply the decomposition method and distributed MARL (DMARL) approach to reduce computational requirements. The approach clusters intersections into subgraphs and trains each subgraph in a synchronized way. This study suggests that the approach is generic and can be used for various types of intersections. In the absence of real traffic data, VISSIM simulation is developed for validation purposes.

1.2.4 Data-driven optimization for dynamic shortest path problem with time-varying travel time and traffic safety for urban navigation

This Chapter is the integration of traffic mobility, safety, and route planning. A data-driven optimization approach is developed for dynamic shortest path problems (DDSP) considering traffic safety for urban navigations. The dynamic risk scores and travel times on different road facilities at different times and locations are estimated by the aforementioned Safe Route Mapping (SRM) methodology and Long Short Term Memory (LSTM) with Autoencoder, respectively, where possible variations in the future are considered. The DDSP is formulated as a mixed-integer linear programming problem under risk constraints to minimize the total travel cost, which is defined as the weighted sum of distance and travel time. An improved Double Search Algorithm (DSA) with alternative initial-solution algorithms is designed to accommodate various problem scales.
Moreover, the subgraph and self-adaptive insertion are adopted as acceleration strategies to improve the efficiency of the DDSP. Numerical experiments investigate the algorithm's computational performance and are compared to the CPLEX solver, a label-setting algorithm, a state-of-the-art algorithm, and commercial navigation software. The result shows a satisfactory trade-off between optimality and computational efficiency with the proposed acceleration strategies. The implementation of the method in a real network shows that the algorithm can provide the same quality of routing decisions on the shortest and fastest path as the Google Map, which is promising for Urban Navigation.
2. DEVELOPING A TWO-DIMENSIONAL KEY PERFORMANCE INDICATOR OF SAFETY AND MOBILITY FOR INTERSECTIONS

2.1 Introduction

In 2012, urban Americans traveled 5.5 billion hours more and purchased an extra 2.9 billion gallons of fuel due to traffic congestions. Today, commuters spent about 54 hours a year in traffic, resulting in a total waste of 3.3 billion gallons of gas [1]. The cost of traffic congestion is immense. For instance, in 2013, traffic congestion costs Americans $124 billion, and this number will increase to $186 billion in 2030, a 50% increase over 2013 [2]. According to the National Transportation Operations Coalition, poor signal timing accounts for 5 to 10 percent of all traffic delays [3]. Moreover, according to the Federal Highway Administration (FHWA), more than 50 percent of the combined fatal and injury crashes occur at intersections [4]. While there is no singular cause of vehicle crashes, a combination of human, roadway, and vehicle factors contribute to these occurrences. At the same time, human errors (e.g., risky driving behavior) are associated with nearly 90 percent of light-vehicle crashes [5]. Given these sober statistics, the effects of traffic signal timing and drivers’ behavior and their interactions on mobility and safety need further investigations.

This chapter intends to develop an understanding of how traffic signal timing and drivers’ behavior impact mobility and safety measures at a network of two intersections and the connecting roadway segments. A SUMO simulation model is built to integrate mobility and safety measures into a single two-dimensional Key Performance Indicator (KPI).
2.2 Literature Review

Over the past years, there have been a considerable number of studies exploring the effects of intersection traffic signal timing on traffic mobility and safety. The Federal Highway Administration (FWHA) sponsored a project to develop algorithms that balance safety and capacity under adaptive signal control schemes. The results demonstrated that the cycle length significantly contributed to the frequency of crashes at intersections, and adopting a longer cycle length resulted in reduced traffic conflicts regardless of their types [6]. Robles [7] developed a multi-objective computational framework in identifying signal control strategies to improve traffic mobility and environmental impacts. Yang et al. [8] established an effective procedure to enhance intersection signal timing by minimizing total delay for both vehicles and pedestrians. Taking advantage of a random-parameter negative binomial model, Agbelie and Roshandeh [9] demonstrated that an increase in the number of traffic signal phases would increase the probability of crashes at urban intersections. The results from a study conducted by Chin and Quddus [10] also revealed that the number of traffic signal phases contributed significantly to the traffic crashes. Jolovic et al. [11] explored the association between signal timing parameters and intersection-related crashes. Feng [12] evaluated the safety impacts of signal timing optimization on urban intersections with red-light running photo enforcement. Stevanovic et al. [13] developed a novel approach for optimizing signal timings to mitigate the surrogate safety measures and reduce the risks of real-world crashes. In another study, Pant et al. [14] evaluated the impacts of green extension on crash reductions at closely-spaced high-speed intersections.
As evident from the above works, several works address the impacts of signal timing and driver behavior on mobility and safety. But no previous study that attempts to integrate mobility and safety measures into a single multi-dimensional Key Performance Indicator (KPI) for intersections. In this paper, a real-world roadway network with two intersections is built by SUMO in China to conduct scenario-based analysis using a combination of different traffic signal timings and drivers’ behavior (i.e., aggressive and conservative).

The objective is to identify the conflicting relationship between safety and mobility under various circumstances. The results obtained from this study will serve a more significant objective, namely, scoring roadways using the proposed KPI. To be specific, it is anticipated to develop a tool, which can extract aggregated information from multiple sources (e.g., floating vehicles and crowdsourcing data) and determine the safety-mobility KPI for different roadway facilities (e.g., segments, intersections, roundabouts, and ramps).

This study also provides valuable information for researchers, policymakers, engineers, and law enforcement to improve the mobility, safety, efficiency, and reliability of transportation systems.

2.3 Modeling and Problem Formulation

This section uses a case study to demonstrate the methodology. A network of two signalized intersections (i.e., Huanghan-Kexue and Huanghan-Tianzhi) in Hefei, China is considered (see Figure 2.1).
The problem of interest is to evaluate the impacts of traffic signal timing and driver’s behavior on safety and mobility over the two intersections. It is noted that the two intersections use the same fixed-time signal control with the same traffic signal timing plans, as shown in Table 2.1.

**Table 2.1 Traffic Signal Timing Plans of Study Intersections**

<table>
<thead>
<tr>
<th>Plan No.</th>
<th>Cycle Length (sec.)</th>
<th>Active Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>106</td>
<td>0:00-6:00; 21:00-24:00</td>
</tr>
<tr>
<td>2</td>
<td>136</td>
<td>6:00-7:30; 9:00-17:00; 19:00-21:00</td>
</tr>
<tr>
<td>3</td>
<td>194</td>
<td>7:30-9:00</td>
</tr>
<tr>
<td>4</td>
<td>138</td>
<td>17:00-19:00</td>
</tr>
</tbody>
</table>

This traffic simulation suite allows modeling of intermodal traffic systems, including road vehicles, public transport, and pedestrians. It should be noted that there is a wealth of supporting tools included with SUMO that can handle tasks such as route finding, visualization, network import, and emission calculation. SUMO can be enhanced with custom modules and provides various application program interfaces (APIs) to control the simulation remotely. Figure 2.2 shows the simulation model in the SUMO environment.
In SUMO simulation, the number of phases, duration, and state of each phase is explicitly defined. Eighty-six thousand four hundred simulation steps are run (i.e., 24 hours) in SUMO. Generally, SUMO aggregates all simulation steps and provides final outputs for each simulation run. Matlab and VBA tools are used to automate batch runs of models with changing parameter values. To evaluate the system’s performance, the mobility indicator - Throughput Rate (TR), is defined to test the traffic flow mobility in discrete time intervals, as specified in Equation 1.1.

\[
TR = \frac{\text{Number of Exiting Vehicles}}{\text{Number of Entering Vehicles}}
\]  

(2.1)

It is noted that a higher TR indicates good mobility of the transportation network. Multiple detectors are put on intersections to measure such capacity of intersections in the simulation model. The detectors are placed at 100 meters (~330 ft.) distance from the traffic signal (yellow arrows in Figure 2.3) to record the total number of vehicles that enter intersections. Moreover, several other detectors also were used 2 meters (~7 ft.) downstream of the traffic signals (red arrows in Figure 2.3) to count the number of exiting vehicles from the intersections. Average TRs are calculated in different time intervals (e.g., every hour) to measure mobility.
In SUMO simulation, conflicts are used as a safety measure. Traffic conflicts include, but are not limited to, collision and emergent stop due to change of traffic light signal. The term Conflict Ratio (CR) is tossed for measuring safety, as defined in Equation 1.2.

\[
CR = \frac{Number \, of \, conflicts}{Number \, of \, Entering \, Vehicles}
\] (2.2)

Either TR or CR is a straightforward indicator of traffic performance. The next section demonstrates how to integrate these two into a single two-dimensional performance measure for drivers.

### 2.4 Numerical Example

The first step is to design several experiments with five critical parameters related to driver behavior (i.e., acceleration, deceleration, minGap, tau, and sigma), each of which takes either low or high value (see Table 2.2).

**Table 2.2 Selected Parameters in the Design of Experiment (SUMO, 2017)**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Low</th>
<th>High</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accel (m/s^2)</td>
<td>2.1</td>
<td>5.6</td>
<td>The acceleration ability of vehicles of this type</td>
</tr>
<tr>
<td>decel (m/s^2)</td>
<td>3.1</td>
<td>9.3</td>
<td>The deceleration ability of vehicles of this type</td>
</tr>
<tr>
<td>minGap (m)</td>
<td>1.0</td>
<td>5.0</td>
<td>Empty space after leader</td>
</tr>
<tr>
<td>tau</td>
<td>0.5</td>
<td>2.5</td>
<td>The driver's desired (minimum) time headway</td>
</tr>
<tr>
<td>Sigma (m)</td>
<td>0.0</td>
<td>1.0</td>
<td>The driver imperfection, between 0 and 1</td>
</tr>
</tbody>
</table>
The combination of these parameters results in 32 different types of drivers for studying how these parameters impact Conflict Ratio and Throughput Rate. Notably, for the purpose of comparison, the original traffic light signal timing and traffic volume data are used in these experiments. Figure 2.4 scatterplots Conflict Ratio vs. Average Throughput Rate of intersections of each type of driver.

![Figure 2.4 Conflict Ratio vs. Average Passing Rate](image)

The Silhouette [16] method is used to determine the best number of clusters. The number that leads to the highest Silhouette value is often considered the ideal number of clusters. Figure 2.5 shows the Silhouette values of each possible number of clusters.

![Figure 2.5 Number of Clusters vs. Silhouette Values](image)
The maximum Silhouette value appears when the number of clusters reaches 3. Therefore, the best number of clusters is 3. Figure 2.6 plots the four distinct clusters using K-Means clustering.

Figure 2.6 Results of Clustering

Cluster 1 has a low Conflict Ratio and a high Average Throughput rate. On the contrary, Cluster 4 has a high Conflict Ratio and low Average Throughput Rate. Cluster 2 and 3 sit somewhere in between. This essentially presents a straightforward way to classify the driver types. Any driving behavior that falls in Cluster 1 can be considered as Conservative Driver since they show less risk in driving; Cluster 4 as Aggressive Driver since they are riskier on the road; Cluster 2 and 3 as the normal driver.

Knowing what parameters in Table 2.2 are related and their values in each scenario, it is possible to utilize it to help define driving behaviors, i.e., aggressive, normal, and conservative. Based on Figure 2.6, accel, decel, and sigma are negatively correlated with aggressiveness of driving behavior while minGap and tau are positively correlated. The
relationship between parameters and driving behavior helps design some tests to compare the network's safety and mobility performance using aggressive drivers and conservative drivers. Table 2.3 lists the aggregated performance of conservative and aggressive at two intersections, respectively.

**Table 2.3 Samples of Calculated Throughput Rates for Two Different Driver Types**

<table>
<thead>
<tr>
<th>Driver Type</th>
<th>Intersection</th>
<th>Number of Entering Vehicles</th>
<th>Number of Exiting Vehicles</th>
<th>Throughput Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conservative 1</td>
<td>30,117</td>
<td>29,876</td>
<td>99.2</td>
<td></td>
</tr>
<tr>
<td>Conservative 2</td>
<td>30,698</td>
<td>30,544</td>
<td>99.5</td>
<td></td>
</tr>
<tr>
<td>Aggressive 1</td>
<td>29,636</td>
<td>21,971</td>
<td>74.1</td>
<td></td>
</tr>
<tr>
<td>Aggressive 2</td>
<td>29,642</td>
<td>18,945</td>
<td>63.9</td>
<td></td>
</tr>
</tbody>
</table>

Conservative drivers will make the traffic move smoothly, while aggressive drivers will cause more congestions. A full-day test is run to monitor the network's mobility using aggressive driver, normal driver, and conservative driver in the simulation testbed, respectively. Figure 2.7 shows the mobility of using different driver types.

![Figure 2.7 Mobility Performance of Different Driver Types in a day](image)

The mobility performance perfectly matches the driving behavior discovered above. Aggressive drivers negatively impact the traffic, while Conservative Drivers are
consistently good. Also, the curve reflects the mobility change with hours of the day. Peak hour mobility is significantly worse than an off-peak hour, especially for bad driving behavior.

Next, the effect of traffic signal timings on mobility and safety is evaluated. The experiment considers two different driver types (i.e., aggressive and conservative) and different traffic timings that varied by either +20% or -40% off actual timings. CRs and TRs from these scenarios are compared to the default signal timings that are collected from the real system. Table 2.4 compares the percentage CR, and TP changed after the treatments on signal lengths for conservative and aggressive drivers.

Table 2.4 Samples Hourly Throughput Rates for Two Different Driver Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Scenario</th>
<th>Coefficient of TLS 1</th>
<th>Coefficient of TLS 2</th>
<th>Avg CR</th>
<th>Avg TR</th>
<th>Percentage CR changed</th>
<th>Percentage TR changed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For each type of driver, scenario 1 is the baseline case with actual signal lengths. The percentage changes of different treatments on the signal length are calculated. For example, signal lengths are cut by 40% for both intersection one and intersection two in scenario 2, and a -1.67% change in CR and a -5.59% change in TR are observed for conservative drivers. To better visualize the trend, Figure 2.8 draw the changes of CRs against the TRs under these scenarios.

![Figure 2.8 Change of CR vs. TR for Two Different Driver Types](image)

Green dots represent scenarios with different signal timings for conservative drivers, and red dots represent aggressive drivers. It is noted that traffic signal timings impact more on mobility for conservative drivers and more on safety for aggressive drivers.

### 2.5 Conclusion

This study demonstrates how traffic signal timings in a roadway network impact a two-dimensional KPI that simultaneously accounts for safety and mobility. The study is carried
out in a small roadway network in China using real traffic timing data and traffic volume data. In the study, two measures are defined for each intersection: Throughput Rate and Conflict Ratio. The combination of the two measures is useful in classifying aggressive, normal, and conservative drivers. According to the findings, the changes in traffic signal timings have higher impacts on safety for aggressive drivers and more on mobility for conservative drivers. In the world of smart roadways and vehicles, the findings from this work can be used to alert drivers adequately and to optimize signal timings at intersections through closer monitoring of road traffic and drivers’ behavior. This study can be extended to include more aspects of intersection characteristics in the future. The model is also appropriate for road safety audits (RSAs) to prioritize improvements with respect to the combined indicator.
3. SAFE ROUTE MAPPING OF ROADWAYS USING MULTIPLE SOURCED DATA

3.1 Introduction

Hot spot identification of roadways using systematic and systemic techniques with historical crash data and qualitative measures [17] has been commonly practiced by traffic authorities and law enforcement agencies for many years. The risk measures computed using these methods do not correctly reflect the fast-changing conditions in roadways, traffic, and weather, and other externalities. Moreover, these static techniques hardly account (aggregation at best) for direct human factors and errors. Many reports consider human errors as the most significant contributing factor to traffic accidents [18–23]. This Chapter intends to fill this important gap by developing a data-driven methodology, referred to as Safe Route Mapping (SRM). This methodology combines over-a-trip safety data and historical crashes to create dynamic risk heat maps of roadways. Onboard vehicle devices, smartphones, and infrastructure-based IoT provide ample opportunities to collect over-a-trip safety data in real-time. These heat maps can be used by authorities to designate safe corridors, dispatch law enforcement, and strategize safety projects using hard near real-time dynamic data. At the same time, individual drivers can also benefit from having access to heat maps data for enhanced situational awareness. This chapter focuses on the SRM methodology and leaves the discussion on the application topics, including the recent attempts by startups and technology companies, to a separate article [24].

In principle, SRM can feed on multiple data sources, e.g., historical crashes and roadway static characteristics from legacy systems, driver-based data from smartphones or onboard devices, any existing systemic data on traffic safety, and dynamic roadway characteristics
from IoT and social media (e.g., black ice, slippery road, etc.). This study will only focus on a subset that includes historical crashes and simulated driver-based data (due to lack of actual data). For illustrative purposes, a real roadway segment where actual crash data is available for a number of years [25] is used. Simulated conflicts will be calculated from the Surrogate Safety Assessment Model (SSAM) that operates on the traffic data from the VISSIM simulation. The contributions of this paper are several folds: (i) An advanced Safety Performance Function (SPF) to estimate the number of crashes [26]. (ii) A methodology to create risk profiles of individual drivers with location and time attributes. A risk profile is a collection of risk measures that are sampled from a driver-based Machine Learning (ML) model. The model takes into account key driving behavioral safety attributes (e.g., aggressiveness). (iii) Aggregation of risk profiles into statistical risk measures for a roadway segment and a given period of time. These statistical measures change with driver mix and traffic flow. (iv) Risk heat mapping of roadway segment or facilities using a hybrid measure composed of SPF estimates and aggregated statistical measures from drivers. This research presents an approach that translates the number of crashes and the probability of conflicts into fuzzy measures and uses fuzzy logic to determine a common denominator.

3.2 Literature Review

The American Association of State Highway Transportation Officials’ (AASHTO) Highway Safety Manual (HSM) guidelines on predictive safety measures for hot-spot ranking and the follow-up diagnosis and countermeasures have been widely used across the United States by many states and municipalities [27]. A number of safety predictive models using crash data have been proposed and widely used in the literature and practice.
SPF is a crash prediction model that estimates the expected average crash frequency of a network, facility, or individual site [27]. A crash modification factor (CMF), which could vary by site specifics and collision, is also applied to estimate the change in the expected number of crashes at the site when a specific countermeasure is implemented [28]. Crash frequency belongs to a general family of stochastic models that are often referred to as counting processes. The Poisson regression model is usually suggested to explain the relationship between crash-frequency with roadway and traffic characteristics [29, 30]. Researchers also use the Poisson-Gamma or Poisson-lognormal model/Negative Binomial (NB) model in which the Poisson parameter follows a gamma or lognormal distribution [31, 32]. Empirical Bayes (EB) method has been suggested to improve the quality of results [33–35]. The EB method uses a weight factor, a function of the SPF over-dispersion parameter, to combine the two estimates into a weighted average [36]. In cases when researchers may not have data to build their own SPF models, paper [37] explores methods to enhance the developed SPF model's transferability.

The current practices of SPF-based predictions suffer from data integrity, lack of sufficient data, and model accuracy due to aggregation and calibration factors. To fill these gaps, risk-based approaches that take advantage of hard data collected from vehicle and roadway sensors slowly but surely reform the traffic safety literature. The SHARP-II Naturalistic Driving Behavior (NDB) and the subsequent safety studies [38–40] strongly support this point. With crashes being rare events, traffic conflicts are used as rough proxies or surrogates for traffic accidents in the Highway Safety Improvement Program [41]. In real-life traffic safety studies, Near-miss accidents often play this role. A near-miss accident can be seen as a vehicle conflict requiring immediate maneuvers to avoid a crash [42].
Near-misses were identified through two-step data processing. Firstly, events were detected using predefined threshold values. If at least one of the threshold values, namely, Lateral Acceleration, Longitudinal Acceleration, Forward/Rear Time-To-Collision (TTC), or Yaw Rate, had been violated, or the driver had activated the Button, a near-miss accident was recorded [42]. Reference also [42] presents detailed information about threshold values and definitions. Several works exist in the literature use driver-based data to analyze collisions and crash surrogates, assess risks, and construct collision avoidance advisory systems [43–46]. Arbabzadeh and Jafari [47] developed an enhanced Advanced Driver Assistance System using real-time safety risk measures predictions based on driver-based data. Using fuzzy logic, the relationships between risk sources and the consequences were identified and quantified in [48]. A systemic analysis in [49] uses a risk-based methodology to identify the relationship between roadway geometric features and crash types. The approach is qualitative and uses crash history on an aggregate basis.

The driver-based and vehicle-based data approaches have been suffering from low market penetration, which gives a non-reliable or low-density level of data. On the other hand, infrastructure sensing requires significant instrumentation via the conventional approach and becomes too costly for potential deployment [50, 51]. With the anticipated increase in the number of connected vehicles, smartphone applications [52–54], and onboard devices, driver-based or vehicle-based data will be less challenging. However, there will still be data confidentiality and security issues to be addressed. To make up for the lack of data, micro traffic simulation software such as VISSIM and SUMO along with Surrogate Safety Assessment Model (SSAM) [55], has been widely used for traffic safety assessment due to its capability to model driver behaviors, vehicle movement, and interactions with
infrastructures and other vehicles [56]. The simulation-based approach provides a reliable and insightful way to proactively diagnose traffic safety problems and evaluate appropriate initiatives without getting into actual crashes [56].

3.3 Description of SRM Methodology

The objective of this Chapter is to develop a methodology that creates dynamic risk heat mapping of road facilities (e.g., intersection, segment, ramps) considering multiple-sourced data such as historical crashes, roadway characteristics, vehicle-based data, and driver-based data. Actual applications using the SRM methodology include, but are not limited to, Real-time Alert for drivers in high-risk situations due to dangerous behaviors such as drowsiness, lane departure, and speeding; Post-trip analysis for driving skill evaluation; and Safe Routes option in navigation using risk heat maps. Table 3.1 summarizes suggested data types for traffic safety studies and brings foundations for implementing the proposed SRM methodology [57].

Table 3.1 Suggested data for a traffic safety study

<table>
<thead>
<tr>
<th><strong>Internal</strong></th>
<th><strong>External</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driver</strong></td>
<td><strong>Road</strong></td>
</tr>
<tr>
<td>Static</td>
<td>Static</td>
</tr>
<tr>
<td>Demographics</td>
<td>Infrastructure, Road geometry, Historical crash</td>
</tr>
<tr>
<td>(age, gender, years of driving, etc.)</td>
<td></td>
</tr>
<tr>
<td>Semi-</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Traffic flow, Traffic signal</td>
</tr>
<tr>
<td>Driver behavior</td>
<td>(drowsy, distracted, texting, etc.)</td>
</tr>
<tr>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Vehicle characteristics (model, make, etc.)</td>
<td>Precipitation, Light, Visibility</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Time</td>
</tr>
<tr>
<td>Speed, Acceleration</td>
<td>Day/night, Traffic peak hour</td>
</tr>
</tbody>
</table>

Asides from basic data types such as speed, yaw rate, and acceleration, more advanced data
types such as drowsiness, distraction, tailgating, and careless lane-change can now be detected by smartphone apps [53, 54]. It is noted that driver-based data changes fast in near real-time, but crash-based data changes very slowly. In the absence of actual driver-based data, this study uses VISSIM for traffic simulation and SSAM for traffic conflict analysis. SSAM computes surrogate measures of safety for each conflict that is identified in the trajectory data from VISSIM (see Appendix Table A1, A2, and Figure A1 for conflict angles and threshold values are taken from [58]). Figure 3.1 shows the framework of the proposed SRM methodology.

![Figure 3.1 SRM implementation flow](image)

The SPF model estimates crash counts for a given roadway segment or facility using Negative Binomial Regression on historical crash data. And the Neural Network (NN) is used to estimate risks from driver-based data. The NN selection follows a study that compares different models for this application. A fuzzy logic-based integration method is employed to fuse the SPF model and Driver-based model data to obtain a single risk measure. In a real-life implementation of SRM, there will be no need for the simulation model.
3.3.1 Crash Prediction Model

In this section, an NB-based SPF model is introduced for crash count prediction using the real crash data, and the EB method is used to improve the quality of results. With crashes being count data, they are often handled by Poisson regression for prediction [56]. Suppose \( \{ N_i(t), t \geq 0 \} \) is the number of crashes in year \( t \) at site \( i \). Many studies assume \( N_i(t) \) a Poisson process [56] with parameter \( \lambda_i \):

\[
f(N_i(t), \lambda_i) = e^{-\lambda_i t} \frac{(\lambda_i t)^{N_i(t)}}{N_i(t)!}
\]  

The mean and variance of such a process are the same and equal to \( \lambda_i t \). Since the crash count is yearly, \( t \) is fixed to 1, and the mean and variance is reduced to \( \lambda_i \). [59] suggests a Poisson regression model with a crash exposure variable for crash count prediction, which can be expressed as:

\[
E(N_i(t)) = \lambda_i = VMT_i^\gamma e^{\sum_j \beta_j x_j}
\]  

Here, \( x_j \)'s are roadway features and traffic conditions for site \( i \) and \( \beta_j \)'s are coefficients. \( VMT \) is vehicle-miles traveled along the site \( i \), and \( \gamma \) poses a potential nonlinear relationship between the crash count and \( VMT \) [59]. The exposure variable modifies the crash count to be weighted on the length of the site. Under a special condition when all sites are uniformly separated, no modification is needed, and (3.2) can be reduced to:

\[
E(N_i(t)) = e^{\sum_j \beta_j x_j}
\]  

The Poisson model assumes the mean equals the variance. Nevertheless, the count data variation is often higher than the mean (i.e., over-dispersion) [56]. The over-dispersed data violates the Poisson assumption and makes it not applicable to use Poisson regression for crash data. \( \lambda_i \) is assumed to be not fixed and follows a Gamma distribution to overcome
the over-dispersion:

\[
f(N_i(t)|\lambda_i, \nu, \delta) = \int_0^{\infty} e^{-\lambda_i} \frac{(\lambda_i)^{N_i}}{N_i!} G(\lambda_i|\nu, \delta) d\lambda_i = \int_0^{\infty} e^{-\lambda_i} \frac{(\lambda_i)^{N_i}}{N_i!} \delta^v \lambda_i^{v-1} e^{-\delta \lambda_i} d\lambda_i \quad (3.4)
\]

Where \( G(\lambda_i|\nu, \delta) \) is Gamma distribution with parameters \( \nu \) and \( \delta \). Using the properties of Gamma distribution, (3.4) can be rewritten as:

\[
f(N_i(t)|\lambda_i, \nu, \delta) = \frac{\delta^v \Gamma(\nu+N_i)(1+\delta)^-(N_i+v)}{\Gamma(v)\Gamma(N_i+1)} \int_0^{\infty} e^{-\lambda_i(1+\delta)} \left( \frac{\lambda_i}{1+\delta} \right)^{N_i} d\lambda_i \quad (3.5)
\]

The outside part resembles a gamma distribution, and the integral part equals 1. Therefore, (3.5) can be rewritten as:

\[
f(N_i|\nu, \delta) = \frac{\Gamma(\nu+N_i)}{\Gamma(v)\Gamma(N_i+1)} \left( \frac{\delta}{1+\delta} \right)^v \left( \frac{1}{1+\delta} \right)^{N_i} \quad (3.6)
\]

This equation is similar to Type 2 Negative Binomial (NB2) distribution with parameters \( r = \nu \) and \( p = (1 + \delta)^{-1} \). The mean of NB2 can be expressed as \( E(N_i) = \mu_i = \frac{\nu}{\delta} \) and the variance of NB2 can be expressed as \( Var(N_i) = \mu_i + \frac{\mu_i^2}{\nu} \). It indicates that the variance is higher than the mean of crash counts. Therefore, NB2 can handle over-dispersion. Remember that \( \mu_i^2 \) is the variance of the Poisson random variable. Therefore, \( \frac{1}{\nu} \) (sometimes shows as \( a \)) is the magnitude of over-dispersion from a Poisson random variable. Using the variance of NB2, (3.6) can be tailored as:

\[
f(N_i|a, \delta) = \frac{\Gamma(\frac{1}{a}+N_i)}{\Gamma\left(\frac{1}{a}\right)\Gamma(N_i+1)} \left( \frac{1}{1+a\mu_i} \right)^{\frac{1}{a}} \left( 1 - \frac{1}{1+a\mu_i} \right)^{N_i} \quad (3.7)
\]

Since the sites are uniformly divided, similar to (3.3), the expected number of crashes is

\[
E(N_i) = \mu_i = e^{\sum_{j=0}^{\nu} \beta_j x_j} \quad (3.8)
\]

\( \beta_j \)'s are estimated using the Maximum Likelihood Estimator (MLE) [60]. Using results from [61] on canonical link negative function and several transformations, a functional
form in terms of input variables (i.e., roadway characteristics, data from crash reports) is found:

\[ \ell(N_i|\boldsymbol{x}_i, \beta, \alpha) = \sum_{i=1}^{n} \left( N_i \ln \frac{\alpha \exp(x_i'\beta)}{1+\alpha \exp(x_i'\beta)} - \frac{\ln(1+\alpha \exp(x_i'\beta))}{\alpha} + C_i \right) \] (3.9)

where \( \alpha = \frac{1}{v} \) and \( C_i \) is \( \ln \Gamma \left( N_i + \frac{1}{\alpha} \right) - \ln \Gamma \left( N_i + 1 \right) - n \Gamma \left( \frac{1}{\alpha} \right) \). \( \alpha \) and \( \beta \) can be obtained numerically [61]. Given estimates of \( \beta \)’s, the model calculates the probability of crashes and necessary statistics on the number of crashes.

An effective way to improve crash predictive methods is to use past crash information, i.e., the EB method, to create a weighted average for improving crash prediction. From (3.5), the posterior distribution of \( \lambda_i \) follows a gamma distribution with parameters \( \nu + N_i \) and \( 1 + \delta \). Thus, given historical data on \( N_i \), the conditional expected value of \( \lambda_i \) is computed as below:

\[ E(\lambda_i|N_i) = \frac{\nu + N_i}{1 + \delta}. \] (3.10)

Recall \( \alpha = \frac{1}{v} \) and \( \delta = \frac{1}{\alpha \mu_i} \), (10) can be further expressed as:

\[ E(\lambda_i|N_i) = \left( \frac{1}{1+\alpha \mu_i} \right) \mu_i + \left( 1 - \frac{1}{1+\alpha \mu_i} \right) N_i \] (3.11)

It shows that the expected number of crashes at site \( i \) is a weighted average of estimated crash counts \( \mu_i \) and actual crashes \( N_i \).

### 3.3.2 Driver-based Model

In this section, a Neural Network (NN) based driver-based model is introduced to predict risk probabilities using simulated data. The selection of NN follows a study that compares different models for this application. Traffic risks are associated with the behaviors of the target vehicle and the interaction with other traffic streams. Besides, vehicles react
differently in different traffic conditions and road sites such as intersections and segments. The proposed model considers those factors to account for the possible impact on traffic risks. Moreover, with crashes being rare events, this study will use traffic conflicts as crash surrogates for developing a driver-based model. Traffic conflicts are influenced by a multitude of factors, such as roadway conditions, driver behavior, and vehicle condition. In actual applications, near-misses can be considered as traffic conflicts. The proposed NN model has three layers: the input layer, hidden layer, and output layer. There are fourteen nodes in the input layer and ten nodes in the hidden layer. The model's input set includes five groups of variables: driver behaviors, roadway characteristics, vehicle statuses, and factors that are distance and speed-related.

It is assumed that three types of driver behaviors, five roadway characteristics, vehicle queuing status, two distance-related measures, and three speed-related measures for model training. The introduced variables are able to explain exogenous influences such as speed difference, distance difference, and headway time between one vehicle and another other than a single-vehicle. Roadway characteristics such as merge, diverge and turning and queue status can also distinguish the impact from different road sites on risks. For instance, most turnings happen at intersections, while moving straight often happens in segments. Besides, if a vehicle’s in-queue status is true, it is safe to say the red traffic signal is on, and the vehicle is waiting at the intersection. A detailed description of the explanatory and response variables used in the study are listed in Appendix Table A5. The complete structure of the neural network is shown in Figure 3.2.
The output of the model is the probability of conflict, and the Sigmoid function is used. The output ranges between 0 and 1, and the input ranges between $-\infty$ and $+\infty$. Gradient descent is used to update the weights and bias. In the output layer, the input, $z$, is given by:

$$z = \sum_{i=1}^{10} w_i x_i + b$$

(3.12)

where $x_i$ is the output from the $i$th neuron in the hidden layer, $w_i$ is the weight multiplied to the $x_i$, and $b$ is the bias. The conflict probability, $y$, is given by the Sigmoid Function:

$$y = \frac{1}{1+e^{-z}}.$$  

(3.13)

### 3.3.3 Integrated Risk Model

Finally, an Integrated Risk Model is introduced to compute a single risk measure from the estimated number of crashes and conflict probabilities. The model uses a fuzzy logic reasoning process. The use of fuzzy logic is motivated by its tolerance to imprecise data and the flexibility of handling complex data. It can model nonlinear relationships of arbitrary complexity by fuzzy rules. The crash count and conflict probability obtained from
the SPF model and driver-based model can be integrated into a single performance measure-risk score by user-defined fuzzy rules. Figure 2.3 depicts the flow of the fuzzy logic reasoning process.

![Figure 2.3 Flow of fuzzy logic reasoning process](image)

The input and output of the reasoning process are crisp numbers limited to a specific range. For instance, the crash probability range is [0,1], and the range of the risk score is [0,100]. Then all the user-defined fuzzy rules are evaluated in parallel using fuzzy reasoning. Finally, the results of the rules are combined and defuzzified into a crisp output value. The study uses \( \text{Min}(\cdot) \) operator for AND fuzzy operation and implication operation; \( \text{Max}(\cdot) \) operator for OR fuzzy operation and aggregation operation. The Center of Area (COA) algorithm is used to find the center of the final aggregated shape for the crisp output value (defuzzification). Also, \( \tilde{C}C \) and \( \tilde{C}P \) denote the fuzzy set of estimated crash counts and the probability of conflicts, respectively. The fuzzy output set, risk score, is denoted as \( \tilde{RS} \).

Gaussian membership functions are used for the fuzzification of input sets and defuzzification of the output set. Each fuzzy set has five membership functions that describe the probabilities of linguistic variables “very low” (VL), “low” (L), “medium” (M), “high” (H), and “very high” (VH), respectively. Figure 4 (a) – (c) show the
membership functions and corresponding standard deviation and mean for conflict probability, crash count, and risk score, respectively.

![Membership functions](image)

**Figure 3.4 Membership functions (a) conflict probability (b) crash (c) risk score**

It is noted that any pair of input values (a, b) for conflict probability and crash count yield fuzzy membership values \(VL(a) \sim VH(a), \ VL(b) \sim VH(b),\) respectively. Table 3.2 shows the matrix representation of all fuzzy rules for \(\tilde{RS}\).

**Table 3.2 The Fuzzy Logic Reasoning Rule Matrix**

<table>
<thead>
<tr>
<th>(\tilde{CP}) (\tilde{CC})</th>
<th>VL</th>
<th>L</th>
<th>M</th>
<th>H</th>
<th>VH</th>
</tr>
</thead>
<tbody>
<tr>
<td>VL</td>
<td>VL</td>
<td>VL</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>L</td>
<td>VL</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>VH</td>
</tr>
<tr>
<td>VH</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>VH</td>
<td>VH</td>
</tr>
</tbody>
</table>

Row captions in the matrix contain the values of the conflict probability can take, column captions contain the values of the crash count can take, and each cell is the resulting risk score when the input variables take the values in that row and column. For instance, the cell (4, 2) can be read as follows: if the crash count is high and the conflict probability is low, then the risk score is medium.
3.4 Illustrative Example

This section demonstrates how crash counts are estimated by the developed SPF model using real crash data for sample roadway segments. Simulated data is used to train the NN model, predict conflict probabilities for the same roadway, and show individual drivers’ risk profiles. Finally, the model integrates the crash count estimates and conflict probabilities into risk scores.

3.4.1 Prediction of Crash Count

Consider the rural roadway, as shown in Figure 3.5. The roadway is divided into ten uniform segments, each segment corresponding to a unique set of physical attributes. It is possible to further divide these segments due to the non-uniformity of risk scores. A total of 1,567 actual crashes were reported for this roadway between 2006 and 2017, with 840 rear-end crashes, 515 as crossing crashes, and 182 lane-change crashes.

![Figure 3.5 Historical crashes identified in study roadway](image)

In Figure 3.5, dots represent crashes, and dot color indicates the severity of crashes; darker color means more severe crashes. Note that a single dot may indicate more than one crash depending on the geospatial resolution. Table 3.3 gives a comprehensive list of
independent variables and sample values used for training purposes.

Table 3.3 Variables for the Proposed Crash Prediction Model

<table>
<thead>
<tr>
<th>Input</th>
<th>Variable name</th>
<th>Data type</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>YEAR</td>
<td>INTEGER</td>
<td>2018</td>
</tr>
<tr>
<td>$x_1$</td>
<td>SEGMENT ID</td>
<td>INTEGER</td>
<td>8</td>
</tr>
<tr>
<td>$x_2$</td>
<td>SRI</td>
<td>INTEGER</td>
<td>18</td>
</tr>
<tr>
<td>$x_3$</td>
<td>FACILITY TYPE</td>
<td>CATEGORICAL</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>AREA TYPE</td>
<td>CATEGORICAL</td>
<td>2</td>
</tr>
<tr>
<td>$x_5$</td>
<td>SEGMENT LENGTH</td>
<td>REAL</td>
<td>0.3</td>
</tr>
<tr>
<td>$x_6$</td>
<td>START-POINT</td>
<td>REAL</td>
<td>0.00</td>
</tr>
<tr>
<td>$x_7$</td>
<td>END-POINT</td>
<td>REAL</td>
<td>3.42</td>
</tr>
<tr>
<td>$x_8$</td>
<td>NUMBER OF LANE</td>
<td>INTEGER</td>
<td>3</td>
</tr>
<tr>
<td>$x_9$</td>
<td>ROAD TOTAL WIDTH</td>
<td>REAL</td>
<td>40</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>SPEED LIMIT</td>
<td>INTEGER</td>
<td>60</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>AADT</td>
<td>REAL</td>
<td>30,921</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>LANE WIDTH</td>
<td>REAL</td>
<td>12</td>
</tr>
<tr>
<td>$x_{13}$</td>
<td>SHOULDER WIDTH</td>
<td>REAL</td>
<td>8</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>SHOULDER TYPE</td>
<td>CATEGORICAL</td>
<td>1</td>
</tr>
<tr>
<td>$x_{15}$</td>
<td>PRESENCE OF MEDIAN</td>
<td>BINARY</td>
<td>0</td>
</tr>
<tr>
<td>$x_{16}$</td>
<td>MEDIAN WIDTH</td>
<td>REAL</td>
<td>30</td>
</tr>
<tr>
<td>$x_{17}$</td>
<td>MEDIAN BARRIER</td>
<td>BINARY</td>
<td>0</td>
</tr>
<tr>
<td>$x_{18}$</td>
<td>PASSING LANE</td>
<td>INTEGER</td>
<td>0</td>
</tr>
<tr>
<td>$x_{19}$</td>
<td>2-WAY LEFT-TURN</td>
<td>BINARY</td>
<td>0</td>
</tr>
<tr>
<td>$x_{20}$</td>
<td>LIGHTING</td>
<td>BINARY</td>
<td>0</td>
</tr>
<tr>
<td>$x_{21}$</td>
<td>PRESENCE OF ON-STREET PARKING</td>
<td>BINARY</td>
<td>0</td>
</tr>
<tr>
<td>$x_{22}$</td>
<td>TYPE OF ON-STREET PARKING</td>
<td>BINARY</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Variable name</th>
<th>Data type</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>TOTAL NUMBER OF CRASHES</td>
<td>INTEGER</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 3.6 compares the actual and estimated crash counts by using the above SPF model between 2006 and 2017. Each time bucket includes ten uniform rural roadway segments.
Figure 3.6 The actual and estimated number of history crashes for the sample roadway segments in a rural multilane highway

Using the Freeman-Tukey coefficient of determination ($R^2 = 0.98$), it can be seen that the above prediction model can adequately explain the variation around crash counts. Recall that a calibration factor is introduced to adjust the estimated crashes using real crash data. Here, the calibration values $C_r$’s for the rural model are at 0.99, which means that exogenous factors that can potentially affect the crash rates in this location are negligible. Table 3.4 lists the predicted number of crashes in each segment for 2018 by the proposed SPF model.

**Table 3.4 Predicted Number of Crashes For 2018**

<table>
<thead>
<tr>
<th>Segment ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crashes counts</td>
<td>4</td>
<td>2</td>
<td>25</td>
<td>4</td>
<td>20</td>
<td>11</td>
<td>25</td>
<td>9</td>
<td>19</td>
<td>12</td>
</tr>
</tbody>
</table>
The above analysis evaluates the risk of roadway from a macro perspective by predicting the number of crashes. To assess the roadway from a micro perspective, the driver-based risk profile is created next.

### 3.4.2 Risk Profile of Individual Drivers

This section will demonstrate how driver-based conflict probability can be used to create risk profiles for individual drivers. The following study shows how these profiles can be aggregated temporally and spatially. Traffic conflicts will be calculated using VISSIM and SSAM models of the roadway in Figure 3.5. According to SSAM, there are three types of conflicts: Rear-end, Crossing, and Lane change. In addition, three different driver behavior models (i.e., Type 1, Type 2, and Type 3), used in a report by the Kentucky Transportation Cabinet [62], will be considered. Appendix Table A3 and A4 define the car following driver behavior and lane change parameters used in the driver-based study, respectively. According to the two tables, observed vehicles, average standstill distance, additive and multiplicative parts of safety distance are the variables that define car-following behavior. Similarly, variables such as maximum deceleration and safety distance reduction factor distinguish lane changing behaviors. Three levels of traffic volume (L1: 400 vehicles/hour), (L2: 800 vehicles/hour), and (L3: 1,200 vehicles/hour) are considered. A mixed case, which includes all driver types, is used in the simulation. Table 3.5 shows the conflicts by type from SSAM for a total of 16 simulation scenarios.
Table 3.5 Conflict Identified by SSAM In Different Scenarios

<table>
<thead>
<tr>
<th>id</th>
<th>Driver type</th>
<th>Volume</th>
<th>Rear-end</th>
<th>Crossing</th>
<th>Lane change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>L1</td>
<td>186</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>L2</td>
<td>740</td>
<td>25</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>L3</td>
<td>5,048</td>
<td>58</td>
<td>301</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Mixed</td>
<td>1,861</td>
<td>30</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>L1</td>
<td>184</td>
<td>11</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>L2</td>
<td>745</td>
<td>20</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>L3</td>
<td>4,933</td>
<td>52</td>
<td>382</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Mixed</td>
<td>1,857</td>
<td>32</td>
<td>133</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>L1</td>
<td>183</td>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>L2</td>
<td>764</td>
<td>19</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>L3</td>
<td>5,244</td>
<td>58</td>
<td>348</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Mixed</td>
<td>1,853</td>
<td>33</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>L1</td>
<td>172</td>
<td>7</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>L2</td>
<td>1,103</td>
<td>20</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>L3</td>
<td>4,456</td>
<td>70</td>
<td>307</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Mixed</td>
<td>1,857</td>
<td>28</td>
<td>160</td>
<td></td>
</tr>
</tbody>
</table>

9,210 observations are selected from the simulation and divided into a training dataset (60%) and a testing dataset (40%). Using the Neural Network of Figure 3.2, the conflict probability concerning different driver types and traffic levels is computed. Table 3.6 compares training results and testing results.

Table 3.6 Training Vs. Testing the Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Training N=5526</th>
<th>Actual Non-conflict</th>
<th>Actual conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Non-conflict</td>
<td>3846</td>
<td>533</td>
<td>87.8%</td>
</tr>
<tr>
<td>Predicted Conflict</td>
<td>537</td>
<td>610</td>
<td>53.2%</td>
</tr>
<tr>
<td></td>
<td>87.7%</td>
<td>53.4%</td>
<td>81.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Testing N=3684</th>
<th>Actual Non-conflict</th>
<th>Actual conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Non-conflict</td>
<td>2554</td>
<td>377</td>
<td>87.1%</td>
</tr>
<tr>
<td>Predicted Conflict</td>
<td>370</td>
<td>383</td>
<td>50.9%</td>
</tr>
<tr>
<td></td>
<td>87.3%</td>
<td>50.4%</td>
<td>79.6%</td>
</tr>
</tbody>
</table>

The rows of the matrices are the predicted class, and the columns are the actual class. The diagonal cells represent correctly classified observations. The off-diagonal cells represent
observations that are incorrectly classified. The last column of the plot shows the percentages of correct and incorrect predictions (positive predictive value and false discovery rate, respectively). The bottom row shows the percentages of classified and unclassified examples in each class. These metrics are often called the true positive rate and false-negative rate, respectively. It is noted that the overall prediction accuracy of training data is 81.0%, and testing data is 79.6%. Asides from the confusion matrix, the ROC curve also measures the goodness of the classification is. Figure 3.7 shows the ROC curve of the Neural Network. AUC values for training and testing are 0.84 and 0.81, respectively.

![Training ROC vs Testing ROC for Neural Network](image)

**Figure 3.7 Training ROC VS. testing ROC**

An important remark is in order here. The above model does not have any inputs for weather conditions, vehicle conditions, or time of day. These could significantly influence traffic conflicts. The impact of these factors cannot be demonstrated in a simulation model, and as such, are omitted from the analysis. Thus, in this article, and only due to simulation
data, risks are not dependent on the time of day, nor are they dependent on vehicle or weather conditions. In the absence of these factors, the traffic conditions and vehicle speed are the fast-changing inputs that could alter risks as one drives through a roadway stretch. For a given driver type traveling through a roadway distance, a risk profile is then defined by collecting conflict risks sampled at some constant time intervals (e.g., every 1 second). To demonstrate the idea of risk profiles, a 700-feet long roadway sub-section that includes an intersection is selected from Figure 3.5. This sub-section is presented in Figure 3.8.

![Sample roadway section](image)

**Figure 3.8 Sample roadway section**

For demonstration purposes, the example focuses on the traffic from West to East. The subsection is further divided into three segments: Link 1, Link 2 (intersection), and Link 3. Running the trained Neural Network model for different traffic levels and driver behaviors, the probability of conflicts is computed at constant time intervals. Figure 3.9 depicts the relationship between risk profiles and locations given a fixed time for Type 1 driver in L1 traffic.
Figure 3.9 Risk profile of the sample roadway section

Figure 3.9 (a) gives a risk profile for a Type 1 driver passing through the three links in L1 traffic level, and Links are colored differently. Figure 3.9 (b) shows conflict risks for many Type 1 drivers who pass through the three links. Finally, Figure 3.9 (c) shows the average conflict risks for all sampled drivers. It is noted that these samples are taken within a given time period on a given day. One can always repeat this sampling for the same period for many days to receive aggregate risk measures that span over drivers and days. Figures 2.10 and 2.11 show how average risk profiles change with driver types and traffic conditions, respectively.
In all cases, Link 2 shows a higher conflict probability than the others. As the traffic level changes, the overall risk profiles fluctuate. Besides, the Type 2 driver seems to be more conflict-prone. Next, the model fuses crash count estimates with driver-based risk measures to obtain a more comprehensive risk heat map.
3.4.3 Integration of Crash Data and Conflict Data

From the results of crash and conflict estimates, the range of membership functions for crash counts is set to [0, 25] since the maximum number of predicted crashes in 2018 is 25 (Table 4), and the range of membership functions for conflict probability is set to [0, 1]. Inspired by safe driving Apps [63, 64], the range of membership functions for risk score is set to [0, 100], with 100 being extremely risky. It is also a straightforward way to display the risk level for the driver. By defuzzifying the aggregated shape of pairwise comparisons for all the rules, risk scores are obtained. Figure 3.12 illustrates the reasoning process of 3 sample fuzzy rules for an input pair (conflict probability = a%, crash count = b).

![Figure 3.12 Illustration of the fuzzy reasoning process](image)

Simulation is run for 5,400 simulation seconds, and conflict analysis is carried for the roadway in Figure 3.5. The simulation is divided into 20-time slots, each with 270 seconds. The result is a 10x20 matrix where rows correspond to segment IDs and columns to the time slots. Consider Cell (10,3) for segment three and the time slot between 2,430 and 2,700 seconds. There is an estimate of 25 crashes (see Table 3.4) for this segment that stays the same for the simulation period since no traffic crash can be observed in the simulation.
model. For every driver who uses segment 3 for 2,430-2,700 seconds, a conflict risk at constant time intervals is computed. These samples constitute a risk profile of a driver stamped by segment ID. Recall that a risk profile is a collection of risk values sampled at constant time intervals as a driver goes a distance. In theory, many of these risk profiles correspond to Cell (10,3). Figure 3.13 gives risk scatter plots of Type 1 drivers for Cell (10,3) for L1 and L3 traffic volumes. According to these plots, the average and disparity of conflict risks for the same roadway segments and the corresponding periods increase with traffic volume.

![Figure 3.13 Individual risk profiles of Type 1 driver in (a) L1 and (b) L3 traffic](image)

**Figure 3.13 Individual risk profiles of Type 1 driver in (a) L1 and (b) L3 traffic**

Here, a simple grand average of conflict risks is taken in a given cell. With the driver-based conflict and crash estimate for each cell, the fuzzy model is used to estimate the risk score. Figure 14 and 15 show the risk score of different driver types in L2 traffic and Type 1 driver in different traffic levels, respectively.
Figure 3.14 - Risk heat maps of different types of drivers in L2 traffic

Figure 3.15 Risk heat maps of Type 2 driver in different traffic levels

The comparison of the same cells in Figure 3.14 (a-c) yields interesting results concerning the driver type. Type 2 drivers tend to have high risks since their scores are higher, while Type 3 drivers show low risks. The risk heat maps also reflect the impact of the crash count on different roadway segments. Associating risk heat maps with Table 3.4, it can be seen
that segments with high-risk scores also with high crash frequency, such as 3, 5, 7, and 9. Besides, Figure 3.15 depicts the correlation between traffic volumes and risk scores. L1 traffic results in the lowest risk score, and L3 traffic leads to the highest risk score.

3.5 Conclusion

In this chapter, a novel Safe Route Mapping (SRM) method is developed to score roadway safety using legacy crash-based data and driver-based data that can be obtained from onboard devices and infrastructure sensors. An NB-based SPF model considering many roadway features is used to estimate the number of crashes, and the EB method is used to improve its performance. An NN model considering endogenous and exogenous factors from drivers and roadways is trained to predict vehicle conflicts dynamically. A fuzzy logic-based integration method is utilized to integrate different performance metrics into a single one. The study first demonstrates how to develop the SPF model and use real crash data for crash prediction on sample roadway segments. Then, it shows how to train the NN model with multiple sources of data such as roadway information, driver behaviors, and speed difference as inputs to predict conflict probabilities for the same roadway. Lastly, it illustrates the procedure to integrate crash estimates with conflict probabilities into a hybrid risk score. Due to the lack of real-time driver-based data, a VISSIM simulation model is built to get vehicle trajectories for the case study and use the SSAM tool to obtain conflict data. It is shown that the SPF and the NN estimates are statistically sound. Sampling risk scores at constant time intervals as vehicles travel over a distance yield risk profiles, which are then used for risk heat mapping. Results show that road safety and drivers’ risk profiles are associated with driver behaviors, road characteristics, and traffic conditions. The proposed SRM methodology is general and can be applied to real-life applications if actual
data can be collected. However, this paper's demonstration is only limited to driver-based data that can be obtained from simulation sources.

There are certain limitations that can be addressed in future work. For instance, the developed SPF model can be applied to roadways in different states to test its transferability. The driver-based model can be extended to include more factors such as road facility types, road sites, and traffic signals for real-time traffic conflicts prediction. With additional data sources from onboard sensors equipped on future connected and automated vehicles (e.g., LiDAR, cameras, sensors), safe route recommendations for individual drivers can be realized using the risk heat maps generated by the SRM methodology.

3.6 Appendix

Table A1 Conflict Thresholds Used in the Simulation

<table>
<thead>
<tr>
<th>Conflict-related parameters</th>
<th>Threshold value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum time-to-collision (TTC)</td>
<td>1.5″</td>
</tr>
<tr>
<td>Maximum post-encroachment time</td>
<td>5″</td>
</tr>
<tr>
<td>Rear-end angle</td>
<td>30°</td>
</tr>
<tr>
<td>Crossing angle</td>
<td>80°</td>
</tr>
</tbody>
</table>

Table A2 Criterion of Defining Conflict Type

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Conflict type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0^\circ \leq \theta &lt; \theta_1$</td>
<td>Read-end</td>
</tr>
<tr>
<td>$\theta_1 \leq \theta &lt; \theta_2$</td>
<td>Lane change</td>
</tr>
<tr>
<td>$\theta_2 \leq \theta \leq 180^\circ$</td>
<td>Crossing</td>
</tr>
</tbody>
</table>
Table A3 Car Following Driver Behavior Parameters

<table>
<thead>
<tr>
<th>Car Following Parameters</th>
<th>Driver Behavior Model</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look Ahead Distance (ft)</td>
<td>0 - 820.21</td>
<td>0 - 820.21</td>
<td>0 - 820.21</td>
<td></td>
</tr>
<tr>
<td>Observed Vehicles</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Look Back Distance (ft)</td>
<td>0 - 492.13</td>
<td>0 - 492.13</td>
<td>0 - 492.13</td>
<td></td>
</tr>
<tr>
<td>Temporary Lack of Attention</td>
<td>Duration (s)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Probability (%)</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Average Standstill Distance (ft)</td>
<td>5.00</td>
<td>6.56</td>
<td>8.00</td>
<td></td>
</tr>
<tr>
<td>Additive Part of Safety Distance</td>
<td>1.50</td>
<td>2.60</td>
<td>2.50</td>
<td></td>
</tr>
<tr>
<td>Multiplicative Part of Safety Distance</td>
<td>2.50</td>
<td>3.60</td>
<td>3.50</td>
<td></td>
</tr>
</tbody>
</table>

Table A4 Lane Change Parameters

<table>
<thead>
<tr>
<th>Lane Change Parameters</th>
<th>Driver Behavior Model</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Deceleration (ft/sec(^2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Own</td>
<td>-16.00</td>
<td>-13.12</td>
<td>-10.00</td>
<td></td>
</tr>
<tr>
<td>Trailing Vehicle</td>
<td>-12.00</td>
<td>-9.84</td>
<td>-8.00</td>
<td></td>
</tr>
<tr>
<td>-1 ft/sec(^2) per distance (ft)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Own</td>
<td>75</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Trailing Vehicle</td>
<td>75</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Accepted Deceleration (ft/sec(^2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Own</td>
<td>-3.28</td>
<td>-3.28</td>
<td>-3.28</td>
<td></td>
</tr>
<tr>
<td>Trailing Vehicle</td>
<td>-3.28</td>
<td>-3.28</td>
<td>-3.28</td>
<td></td>
</tr>
<tr>
<td>Waiting Time Before Diffusion (sec)</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Minimum Headway (front/ear) (ft)</td>
<td>1.64</td>
<td>1.64</td>
<td>1.64</td>
<td></td>
</tr>
<tr>
<td>Safety Distance Reduction Factor</td>
<td>0.55</td>
<td>0.70</td>
<td>0.65</td>
<td></td>
</tr>
<tr>
<td>Cooperative Lane Changed Allowed</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Table A5 Predictor and Response Variables Used for The Risk Model

<table>
<thead>
<tr>
<th>Variable Category</th>
<th>Independent Variable</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type1</td>
<td>bool</td>
<td>Type 1 driver</td>
</tr>
<tr>
<td></td>
<td>Type2</td>
<td>bool</td>
<td>Type 2 driver</td>
</tr>
<tr>
<td></td>
<td>Type3</td>
<td>bool</td>
<td>Type 3 driver</td>
</tr>
<tr>
<td>Roadway Characteristic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>bool</td>
<td></td>
<td>Left turn lane</td>
</tr>
<tr>
<td>Right</td>
<td>bool</td>
<td></td>
<td>Right turn lane</td>
</tr>
<tr>
<td>Straight</td>
<td>bool</td>
<td></td>
<td>Straight lane</td>
</tr>
<tr>
<td>Merge</td>
<td>bool</td>
<td></td>
<td>Merge lane</td>
</tr>
<tr>
<td>Diverge</td>
<td>bool</td>
<td></td>
<td>Diverge lane</td>
</tr>
<tr>
<td>Vehicle Status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InQueue</td>
<td>bool</td>
<td></td>
<td>In queue</td>
</tr>
<tr>
<td>Distance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SafeDist</td>
<td>double</td>
<td></td>
<td>Safe distance</td>
</tr>
<tr>
<td>HDWY</td>
<td>double</td>
<td></td>
<td>Headway distance</td>
</tr>
<tr>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpeedDiff</td>
<td>double</td>
<td></td>
<td>Speed difference between lead and following vehicles</td>
</tr>
<tr>
<td>vehSpeed</td>
<td>double</td>
<td></td>
<td>Vehicle’s actual speed</td>
</tr>
<tr>
<td>Acceleration</td>
<td>double</td>
<td></td>
<td>Vehicle’s acceleration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable Category</th>
<th>Dependent Variable</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflict probability</td>
<td>Conflict</td>
<td>bool</td>
<td>Conflict</td>
</tr>
</tbody>
</table>
4. A DISTRIBUTED MULTI-AGENT REINFORCEMENT LEARNING WITH GRAPH DECOMPOSITION APPROACH FOR LARGE-SCALE ADAPTIVE TRAFFIC SIGNAL CONTROL

4.1 Introduction

Traffic congestion is an inescapable problem that frustrates drivers in megacities. Although there is hardly a way to eliminate the congestion in its entirety, it is possible to mitigate the impact and reduce its size. Many studies have shown that adaptive traffic signal control (ATSC) improves traffic performance, such as emissions, travel time, and fuel consumption by at least 10% [65]. If the system is under saturated conditions and with extremely outdated signal timing, the improvement can be 50% or more. However, most of the currently deployed traffic signal systems do not utilize ATSC. In the United States alone, people must collectively wait 296 million hours every year, averaging one hour per person due to traffic control systems [66]. These delays negatively impact the economy and environment.

There are already some adaptive solutions to the literature's traffic signal scheduling problem [67–69]. Theoretically speaking, these solutions' optimality is hard to reach, and computational requirements are excessive under real-life scenarios [70]. However, significant traffic congestion improvements have been reported in many real-life implementations [71-72]. Different from papers [73-74] that consider cumulative travel time and waiting time as elements in the control algorithms affecting drivers’ patience and safety awareness, this study introduces the priority function that takes the accumulated exponentially weighted waiting time of each vehicle in the network into account to amplify the impact of waiting time. This chapter then develops a novel RL-based ATSC algorithm
using the priorities mentioned above for signal decisions. Methodologically speaking, the proposed approach improves existing models [73-74] by considering accumulated exponentially weighted waiting time of each vehicle at an intersection for signal sequencing and adopts Multi-Agent Reinforcement Learning (MARL) methods for signal timing. The priority grows exponentially as waiting time accumulates. The longer a vehicle waits at the intersection, the higher priority is incurred. The traffic controller is more imperative to let go of the vehicles in directions with higher priorities. Such a smart controller will find proper control strategies to reduce congestions, traffic delays, and the number of vehicles in queues at the intersection. The interaction between road users and the controller will train this controller to set the right traffic signal times, given any traffic patterns.

Dynamic traffic signal control is a complex problem, especially when it comes to multiple intersections. Exact formulation and solution to network-level problems with interacting traffic signal controllers would be prohibitive using traditional methods. In recent years, the idea of using RL in adaptive traffic signal control has received attention from many researchers. RL is formulated under the framework of the Markov Decision Process (MDP) and presents an unsupervised way to solve the problem where patterns are dynamically changing. Unlike traditional learning methods, it utilizes a reward signal, with no explicit mapping from input to target data, but instead aims to maximize the reward it receives, e.g., minimum waiting time or the number of vehicles in the queue ATSC. This paper adopts the state-of-the-art deep RL technique, Double Dueling Deep Q Network (DDDQN) with Prioritized Experience Replay (PER). The proposed AEWWT-ATSC with cooperative Multi-agent Reinforcement Learning (MARL) can improve intersections' performance in
terms of the number of vehicles in the queue and total waiting time. It is ideal to train a single model that uses full information for everything. However, it brings tremendous challenges to large-scale cooperative MARL ATSC problems computationally. Therefore, a graph decomposition method that breaks down the graph into subgraphs and a distributed MARL approach that trains each subgraph in a synchronized way are proposed. Subgraphs are trained separately, but the information is passed between subgraphs at fixed time intervals. In this way, the target subgraph agents can communicate with “dumb” agents from other subgraphs. This approach is generic and can be extended to various types of use cases. The study also examines the performance of different RL-based control methods for single and multiple intersection set-ups, including a large network in Manhattan, NYC, with the proposed distributed training approach.

In principle, this chapter's contributions are several folds: (i) An RL-based AEWWT-ATSC that captures both the number of vehicles in the queue and exponentially weighted waiting times as the control mechanism, aiming at investigating the impact of cumulative waiting time on the priority in signal decisions. The proposed AEWWT-ATSC model calculates priorities to signify the current traffic condition in the sequencing of the signals. The control agents are also allowed to choose appropriate signal times rather than taking action at fixed times. (ii) A comprehensive study of different sizes of networks, traffic volumes, and control policies. The experiment compares the performance of fixed timing, fixed sequencing, dynamic sequencing, and dynamic timing with and without AEWWT-ATSC to reveal the proposed method's superiority. (iii) A distributed MARL (DMARL) training approach decomposes large graphs into several subgraphs and trains each subgraph
in a synced way. This approach could save computational time for large-scale cooperative MARL ATSC and achieve near-optimal performance.

### 4.2 Literature Review

Traditional signal systems use fixed timing plans to prepare offline control based on historical traffic flow data. Fixed-time plans, however, cannot deal with the variability of traffic patterns throughout a day. Hence, they become inefficient due to traffic growth and changes in traffic patterns [75]. There are also signal systems that employ nonfixed-time plans to account for the variability of traffic flow. [73-74] consider the cumulative travel-time of vehicles to determine optimal green signal timing for the next signal cycle under the connected vehicle environment. These plans are based on traffic volume and road occupancy from detectors located in the roadway's critical locations. The system operator may also override the fixed timings based on real-time surveillance data [76]. Unlike the traditional signal timing process, “Online” control systems update real-time timing plans based on detectors data. There are two major categories in this strategy: (a) signal configuration is adjusted while keeping a common cycle length; and (b) Adaptive Traffic Control, where signal timings are optimized frequently. Adaptive control systems rely on accurate and fast detection of the current conditions in real-time to allow for an effective response to any changes in the current traffic situation [77].

The formulation of the ATSC varies by different methods. [78] discusses a 0-1 mixed-integer linear programming formulation of the traffic signal control problem in some work. A distributed-multiagent-based approach for traffic signal control is presented in [79]. In [80] and [81], the traffic management problem is formulated as an optimization problem, and genetic algorithms are used to solve this problem. Genetic algorithms provide a
heuristic optimization technique for such problems. In [82], a Markov decision process (MDP) framework for adaptive control of traffic lights is considered. However, the information on the system's transition probabilities is often not available. Therefore, RL methods are introduced to fill in the gap. There are three classes of RL methods: value-based, policy-based, and Actor-critic. Value-based methods first find the optimal value function and then extract an optimal policy [83]. Popular methods are Temporal Difference (TD), State-action-reward-state, and Q-learning. Policy-based methods directly search in policy spaces without using a value function [84]. An important member of policy-based methods is Policy-gradient [84]. Actor-critic is a mix of policy-based and value-based methods, which solves the conventional action selection policy and value function simultaneously to find the optimal policy [85]. For large-scale reinforcement learnings, it is difficult to train the agents in a reasonable amount of time by a single model. Therefore, researchers propose boosting strategies for deep reinforcement learning to accelerate performance [86-88]. [86] presents the massively distributed architecture for Deep Q-Network with parallel neural networks and relay memories. [87] proposes an accelerating distributed RL approach to reduce the end-to-end network latency for synchronous training and improve the convergence rate. [88] proposes Q-decomposition, where a complex agent is built from simpler subagents. Each subagent has its reward function and runs its reinforcement learning process.

In [89-90], the authors adopted Double Deep Q Network (double DQN) [91], Dueling Network Deep Q Network (Dueling DQN) [92], and Prioritized Experience Replay (PER) [93] for a single intersection. They consider timing change as the action, vehicle position, and speed as the state, and the shift in queue size as the reward. With Double and Dueling
DQN being state-of-the-art RL that can effectively reduce the overestimation and improve the system performance, PER allows the system to reuse important memories to boost the learning. The two papers achieve a significant reduction in the traffic delay. Asides from ATSC for a single intersection with one agent, some other papers use MARL for multiple intersections [94-95]. In [94], the authors adopt Q-learning with function approximation, which takes advantage of feature-based state representations that characterize the traffic congestions into low, medium, and high levels to solve the curse of state dimensionality when using full traffic states. It also compares the result of the independent model. Each agent independently controls the signal, and the integrated model, where each agent interacts with its neighboring agents for a partially observable network. At fixed times, each agent takes action to determine whether to switch the signal (with randomly selected directions that receive the next green signal).

Similarly, paper [95] compares dependent Advantage Actor-critic (A2C) with independent A2C and independent Q-learning for multiple intersections. Researchers attempted to synthetic the traffic signals for intersections by sharing common signal timings and sequencings among neighboring intersections rather than the entire network to stabilize the learning procedure and training difficulty for each agent in large road networks [94-95]. These studies' action space is the possible phase configurations, and all decisions are made at fixed intervals. Some researchers discuss the clustering method for traffic intersections, which can be used for distributed MARL in traffic signal control [96]. The author clusters urban intersections by incorporating traffic flow uncertainties such as demand volume to capacity ratio, queue length, and delay [96].
There are significant gaps between theoretical advances and real-world practices in adaptive traffic signal control: (i) Current RL-based ATSC methods do not consider the impact of each vehicle's exponentially weighted waiting time in signal decisions. (ii) Most RL-based ATSC approaches simply assume cyclically or randomly switched signals and take actions at fixed times. (iii) There is no systematic approach that breaks down large-scale ATSC problems into sub-ATSC problems and solves them with boosting strategies.

4.3 Problem Formulation and Modeling

This section introduces a testbed of connected signalized intersections built by VISSIM, as shown in Figure 4.1. Each intersection has its queue, consisting of twelve lanes (d1-d12), and each lane represents a direction. For example, the Eastbound of the left intersection has three lanes (d1, d2, and d3), where d1 is dedicated to the left-turn movements, d2 is dedicated to straight movements, and d3 is dedicated to right-turn movements. The simulation model creates equal length for each lane (500 meters) and makes all vehicles identical when modeling the intersections to eliminate the bias caused by the variation of road geometries and vehicle attributes. The following sections formulate the interaction between vehicles and the traffic signal agent and model the signal control by MDP.

![Figure 4.1 The geometry of the study intersection](image)
4.3.1 *AEWWT-ATSC Model*

The control strategy developed here uses some of the essential elements of Organizing Traffic Light control (SOTL) [97] that gives vehicles waiting longer or a larger group of vehicles and combines them with the exponential function. The exponential function amplifies the impact of the accumulated waiting time. Every time a vehicle comes to a stop at an intersection and waits for the green signal, it engages in a priority calculation with the intersection smart controller. The priority generated by the AEWWT-ATSC model is an exogenous quantity, which provides a basis for signal timing and sequencing.

Before formulating the problem, the grouping policy at each intersection is introduced. Depending on the intersection's traffic signal phasing and engineering design, non-conflicting directions can be grouped into the same queue. Figure 4.2 demonstrates conflicting and non-conflicting groupings.

**Figure 4.2** (a) non-conflicting grouping (b) conflicting grouping

For a single intersection in Figure 4.2 (a), traffic flows along with directions d1, d2, and d3 are conflict-free and can use a single green phase. In contrast, releasing vehicles from d4 and d8 at the same time will cause conflicts as they will meet up at some points. Therefore, when grouping directions into queues, conflicts shall be strictly avoided. The
study only allows one group in the following list to open at a time: \{[d1, d2, d3], [d4, d5, d6], [d7, d8, d9], [d10, d11, d12]\}. Therefore, vehicles from Eastbound, Westbound, Northbound, and Southbound are named \(Q_E, Q_W, Q_N\) and \(Q_S\). Suppose that every time a vehicle comes to a stop at an intersection along a direction, say \(d\), an interaction is incurred between the intersection smart control agent and the vehicle. This interaction involves a priority value, \(W_i\), the vehicle \(i\) with cumulated waiting time \(T_i\). The priority function for vehicle \(i\) is presented as:

\[
P_i = W_i \cdot e^{T_i}
\]  

where \(W_i\) is set to 1, but the choice of using different values is allowed. For instance, emergency vehicles or transit buses may have higher values than passenger cars, the same for carpool vehicles with two or more passengers. \(T_j, t_n\) is the waiting time of vehicle \(j\) at \(t_n\).

A cycle is a time between two red signals, and the composition of each queue can be a mix of remaining vehicles from the last cycle and new arrivals, as shown in Figure 4.3.
Thus, the total waiting time of a vehicle $j$ at $t_n$ have two possibilities:

$$T_{j,t_n} = \begin{cases} T_{j,r_m} + t_n - r_m, & H_j < r_m \ j \in \Omega_{d,t_n} \\ t_n - H_j, & H_j \geq r_m \ j \in \Omega_{d,t_n} \end{cases}$$  \hspace{1cm} (4.3)$$

where $T_{j,r_m}$ is the cumulative waiting time of vehicle $j$ that arrives in the previous cycle. $T_{j,r_m}$ is zero if vehicle $j$ arrives in the current cycle. $H_j$ is the arrival time of vehicle $j$. The priority for queue $q$ at $t_n$ is given by:

$$\pi_{q,t_n} = \sum_{d \in \Omega_q} P_{d,t_n}$$  \hspace{1cm} (4.4)$$

where $\Omega_q$ is the set of directions in queue $q$ at $t_n$. Note that the proposed AEWWT-ATSC model captures both the number of vehicles waiting at the red signal for all directions $d$ and the waiting times of these vehicles, a similar concept used by SOTL. However, the treatment of waiting times through an exponential function is unique. The term $e^{T_{j,t_n}}$ shows an increasing trend over waiting times. For a particular case without $e^{T_{j,t_n}}$, the AEWWT-ATSC model reduces to the Null model, which captures only the number of vehicles $\sum_{j \in \Omega_q} W_j$. At $t_n$ the green light is assigned to directions that belong to queue $q$ if

$$\pi_{q,t_n}^* \geq \pi_{\ell,t_n} \ for \ all \ \ell \neq q$$  \hspace{1cm} (4.5)$$

In simple terms, the model looks for the queue that has the highest priority at decision time $t_n$ for each intersection. And the new decision is made at the end of the current green cycle.
4.3.2 Formulation of Signal Scheduling

The signal scheduling for the road network is formulated as a Multi-agent Reinforcement Learning (MARL) problem. Given the priority calculated by (4.2), the system internally chooses a queue to receive the next green signal according to (4.5). Each agent focuses on the selection of green signal length. As mentioned in the literature review, two types of MARL shall be discussed: Competitive MARL and Cooperative MARL. The latter can be further divided into fully and partially observable cooperative MARL. The key difference between them is how much information each agent would like to share and if they have the same goal. In a cooperative traffic system, all agents share their traffic states and seek to minimize the entire network's waiting time and queue sizes. The joint state space would be explosive as the network gets bigger.

In contrast, agents in a competitive traffic system keep on their traffic state and work independently. As the network's size grows, it adds complexity and challenges to computational efficiency [98]. In a partially observable cooperative setting, the agent only shares its information with neighborhood agents, and information is only shared among group members. Partially observable cooperative MARL may reduce the state space, but the problem's scale remains very large. Therefore, this research proposes a decomposition method that divides the network into subnetworks and performs fully observable cooperative MARL on each subnetwork. This way, not only the state space but also the problem scale is reduced.

Considering a decomposition of a network $G$ is a set of subnetworks $G_1, G_2, ..., G_k$ that partition the intersections of $G$. That is, for all $i$ and $j$, $V(G_i) \cap V(G_j) = \emptyset$, $E(G_i) \cap E(G_j) \neq \emptyset$, and $\bigcup_{1 \leq i \leq k} G_i = G$. If the number of partitions equals the number of
intersections, then the problem becomes competitive MARL. Each agent is an independent Markov Decision Process (MDP) with \((S^{(v)}, A^{(v)}, R^{(v)}, P^{(v)})\) representing state space, action, reward, and transition probability for Agent, \(v \in V\). The transition probability for Agent \(v\) from the state \(s^{(v)}_{t_n}\) to the new state \(s^{(v)}_{t_{n+1}}\) after taking action \(a^{(v)}_{t_n}\) at \(t_n\) is \(p^{(v)}(s^{(v)}_{t_{n+1}} | s^{(v)}_{t_n}, a^{(v)}_{t_n})\). Specifically, \(a^{(v)}_{t_n}\) is the signal time \(g^{(v)}_{t_n}\) that Agent \(v\) chooses at time \(t_n\) for the next green signal. The state of intersection \(v\) is denoted as \(s^{(v)}_{t_n}\) where, \(s^{(v)}_{t_n} = \{s^{(v)}_{t_n, q_1}, s^{(v)}_{t_n, q_2}, \ldots, s^{(v)}_{t_n, q_i}, \ldots\}\). And \(s^{(v)}_{t_n, q_i}\) is the total number of vehicles in queue \(i\) at \(t_n\) and \(q_i \in \Omega_v\). The reward after taking \(a^{(v)}_{t_n}\) at state \(s^{(v)}_{t_n}\) at time \(t_n\) is

\[
r^{(v)}_{t_n} = \sum q_i (s^{(v)}_{t_n-1, q_i} - s^{(v)}_{t_n, q_i})
\]

It accounts for the change in the total number of vehicles in the queue before and after taking action. A positive reward means the queue size reduces; otherwise, it increases. Since the reward is the greater, the better, setting this reward function will push the agents to minimize the queue size. **If the number of partitions equals one**, then the problem becomes fully observable MARL, the state of which is written as \(s^{(v)}_{t_n} = \{s^{(1)}_{t_n, q_1}, s^{(1)}_{t_n, q_2}, \ldots, s^{(v)}_{t_n, q_1}, s^{(v)}_{t_n, q_2}, \ldots\}\), \(q_i \in \Omega_v, v \in V\) and the reward is written as

\[
r_{t_n} = \sum q_i \sum v \in V (s^{(v)}_{t_n-1, q_i} - s^{(v)}_{t_n, q_i})
\]

Now each agent considers the queuing status of the entire network instead of its queue, and the reward turns to the change of the queue size before and after the action. **If the number of partitions is greater than one and fewer than the number of intersections**, then the problem becomes partially observable cooperative MARL. Assume the number of partitions is \(n\), the status of agents in the partition \(G_k\) is, \(s^{(v)}_{t_n} = \)
\[
\{ s_1^{(t_n, q_1)}, s_2^{(t_n, q_2)}, \ldots, s_1^{(t_n, q_1)}, s_2^{(t_n, q_2)} \}, q_i \in \Omega, v \in V(G_k). \]

And the reward within the partition \( G_k \) can be expressed as

\[
r_{t_n} = \sum_{q_i \in \Omega} \sum_{v \in V(G_k)} (s_1^{(t_{n-1}, q_i)} - s_1^{(t_n, q_i)}) \tag{4.8}
\]

The queue status, actions, and rewards are defined according to the different modes mentioned above. At each decision point \( t_n \) (the time at \( n^{th} \) signal switch), an action \( a \) taken at state \( s \) leads to a new state \( s' \) with probability \( p(s'|s, a) \) and a reward \( r(s, a) \). A priori knowledge of the transition is required, and Deep Q Network (DQN), which samples from experience rather than prior knowledge, is used during its absence. The update rule was modified to consider samples of observed data, which tend to approach transition probabilities. Unlike Offline learning requiring numerous data simultaneously, DQN updates the model by continuously receiving states and rewards from the environment. The objective of DQN is to find an optimal action policy \( \emptyset^* \) that maximizes the expected cumulative future reward, namely \( Q\)-value:

\[
Q^{\emptyset^*}(s, a) = E\left[ \sum_{i=0}^{\infty} \gamma^i r_{t_n+i}^{(v)} | s_t^{(v)} = s, a_t^{(v)} = a, \emptyset^* \right] = \sum_{i=1}^{\infty} \gamma^{i-1} \sum_q (s_1^{(v)} - s_1^{(v)}) \tag{4.9}
\]

where \( \gamma \) is the discount factor (i.e., reward decay), usually between \([0, 1]\). When \( \gamma = 1 \), it equally treats future rewards, and \( \gamma = 0 \) takes only the current reward. In addition, the model uses the \( \epsilon \)-greedy policy to balance the exploration and exploitation during the learning process, where \( \epsilon \) is within \([0, 1]\). It usually starts at one and decays by \( \delta \) amount after every signal switch until it reaches \( \epsilon_{\text{min}} \). Definition of \( \epsilon(t_n) \) is as follows:

\[
\epsilon(t_n) = \max(1 - \delta t_n, \epsilon_{\text{min}}) \tag{4.10}
\]

At each decision point \( t_n \), the system uniformly generates a random value and compares it to \( \epsilon(t_n) \); If the value is greater, it exploits the system by taking the action that leads to the
greatest $Q$-$value$; otherwise, it randomly takes action to explore the system. It allows the system to try different actions before knowing the consequences and eventually mature to select the action that brings the highest reward. When optimal $Q^{\emptyset^*}(s, a)$ for all state-action pairs are obtained, the optimal policy $\emptyset^*$ for the state $s$ is simply the action $a$ that leads to the greatest $Q^{\emptyset^*}(s, a)$:

$$\emptyset^*(s) = \arg\max_a Q^{\emptyset^*}(s, a) \quad (4.11)$$

Therefore, a recursive relationship of $Q$-$value$ for Agent $v$ is found:

$$Q^{\emptyset^*}(s, a) = E\left[r_{tn}^{(v)} + \gamma \max_{a'} Q^{\emptyset^*}(s_{tn+1}^{(v)}, a') \mid s_{tn} = s, a_{tn} = a\right] \quad (4.12)$$

This equation indicates that each agent's optimal cumulative reward is equal to the immediate reward after taking action $a$ in traffic state $s$ plus the optimal future reward. When the optimal policy is known, the agents would know what action to take in terms of interest rate, and green signal duration is given any traffic state.

### 4.3.3 DQN Models and Algorithm

The model combines Double and Dueling DQN and Prioritized Experience Replay to boost the training process. The learning has three steps, and the training process is the same for each of the two agents. Figure 4.6 shows the detailed learning process.

![Figure 4.6. The structure of Double DQN](image-url)
In step 1, the current state and the tentative actions are cast into Main Network with parameter $\theta$ to choose the most rewarding action. Then, the system interacts with the simulation environment and feeds the old state $s$, action $a$, new state $s'$, and the reward $r$, into the memory as a form of four-tuple $(s, a, r, s')$ in step 2. The replay memory data are selected by the Prioritized Experience to generate mini-batches to update the primary neural network’s parameters. In step 3, Target Network with parameter $\theta'$ is introduced to interact with Main Network to update the loss function parameters.

The main DQN with parameter $\theta$ and Target DQN with parameter $\theta'$ have the same structure, but the parameters are updated at different rates. The update rate in Main DQN is designed to be faster than Target DQN to eliminate the maximization bias. According to (4.15), the agent tends to choose the action that leads to the highest $Q$-value. But how to make sure that the best action for the next state is the action with the highest $Q$-value? The accuracy of $Q$-value greatly depends on what action is tried and what neighboring states is explored. As a result, there is not enough information about the best action to take at the beginning of the training. Therefore, taking the maximum $Q$-value as the best action to take can lead to overestimation.

If non-optimal actions are regularly given a higher $Q$-value than the optimal best action, the learning will be biased. The solution is to use two networks, main DQN, and target DQN to decouple the action selected from the target $Q$ value generation [83-84]. The target network generates the target $Q$ value, and the action is generated from the main network. Similar to (4.12), the target $Q$ value for each agent is defined as,

$$Q_{target}(s, a) = r + \gamma Q_{target}(s', \text{arg max}_{a'}(Q_{main}(s', a'; \theta)), \theta') \quad (4.13)$$
where $\theta$ and $\theta'$ denote the parameters in the Main DQN and Target DQN, respectively. To update $\theta'$ in the Target Network, the function calculates the Temporal Difference (TD) Error between the Target Network and Main Network:

$$TD = |Q_{target}(s, a) - Q_{main}(s, a; \theta)|$$  \hspace{1cm} (4.14)

The Target Network is then updated by the Mean Square Error (MSE) of the TD error where Prioritized Experience Replay is introduced:

$$MSE = \sum_s P_s \cdot TD^2$$  \hspace{1cm} (4.15)

Here, $P_s = \frac{p^s}{\sum_k p^k}$ denotes the probability of getting state $s$ in the training mini-batch and $p_s = \frac{1}{\text{rank}(s)}$. Rank is the replay memory sorted by descending order of TD error, and $\alpha$ determines how much prioritization is used. When $\alpha$ is 0, it is uniform random sampling. When $\alpha$ is 1, it only selects the experiences with the highest priorities. Prioritization also introduces bias into the system but can be corrected by reducing the weights of the often-seen samples:

$$w_s = \left(\frac{1}{M} \cdot \frac{1}{P_s}\right)^\beta$$  \hspace{1cm} (4.16)

where $M$ is the memory size, $\beta$ is linearly annealed towards 1 through the training process. The role of $\beta$ is to control how much these importance sampling weights affect learning. In practice, when $\beta$ equal to 1, the important sampling weights fully compensate for the non-uniform $P_s$.

Since $Q$-value corresponds to how good it is to be at a state and taking action at that state, the Dueling DQN decomposed the $Q$-value as the sum of value at the current state $V(s)$ and each action’s advantage compared to other actions $A(s, a)$. The value of a state $V(s)$ denotes the overall expected rewards by taking probabilistic actions in the future steps. The
advantage corresponds to every action, which is defined by $A(s, a)$. Neural Network implicitly calculates both values. Figure 4.7 shows the structure of the Dueling DQN model for each agent. The states of the four queues are inputs for the network, and the outputs are action values. The length of the action array and advantage array depends on the size of the action space. As described previously, the number of inputs is the size of queues, and the number of outputs is the number of actions.

**Figure 4.7 The structure of Dueling DQN**

The pseudocode of the two-agent Double Dueling DQN with prioritized experience for agents in each partition $G_k$ is shown in Algorithm 4.1.

---

**Algorithm 4.1: Multi-agent DDDQN Algorithm**

Initialize replay memory and DQN network

1. While $episode \leq max\_episode$ do
2.     Initialize VISSIM, $T \leftarrow 40, t \leftarrow 0, n \leftarrow 0$
3.     While $t \leq max\_simulation\_time$ do
4.         $g \leftarrow 0$
5.         $s_{tn}^{(v)} \rightarrow a_{tn}^{(v)} \rightarrow g_{tn}^{(v)}$
6.         While $g \leq g_{tn}^{(v)}$ do
7.             $g \leftarrow g + 1, t \leftarrow t + 1$
8.             $(s_{tn}^{(v)}, a_{tn}^{(v)}) \rightarrow r_{tn}^{(v)}$
9.             $(s_{tn}^{(v)}, a_{tn}^{(v)}, r_{tn}^{(v)}, s_{tn+1}^{(v)}) \rightarrow memory$
10. If memory full
11.     Train MARL DDDQN $\rightarrow \theta^{(1)}, \theta^{(2)}$
12.     Update $\theta^{(1)}' \leftarrow \theta^{(1)}, \theta^{(2)}' \leftarrow \theta^{(2)}$ every $T$ iterations
13.     $\epsilon = max(1 - \delta, \epsilon_{min})$
14. $n \leftarrow n + 1$
**4.3.4 Distributed Multi-agent Reinforcement Learning with Graph Decomposition Approach**

As the size of the network grows, the training efforts and time increase exponentially. A distributed MARL training approach with graph decomposition is proposed to solve this issue for large-scale traffic networks, which is presented in Figure 4.8.

![Diagram](image)

**Figure 4.8 The general flow of the Distributed Multi-agent Reinforcement Learning with Graph Decomposition Approach**

The very first step for the proposed method is to calculate the average residual capacity (ARC). The ARC is defined by the total number of vehicles in the queue over the full queue capacity over a long horizon, obtained by either smart road sensors or simulations. The experiment starts with developing a simulation model, then applying actual signal configurations and running simulations to get the ARC. Figure 4.9 shows low, medium, and high sample residual capacities.
Figure 4.9 (a) Low residual capacity (b) Medium residual capacity (c) High residual capacity
A low average residual capacity case indicates severe congestions and delays. It also suggests that the signal configurations for the impacted intersections require continuous improvement. A medium residual capacity case shows that the current traffic condition is manageable, but it needs monitoring. A high residual capacity case indicates free flow in the traffic, and there is no action required at the moment. The level of connectivity (LoC) is introduced as the similarity measure between intersections for graph decomposition.

There are three levels of similarities: the traffic intersection \( a \) is strongly connected \((\tilde{s})\) to intersection \( b \) if the long term ARC of the link from \( a \) to \( b \) is low (e.g., \( \text{ARC} \in [0\%, 40\%] \)); if the long term ARC is medium (e.g., \( \text{ARC} \in (40\%, 70\%) \)), then intersection \( a \) is mediumly connected \((\tilde{m})\) to intersection \( b \); if the long term ARC is high (e.g., \( \text{ARC} \in [70\%, 100\%] \)), then intersection \( a \) is loosely connected \((\tilde{l})\) to intersection \( b \). The above definitions introduce the relationship between one-way streets.

Fuzzy rules are applied to elaborate on the relationship between two-way streets. Table 4.1 fuzzifies the relationship between intersections based on the connection level of one-way streets.

**Table 4.1 Level of Connection (LoC) for two-way streets**

<table>
<thead>
<tr>
<th></th>
<th>( s )</th>
<th>( m )</th>
<th>( l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>( s )</td>
<td>( s )</td>
<td>( l )</td>
</tr>
<tr>
<td>( m )</td>
<td>( m )</td>
<td>( m )</td>
<td>( l )</td>
</tr>
<tr>
<td>( l )</td>
<td>( m )</td>
<td>( l )</td>
<td>( l )</td>
</tr>
</tbody>
</table>

For example, if \( a \ \tilde{s} \ b \) and \( b \ \tilde{s} \ a \), then \( a \ and \ b \ \tilde{s} \). After building such a relationship, it is possible to group the connected intersections into strongly, mediumly, or loosely connected classes. It is noticed that only the same type of roadways (one-way or two-way) can be grouped together. If one intersection can be linked to more than one kind of group with the
same number of members, the following **ranking rules** apply: 1. the larger group has a higher priority than the smaller group; 2. two-way group has a higher priority than one-two group; 3. stronger connection has a higher priority than weaker connection. The algorithm for grouping intersections is shown in Algorithm 4.2.

**Algorithm 4.2: Graph decomposition**

Find LoC by ARC, create node set \( N \), edge list \( E \), and LoC final list \( \Omega \).

1. Initialize LoC lists \( \vec{S}, \vec{S}, \vec{M}, \vec{M}, \vec{L}, \vec{L} \)
2. For \( e \) in \( E \):
3. Classify the vertices \( \{n_1, n_2\} \) of \( e \) to \( S, S \in \{ \vec{S}, \vec{S}, \vec{M}, \vec{M}, \vec{L}, \vec{L} \} \)
4. For \( s \) in \( S \):
5. If \( s \cap \{n_1, n_2\} \neq \emptyset \):
6. \( s = s \cup \{n_1, n_2\} \)
7. Else:
8. \( S = S. append(\{n_1, n_2\}) \)
9. Rank all LoC sets according to the **ranking rules** in descending order and save to \( R \)
10. For \( r \) in \( R \):
11. If \( set(\Omega) \cap r = \emptyset \):
12. \( \Omega = \Omega. append(r) \)
13. \( N = N - set(\Omega) \)
14. Update \( E \)
15. Repeat until \( N = \emptyset \)

Below is an example that shows how to apply algorithm 4.2 on a sample graph with pre-calculated connection relationships and the steps to find the final groups.

![Figure 4.10 Graph representation of connection levels for a sample road network](image)

Using the LoC information on each edge and following the procedure below, the final group can be found.
The initial node set $N = \{a, b, c, d, e, f, g, h\}$.

**Iteration 1:**

**Step 1:**

$\bar{S} = [(f, g, h)], \bar{S} = [(a, f), (c, d)], \bar{M} = [(a, b, c)], \bar{M} = [(b, e)], \bar{L} = [(d, e, f)], \bar{L} = [(\emptyset)]$

**Step 2:**

$R = [(f, g, h), \{(a, b, c), (d, e, f), (a, f), (c, d), (b, e)]$

**Step 3:**

$\Omega = [(f, g, h), (a, b, c), (d, e, f, g, h)]$

**Step 4:**

$\Omega = [(f, g, h), (a, b, c), (d, e, f, g, h)]$

**Iteration 2:**

**Step 1:**

$\bar{S} = [(\emptyset)], \bar{S} = [(\emptyset)], \bar{M} = [(\emptyset)], \bar{M} = [(\emptyset)], \bar{L} = [(d, e)], \bar{L} = [(\emptyset)]$

**Step 2:**

$R = [(d, e)]$

**Step 3:**

$\Omega = [(f, g, h), (a, b, c), (d, e)], N = \{a, b, c, d, e, f, g, h\}$

**End**

The final groups are $\{f, g, h\}, \{a, b, c\}, \{d, e\}$. After applying the decomposition, subgraphs are trained by the distributed MARL training approach, as shown in Figure 4.11.

![Figure 4.11 Flowchart of the proposed synced MARL training approach](image)

The process starts with the initialization of the simulation model for all subgraphs. Each training uses the entire graph. The “smart” agents in the target subgraph are continuously trained and able to interact with “dumb” agents in the non-target subgraphs. Initially, “dumb” agents in non-target subgraphs are assigned with fixed signal timing and sequence. After a certain simulation step in the target subgraph, the “dumb” agents are replaced with smarter agents that are partially trained in other subgraphs. For instance, in simulation 1,
only the agents that belong to subgraph 1 are trained, and other agents are kept “dumb”. After certain steps, the dumb agents in subgraph 1 are replaced by partially trained smart agents in other simulations. The same process applies to all subgraphs simultaneously and repeats until reaching the stopping criteria.

4.4 Simulation Experiments

This section first examines the AEWWT-ATSC model's performance for a single intersection and compares different control methods at all densities. The work continues by showing the results of a two-agent DQN model for a road network with two-intersections. The last part demonstrates the implementation of DMARL on clustered intersections in New York City.

4.4.1 Comparison of Different Control Policies for A Single Intersection

This experiment compares the results of four control methods to investigate the AEWWT-ATSC model's performance: 1) RL-based AEWWT-ATSC model with dynamic signal time; 2) RL-based Null model (no exponential term) with dynamic signal time; 3) fixed signal time and sequence; 4) AEWWT-ATSC model with fixed signal time. Each method compares experiment results in High traffic, Low traffic, and Mixed. In I and II, signal times are subject to change, and the sequence is decided by (5). I only consider the number of vehicles in the queue while capturing both queue size and queue time. To train the agent for signal time selection, the DDDQN model is applied. Table 4.2 lists some common practices of DDDQN training parameters in the study [86-87]. The third method adopts fixed signal time (east/west 30s per cycle, north/south 60s per cycle, which is proportional to the average traffic volume) and cyclically switches the signal
The last method takes fixed signal time, but the signal’s switch is based on (5).

TABLE 4.2 Parameters Used in the Training

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Policy 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of trials</td>
<td>150</td>
</tr>
<tr>
<td>Maximum simulation seconds</td>
<td>10,000</td>
</tr>
<tr>
<td>Main-net update frequency</td>
<td>40</td>
</tr>
<tr>
<td>Greedy decrement</td>
<td>0.008</td>
</tr>
<tr>
<td>Min greedy</td>
<td>0.02</td>
</tr>
<tr>
<td>Reward decay</td>
<td>0.95</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.005</td>
</tr>
</tbody>
</table>

Figure 4.12 plots the time-series reward, the total number of vehicles in the queue, and the total accumulated waiting time of each control method in Low, High, and Mixed traffic densities. Table 4.3 compares the average total number of vehicles in the queue and the average total accumulated waiting time and calculates the percentage of reduction to the fixed timing in the converging trials.
Figure 4.12 The training result of RL-based AEWWT-ATSC model, RL-based Null model, AEWWT-ATSC model with fixed signal times, and the model with fixed time and sequence, in (a) Low traffic flow; (b) High traffic flow; (c) Mixed traffic flow.
### TABLE 4.3 Comparison of different control policies in Low, Mix, and High traffic volumes

<table>
<thead>
<tr>
<th>Traffic volume</th>
<th>Control policy</th>
<th>Avg # of vehicles in the queue</th>
<th>Improvement in queue size</th>
<th>Avg waiting time (s)</th>
<th>Improvement in queue time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>RL+ AEWWT-ATSC</td>
<td>16.66</td>
<td>-24%</td>
<td>674</td>
<td>-40%</td>
</tr>
<tr>
<td></td>
<td>RL+NULL</td>
<td>19.48</td>
<td>-12%</td>
<td>872</td>
<td>-22%</td>
</tr>
<tr>
<td></td>
<td>FixTime+AEWWT-ATSC</td>
<td>20.14</td>
<td>-9%</td>
<td>888</td>
<td>-21%</td>
</tr>
<tr>
<td></td>
<td>Fix</td>
<td>22.06</td>
<td>0%</td>
<td>1,122</td>
<td>0%</td>
</tr>
<tr>
<td>High</td>
<td>RL+ AEWWT-ATSC</td>
<td>294.85</td>
<td>-10%</td>
<td>112,894</td>
<td>-30%</td>
</tr>
<tr>
<td></td>
<td>RL+NULL</td>
<td>309.26</td>
<td>-5%</td>
<td>132,888</td>
<td>-18%</td>
</tr>
<tr>
<td></td>
<td>FixTime+AEWWT-ATSC</td>
<td>322.37</td>
<td>-1%</td>
<td>135,276</td>
<td>-16%</td>
</tr>
<tr>
<td></td>
<td>Fix</td>
<td>327.01</td>
<td>0%</td>
<td>161,332</td>
<td>0%</td>
</tr>
<tr>
<td>Mix</td>
<td>RL+ AEWWT-ATSC</td>
<td>215.06</td>
<td>-27%</td>
<td>64,615</td>
<td>-54%</td>
</tr>
<tr>
<td></td>
<td>RL+NULL</td>
<td>242.68</td>
<td>-17%</td>
<td>83,830</td>
<td>-41%</td>
</tr>
<tr>
<td></td>
<td>FixTime+AEWWT-ATSC</td>
<td>267.29</td>
<td>-9%</td>
<td>95,631</td>
<td>-32%</td>
</tr>
<tr>
<td></td>
<td>Fix</td>
<td>294.03</td>
<td>0%</td>
<td>141,577</td>
<td>0%</td>
</tr>
</tbody>
</table>

Each dot represents the performance of a trial. All policies are run five replications to produce statistically significant results. Both mean and standard deviation are plotted in the graph. In any traffic density, the RL-based methods outperform others in terms of average queue size and waiting time, and exponentially accumulated waiting time positively impacts performance in either fixed signal time scenario or dynamic signal time scenario. In the RL-based methods, the performance initially fluctuates since the agent randomly samples actions to explore the reward, queue size, and waiting time it receives. The agents gradually gather enough experience to take reasonable actions for higher rewards, smaller
queue sizes, and shorter waiting times. This experiment's results show the RL-based AEWWT-ATSC model's superiority and lay a foundation for the following study.

4.4.2 Comparison of Competitive and Cooperative MARL for Two Intersections

For a road network with more than one intersection, Multi-agent Reinforcement Learning (MARL) methods kick in. In this experiment, the performance of a fully competitive and cooperative MARL is tested. Each agent adopts an RL-based AEWWT-ATSC model. Similar to the settings in a single intersection VISSIM model, each scenario is run five replications. Figure 4.13 plots the time-series reward, the total number of vehicles, and the total accumulated waiting time of Cooperative and Competitive MARL in Low, High, and Mixed traffic flows. Table 4.4 compares the average total number of vehicles in the queue and the average total accumulated waiting time in the converging trials and calculates the percentage of reduction to the competitive MARL in the converging trials.

<table>
<thead>
<tr>
<th>Traffic volume</th>
<th>Control policy</th>
<th>Avg # of vehicles in the queue</th>
<th>Improvement in queue size</th>
<th>Avg waiting time (s)</th>
<th>Improvement in queue time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Cooperative</td>
<td>16.66</td>
<td>-44%</td>
<td>674</td>
<td>-57%</td>
</tr>
<tr>
<td></td>
<td>Competitive</td>
<td>29.49</td>
<td>0%</td>
<td>1,575</td>
<td>0%</td>
</tr>
<tr>
<td>High</td>
<td>Cooperative</td>
<td>423.19</td>
<td>-13%</td>
<td>146,842</td>
<td>-17%</td>
</tr>
<tr>
<td></td>
<td>Competitive</td>
<td>487.21</td>
<td>0%</td>
<td>176,300</td>
<td>0%</td>
</tr>
<tr>
<td>Mix</td>
<td>Cooperative</td>
<td>116.72</td>
<td>-35%</td>
<td>12,781</td>
<td>-56%</td>
</tr>
<tr>
<td></td>
<td>Competitive</td>
<td>179.71</td>
<td>0%</td>
<td>29,133</td>
<td>0%</td>
</tr>
</tbody>
</table>
Both types of MARL can improve network performance in terms of average queue size and waiting time, but cooperative MARL can do a better job reducing the number of vehicles in the queue and the total waiting time. Moreover, competitive MARL produces more variation, which presents a more unstable performance over cooperative MARL. The advantage of using competitive MARL is that each agent needs to focus on its intersection, thus keeping a compact state space.

### 4.4.3 Case Study of Using DMARL for A Large Real Network

The example consists of 18 intersections in Manhattan, NYC. The experiment starts with building a simulation model of the target area to get the average residual capacity (ARC) using the current signal configurations. Traffic information is provided by the NYS Traffic Data Viewer [99]. Eighteen intersections are grouped into three subgraphs according to algorithm 4.2, as shown in Figure 4.14 (b). It compares with the randomly grouped subgraphs in Figure 4.14 (a) to display the grouping policy's superiority.
The training procedure follows Figure 4.11. Figure 4.15 shows the performance of each subgraph defined in Figure 4.14. To compare different training methods: random group + MARL, LoC group + MARL, and LoC group + DMARL, Table 4.5 lists the average total number of vehicles in the queue and the average total accumulated waiting time in the
converging trials and calculate the percentage of reduction to the Random graph decomposition in the converging trials.

**TABLE 4.5 Comparison of different training approaches**

<table>
<thead>
<tr>
<th>Graph decomposition</th>
<th>Training approach</th>
<th>Avg # of vehicles in the queue</th>
<th>Improvement in queue size</th>
<th>Avg waiting time (s)</th>
<th>Improvement in queue time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>MARL</td>
<td>254.44</td>
<td>0%</td>
<td>37847</td>
<td>0%</td>
</tr>
<tr>
<td>LoC</td>
<td>MARL</td>
<td>228.95</td>
<td>-10%</td>
<td>32,631</td>
<td>-14%</td>
</tr>
<tr>
<td>LoC</td>
<td>DMARL</td>
<td>197.50</td>
<td>-22%</td>
<td>27,440</td>
<td>-27%</td>
</tr>
</tbody>
</table>

Random group + MARL delivers the worst performance among the three approaches, and LoC + DMARL shows the best performance, which results in a 22% reduction in queue size and 27% reduction in queue time.

**4.5 Conclusion**

This paper proposed the Accumulated Exponentially Weighted Waiting Time-based Adaptive Traffic Signal Control (AEWWT-ATSC) model for traffic signal control at connected intersections. The AEWWT-ATSC model takes the exponentially weighted waiting time of each vehicle at the intersection to generate priorities as the control references for signal sequencing and timing. In a dynamic traffic study where signal times are not fixed, each intersection controller is assumed to be an intelligent agent seeking to minimize the traffic delay and the number of vehicles in the queue by selecting a proper green signal duration in each cycle. The control problem is formulated as a Markov Decision Process (MDP), and Double Dueling DQN with Prioritized Experience Replay is utilized to find the optimal solution. A well-trained agent adopts optimal control policies for signal times, according to the current traffic patterns. Under the optimal policy, the traffic delay and the queue size are minimized. To show the superiority of the method, the
study first compares the AEWWT-ATSC model’s performance with other control policies in different traffic densities for a single intersection. A network with more than one intersection requires Multi-agent Reinforcement Learning (MARL). In such a network with two intersections, the comparison is made on competitive MARL performance and fully observable cooperative MARL. The results indicate that fully observable cooperative MARL outperforms the competitive MARL. It is essential for large graphs to apply the decomposition method and distributed MARL (DMARL) approach. The approach clusters intersections into subgraphs and trains each subgraph in a synchronized way. This result suggests that the approach is generic and can be used for various types of intersections.
5. DATA-DRIVEN OPTIMIZATION FOR DYNAMIC SHORTEST PATH PROBLEM WITH TIME-VARYING TRAVEL TIME AND TRAFFIC SAFETY FOR URBAN NAVIGATION

5.1 Introduction

Traffic congestion has become a severe problem delaying the traffic and frustrating the drivers [100-101]. The urban navigation system is widely used to alleviate traffic congestion in megacities with dynamic shortest path problems (DSPP) in its core. The DSPP aims at finding the best route with the shortest distance or travel time, where the traffic information (e.g., travel time) changes over time and cannot be accessed in advance [102-103]. The challenges of solving the DSPP for urban navigation mainly lie in the following two aspects: (a) The optimal route at the beginning of a trip might not be optimal in the middle of the trip. It is ideal to foresee such variations before departure. Taking appropriate traffic information, the model can predict future travel time on each road segment, considerably affecting the quality of navigation solutions [104]. Some scholars only use traffic information (e.g., speed and travel time) at departure to find the shortest path [105]. However, since the traffic information is not static, the solution obtained in such a way may be unreasonable. Figure 5.1 gives an example to illustrate the impact of the time-varying information on routing decisions. (b) The classic Double Search algorithm (DSA) may not be ideal for all sizes of networks. An improved DSA is required to solve large-scale problems for urban navigations without losing much quality of the solution [106].
Figure 5.1 Impact of Time-varying Traffic Information on Routing Decision

The solid and dotted lines represent two travel paths directing node A (origin) to node D (destination). Green means no congestion, and red means the opposite. Travel time is provided on each edge. Suppose it is 8:00 am in Figure 5.1 (a), according to the current traffic information, the solid route's total travel time is 31 minutes, and the dotted route is 41 minutes. Therefore, the solid line is selected when only the current information is considered for computing the shortest path. It is also assumed that two identical vehicles, vehicles 1 and vehicles 2, leaving node A simultaneously. Vehicle 1 travels along the dotted route, and vehicle 2 travels along the solid route. As shown in Figure 5.1 (b), at 8:25 am, vehicle 2 reaches node C and vehicle 1 arrives at node E. The observed travel times C to D and E to D are 15 and 10 minutes. It indicates that the edge (C-D) becomes congested, and the edge (E-D) is now free from congestion. Consequently, the total travel times of vehicle 1 and vehicle 2 are 35 minutes and 40 minutes, respectively. It is noted that a path with a shorter travel time at departure may not guarantee an earlier arrival at the destination. This example demonstrates that routing decisions made by current traffic information may not yield the best result since the traffic conditions are time-varying during the travel. If
both current traffic information and future information can be considered in the DSPP, a better navigation solution may be obtained [107].

In practice, one strategy for overcoming the drawback of only using the current traffic information is to update the existing navigation solution while traveling continuously. The navigation system repeatedly re-optimizes the solution using “new current information” obtained at different time epochs during the travel. Once a new and better solution is received, it sends the new recommended path to the driver. It is called the “re-optimization strategy.” It requires intensive information and computation during the travel and does not consider future traffic conditions at any time epoch. Therefore, though this strategy might outperform the approach that only uses the traffic information at departure, its computational performance and uncertainty in travel plans could make it impractical.

In this Chapter, a special DSSP in a time-varying urban traffic network is investigated. Given a pair of origin and destination points (OD pair), the model determines the route with time-varying traffic information using data-driven methods. The traffic information contains not the only travel time. There can also be some other key elements in urban navigation, such as on-time arrival probability, safety, fuel consumption, and emission [108-109], which can be regarded as the time-varying resources in the DSSP model [110-111]. Conclusively, the problem is a dynamic shortest path problem with time-varying travel time and resources.

The main contributions of the research are as follows: First, a data-driven optimization method, where both current and future forecasts of traffic information are used, is designed. This will enhance routing decisions as the traffic data at departure point changes due to time-varying traffic conditions. Second, in addition to traditional resources such as fuel
consumption, safety factors are considered in routing decisions by taking advantage of the Safe Route Mapping (SRM) introduced earlier. The SRM integrates crash count estimates with risk probability computed from driver-based data to score roadways' safety. By running the SRM methodology, road safety constraints are introduced based on the calculated risk scores. Lastly, the study proposes an effective algorithm to solve the DDSP problem by modifying the classical tabu search algorithm [113] using various initial solution generation techniques and acceleration strategies, to obtain a high-quality solution for large-scale problems efficiently.

The remainder of this chapter is organized as follows. It starts with a review of the related research work in Section 5.2. It continues the study by formulating the DSPP as a mixed-integer linear programming problem in Section 5.3 and introducing the solution approach in Section 5.4. In Section 5.5, numerical results are presented to demonstrate the algorithm's computational performance and validate the contributions. Finally, section 5.6 concludes the paper and discusses future work.

5.2 Literature Review

In the DSPP, traffic information such as travel time and resources are time-varying. Hence, selecting the proper data of travel time for the navigation algorithm gradually becomes a critical issue [114, 115]. Most research focuses only on current information, and some papers integrate the current information with historical information.

With only current traffic information, the shortest path and some alternative paths in response to sudden events are determined in Faro A’s research [116]. A set of efficient algorithms for navigation are developed in both free-flow and congested scenarios. Ardakani M K and Sun L [117] investigate a DSPP in the continuous-time network and
develop an algorithm to find the best route with real-time link travel time. Fu L and Rilett
L R [118] propose a heuristic algorithm based on the k-shortest path algorithm to tackle
the shortest path problem in a dynamic and stochastic environment. Sever D et al. [119]
develop efficient hybrid routing policies to solve the DSPP in a traffic network with
disruptions due to accidents, bad weather and traffic congestion. They believe that although
navigation with real-time data yields good solutions, it results in higher computation time
and information retrieval cost. Davies C and Lingras P [120] study the problem of
generating the shortest path in a dynamic traffic network, where the travel time on each arc
changes as yet-to-known functions of time. In Güner AR’s research, congestion states are
utilized to describe travel time at different times and in various external conditions [121].
The congestion states are classified into recurrent congestion caused by regular traffic
flows such as peak hours and non-recurrent congestion caused by accidents and temporary
policies. Huang H and Gao S [122] suggest that travel time is correlated with time and
space in the traffic network. Considering the correlation of link travel time, they design an
exact label-correcting algorithm to obtain the best path with a new property where
Bellman’s Principle holds.

Besides, some researchers utilize not only real-time data but historical data as well. Liu
Ruilin et al. [123] develop an approach to provide another route to coordinate the traffic
volume to alleviate the congestion by rating the traffic jam according to historical data of
evolving congestion, apart from the conventional travel path derived by current real-time
information. But most people find one optimal path rather than various alternatives. Lim S
[124] develops a stochastic path planning algorithm that incorporates current information
into historical information. Historical information is substituted with the current
information, i.e., the mean travel time is replaced with current observed travel time for road segments with updated information for a predefined time window. This method propagates the current information as far as the road segments are statistically related. However, integrating the two types of information in their paper might not be suitable for urban navigation. First, the observed value could not represent a general case since it might happen with a small probability. Second, replacing the mean travel time with the time-varying value leads to a frequently changed distribution, resulting in unreasonable routing decisions. Lastly, their approach is challenging to implement due to long computational time. Some research devotes to the accurate prediction of the evolving traffic conditions, which integrates the two types of information, mainly performing spatial-temporal random field, gaussian process regression, and congestion immigration flow[125-127]. These prediction methods can result in a highly complicated DSPP model. Thus it might be difficult to design an algorithm to obtain an effective navigation solution efficiently with high accuracy.

Apart from travel time, resources such as safety and fuel consumption also play an essential role in urban navigation. Note that the resources are treated as static parameters in most research works [128-129]. Nie Y and Wu X [130] propose a method that finds a priori shortest path while ensuring a given probability of on-time arrival. Optimal solutions are obtained from local-reliable paths, a set of non-dominated paths under first-order stochastic dominance. Xiao L and Lo HK [131] develop an adaptive navigation approach for risk-averse travelers considering the reliability of on-time arrival. It optimizes the expected prospect of potential route alternatives and ensures that both on-time arrival reliability and expected travel time are acceptable. In Galbrun E’s research, a routing model considering
driving risk has been developed to generate a safe path [132]. The risk mainly comes from possible accidents caused by casual factors, similar to the study's driving risk. Bae K Y et al. [133] investigate a risk-constrained shortest path problem for a combat vehicle. It aims to minimize the total travel time with a limitation on the sum of risk level values. The risk refers to the probability of being found or attacked in a military area, treated as a static resource. Both exact algorithms based on dynamic programming and heuristics are designed for this problem. Wang L et al. [105] consider a resource-constrained shortest path problem with time-varying and correlated link travel times. The resources are fuel consumption on each road and do not vary with time. They use a Lagrangian Relaxation-based heuristic to solve the problem. Strehler M et al. [134] formulates the shortest path problem with charging stations for electric and hybrid vehicles. An efficient algorithm is presented to obtain the optimal path and reduce the energy and fuel consumption treated as static resources. However, these key factors considered in urban navigation are mostly assumed to be static in most existing studies. Although dynamic resources are also taken into account sometimes, the problem differs from the research. Li W et al. [135] address a route guidance problem to find an eco-reliable travel path by the Lagrangian Relaxation approach in a time-varying traffic network, where the emission is regarded as dynamic resource. The authors aim to minimize the late arrival probability defined with a time threshold to specify the on-time arrival timestamp. Nevertheless, the objective function is not the same as ours. Moreover, they only use the historical information, with little consideration of the impact of time-varying traffic information on routing decision.

Although the DSPP for urban navigation has been intensively studied, this research differs from the previous research in two ways: (a) The impact of time-varying traffic information
on routing decisions is rarely considered. Although the current and historical information has been integrated into some research, using a data-driven method to acquire traffic information for DSPP is new; (b) Most scholars assume resources to be static when developing navigation algorithms. This assumption does not hold since these resources vary with time and external conditions (e.g., weather). In contrast, this research can yield more reasonable routing decisions by introducing dynamic resources.

### 5.3 Methodology and Problem Statement

A data-driven optimization method is defined for DSSP with consideration of traffic safety for urban navigation. The dynamic risk score and travel time on different road facilities (e.g., intersection, segment, ramps) at a different time are estimated by the Safe Route Mapping (SRM) methodology [112] and Long Short Term Memory (LSTM) with Autoencoder [136], respectively. Figure 5.2 shows the structure of the proposed data-driven optimization method.

![Figure 5.2 The structure of the proposed data-driven optimization method for DSPP](image)

Figure 5.2 The structure of the proposed data-driven optimization method for DSPP
The SRM method generates crash count estimates, conflict probability, and risk score by the Advanced SPF model, the Risk Prediction Model, and the Risk Integration Model, respectively. The prediction of travel time is done by the LSTM Autoencoder model [136], which considers dynamic impacts such as date, time, and weather. Finally, it takes the projected risk scores and travel times to make the routing decision by the DSPP solver at departure.

5.3.1 Risk Estimation

The risk measure is computed by the SRM methodology [112], introduced in chapter 3. The SRM takes multiple data sources such as crash data, road features, Geographic data, and information (GIS) to forecast the future risks on roadways.

5.3.2 Travel Time Prediction

As travel times are time-series data, it is ideal to utilize time-series tools to forecast future travel time on different road segments at departure as a reference for route planning. Long Short-Term Memory (LSTM) with Autoencoder is an efficient type of recurrent neural network (RNN) that can learn the sequence and oscillation behaviors such as seasonality and trend of sequential data. And Autoencoder can make the computation more efficient by reducing the dimensions into some representation nodes [136]. The LSTM model will then learn the reduced representations instead of the whole network. Since it maps a sequence of past observations as input to n-step ahead observation, the observations must be first transformed into samples from which the LSTM can learn [136]. The study divides the sequence into multiple input/output samples by a fixed window.
The figure demonstrates an example using three-step to forecast one-step ahead result, generating n-4 training samples and one testing sample out of n time steps. In actual training, the model can adjust the number of input steps and training epochs to find the optimal combination that leads to minimal testing errors for all road edges at the same time.

The weighted mean absolute percent error (WMAPE) \([137]\) is used as the loss function to evaluate the training and testing:

\[
WMAPE = 100 \times \frac{\sum_{i=1}^{m} (True_i - Forecast_i)}{\sum_{i=1}^{m} True_i}
\]  

(5.1)

where \(True_i\) is the historical travel time in the testing time step, \(Forecast_i\) is the estimate, and \(m\) is the number of edges. The goodness of WMAPE is that it can avoid infinite error in MAPE when the actual value is zero.

### 5.3.3 Shortest Path Formulation

The problem is defined in a directed graph \(G = (V, A)\). Set \(V = \{o\} \cup N \cup \{d\}\) refers to the set of nodes, where the origin is denoted by \(o\), the destination is denoted by \(d\), and the
rest nodes by \( N. A = \{(i, j): i, j \in V, i \neq j\} \) represents the set of edges. Let \( D_{ij} \) be the distance of edge \((i, j)\). The travel times and resources are dependent on discrete-time epochs \( t=0, 1, \ldots, T \), where \( T \) is the length of the planning horizon. \( T_{ijt} \) represents the travel time of leaving node \( i \) at time \( t \) along the edge \((i, j)\). \( T'_{ijt} \) denotes the travel time of reaching node \( j \) at time \( t \) along the edge \((i, j)\). Let \( K \) be the resource set, where each \( k \in K \) refers to one resource. \( R_{kijt} \) represents the consumption of resource \( k \), leaving node \( i \) at time \( t \) along the edge \((i, j)\). \( S \) denotes the maximum allowance for the average risk score (0-100). It is essential to introduce the decision variables,

\[
X_{ijt} = \begin{cases} 
1 & \text{if leaving for node } j \text{ from node } i \text{ at time } t \\
0 & \text{otherwise}
\end{cases}
\]  

The aim is to minimize the total cost from the origin to the destination. The total cost is formed by a weighted sum of total distance and total travel time, with the weight coefficient \( \alpha \in [0, 1] \). Consequently, the problem can be formulated as a mixed-integer linear programming model.

\[
\text{Minimize } \alpha \sum_{i \in V} \sum_{j \in V} D_{ij} \sum_{t \in T} X_{ijt} + (1 - \alpha) \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} T_{ijt} X_{ijt} 
\]

\[
s.t. \quad \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} X_{ijt} R_{kijt} \leq W_k \quad \forall k \in K \quad (5.4)
\]

\[
\sum_{i \in V} \sum_{j \in V} \sum_{t \in T} X_{ijt} S_{ijt} / \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} X_{ijt} \leq S \quad (5.5)
\]

\[
\sum_{j \in V} X_{ijt} = \sum_{j \in V} X_{ji(t-T'_{ijt})} \quad \forall i \in N, t \in T \quad (5.6)
\]

\[
\sum_{j \in V} X_{ojo} - \sum_{j \in V} X_{joo} = 1 \quad (5.7)
\]

\[
\sum_{j \in V} \sum_{t \in T} X_{adj} - \sum_{j \in V} \sum_{t \in T} X_{jdt} = -1 \quad (5.8)
\]
The objective function (5.3) minimizes the total travel cost. Constraints on each resource are imposed through constraint (5.4). Constraint (5.5) defines the maximum allowance for average risk score. Constraint (5.6) ensures the flow balance for any node except for the origin and destination. Constraint (5.7) and (5.8) indicate that the travel path starts at the origin and ends at the destination.

When trying to solve the above model to obtain a travel path, three input parameters, $T_{ijt}$, $T'_{ijt}$, $S$, and $R_{kij}$, affect the solution quality significantly. Using different input data to initialize such parameters leads to different solutions. Generally speaking, most current navigation software employs current information, i.e., the information exactly at departure to initialize the parameters. However, as stated in Section 1, such information adaption is unreasonable. A different strategy is proposed in the research: the data to infer these parameters is acquired from current and historical traffic information. For an edge near the origin, the chance to meet drastic travel time changes and resource consumption is relatively low. Considering current traffic information is capable of accurately reflecting the actual traffic conditions. Thus, it is ideal to employ current traffic information to characterize such edges. It is better to change the focus to estimating travel time and resource consumption for an edge far from the origin rather than its current status when leaving the origin. It is suggested to pull the associated data from historical traffic data and then using the time-series predictive model to calculate the predicted travel time and resource consumption at a particular time epoch.

For simplicity, the current traffic information is adopted on the edge directly connected to the origin, and the rest of the edges will use predicted traffic information. Suppose it is 8:00 am on Monday, and the total travel time for route A-B-C is calculated. Based on the
current traffic information at 8:00 am, the travel time along the edge (A, B) is 1 hour. Thus, the expected time to arrive at node B is 9:00 am. Since edge (B, C) is not directly connected to node A, historical traffic information is utilized to estimate the travel time for edge (B, C). Suppose the estimated time to reach node C from node B is 0.5 hours, and the expected time arriving at node C is 9:30 am.

5.4 DDSP Solution approach

A modified tabu search algorithm is designed to handle the large-scale problem for real-world applications, as shown in Figure 5.5.

![Figure 5.5 Algorithm Framework](image-url)
It starts with an initial solution and then explores the solution space by moving from the current solution \( s \) to the best solution in its neighborhood \( N(s) \) at each iteration. To avoid cycling and diversify the solution space, some recent moves are declared tabu for several iterations unless they are better than the best feasible solution found in the searching procedure. The algorithm terminates when the number of iterations has reached the upper limit \( UL \), and the best travel path is obtained.

### 5.4.1 Initial Solution

Three algorithms, Label-Setting Algorithm (LSA), Dijkstra’s Algorithm (Dijkstra), and Double Search Algorithm (DSA), can be utilized to generate the initial solution. Algorithm selection depends on the problem scale and efficiency requirement.

#### 5.4.1.1 A Label-Setting Algorithm

The initial solution could be derived by solving a static resource-constrained shortest path problem. The model considers the current traffic information at departure to obtain the travel time and resource consumption on each edge. Then a classical label-setting algorithm is carried out for the initial solution.

A subset of \( K, K' \) is defined to denote the simplified case when not using all the initial solution resources. Given arbitrary node \( i \), a vector \( R_i^{hK'} = (R_i^{h1}, R_i^{h2}, \ldots, R_i^{h|K'|}) \) is defined to represent the consumption of each resource of the \( h \)th path from the origin to \( i \). The \( h \)th path from the origin to node \( i \) is associated with a label (resource, cost) at node \( i \), characterized by \( (R_i^{hK'}, C_i^h) \) \( i \in V, h \geq 1 \). Let \( (R_i^{1K'}, C_i^1) \) and \( (R_i^{2K'}, C_i^2) \) be two distinct labels for two paths from the origin to node \( i \). The first label dominates the second if and
only if $C_i^2 \geq C_i^1, R_i^{2k} \geq R_i^{1k}$ for all $k \in K'$. A label $(R_i^{hK'}, C_i^h)$ at node $i$ is efficient if no other labels at node $i$ dominates it.

Then, two basic operations $\text{EFF}$ and $f_{ij}$. $\text{EFF}$ are introduced to drop the dominated and infeasible labels in a given set to select useful labels (efficient labels). $f_{ij}$ derives a label of node $j$ from node $i$. Based on the two operations, the treatment of the label $(R_i^{hK'}, C_i^h)$ can be defined. Let $T_j$ be the set of labels of node $j$ connected to node $i$. The efficient labels of node $j$ are updated by $T_j = \text{EFF}(f_{ij}(R_i^{hK'}, C_i^h) \cup T_j)$. Similarly, the efficient labels of each node connected to node $i$ for the treatment of $(R_i^{hK'}, C_i^h)$ are updated. A set $P_i$ is introduced to include the labels of node $i$ that have already been treated. The LSA consists of the following steps.

\begin{itemize}
  \item \textbf{Step 1. Initialization:}
    \begin{itemize}
      \item Node 0 is the origin.
      \item $T_0 = \{ (R_0^{hK'}, C_0^h) = (0,0) \}; \quad T_i = \emptyset, \forall i \in N \cup \{d\};$
      \item $P_i = \emptyset, \forall i \in V$
    \end{itemize}
  \item \textbf{Step 2. Selection of the next label to be treated:}
    \begin{itemize}
      \item Choose a label $(R_i^{hK'}, C_i^h)$ with minimal $C_i^h$ from $\cup_{i \in V} (T_i \setminus P_i)$. $i_0$ denotes the node associated with the selected label.
      \item If $\cup_{i \in V} (T_i \setminus P_i) = \emptyset$ or the selected label is associated with the destination, then STOP.
    \end{itemize}
  \item \textbf{Step 3. Treatment of label $(R_i^{hK'}, C_i^h)$:}
    \begin{itemize}
      \item For all $j$ connected to $i_0$, $T_j = \text{EFF}(f_{i_0j}(R_i^{hK'}, C_i^h) \cup T_j)$;
      \item $P_i = P_i \cup \{ (R_i^{hK'}, C_i^h) \}$;
      \item Return to Step 2.
    \end{itemize}
\end{itemize}

5.4.1.2 \textit{Dijkstra’s Algorithm}

The resource constraints are relaxed in the problem defined in Section 4.1.1, leading to a static shortest path problem. Moreover, the travel cost on each edge at $t=0$ is obtained. This problem can be quickly solved by Dijkstra’s Algorithm [138].
5.4.1.3 *Modified Double Search Algorithm*

The Double Search Algorithm proposed in O. Dib et al.’s research [139] is modified to find a path connecting the origin to the destination. The cost and resources are temporarily ignored. It performs a forward search process (FSP) from the origin and a backward search process (BSP) from the destination. Besides, a forward-to-backward (F2B) queue for FSP and a backward-to-forward (B2F) queue for BSP are defined. The origin is added into the F2B queue, and the destination is added into the B2F queue. The algorithm starts with finding the nodes connected to the F2B queue's head node before adding them to the F2B queue. Then the head node is removed from its queue. Similar operations are implemented for the B2F queue by the BSP. FSP and BSP have performed alternately. To keep track of the two search processes, each node is assigned with a flag to indicate which process it is visited. Once a node has been visited by both FSP and BSP, a path from the origin to destination is detected. Figure 5.6 demonstrates the Double Search Algorithm.

![Double Search Algorithm Diagram](image)

*Figure 5.6 Demonstration for Double Search Algorithm*
The purpose is to find intermediate paths from origin A to destination F. Node B and node C are connected to node A, node E and node G are connected to node F, and node B and node E are linked. The Double Search Algorithm first adds node A and node F into the F2B queue and B2F queue. Then, nodes B and C are pushed into the F2B queue, and nodes G and E are pushed into the B2F queue. After the head node is removed from its queue, the algorithm repeats the FSP at node B, and FSP visits node E. Since both FSP and BSP visit node E, the initial travel path is found.

5.4.2 Penalized Objective Function

For a broad exploration of the solution space, infeasible solutions are allowed in the search procedure, which implies that the resource constraints might be violated. The resource constraints are relaxed and incorporated into the objective function with penalty parameters. Therefore, it is reasonable to build a penalized objective function. For a solution $s$, let $TC(s)$ denote the total travel cost, $TR_k(s)$ represent the total violation of the constraint for resource $k$. The solution will be evaluated by the function $f(s) = TC(s) + \sum_k b_k TR_k(s)$, where

$$TC(s) = \alpha \sum_{i \in V} \sum_{j \in V} D_{ij} \sum_{t \in T} X_{ijt} + (1 - \alpha) \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} T_{ijt} X_{ijt}$$ (5.9)

$$TR_k(s) = \max\left( \sum_{i \in V} \sum_{j \in V} \sum_{t \in T} X_{ijt} R_{kijt} - W_k, 0 \right)$$ (5.10)

Each $b_k$ is a positive parameter indicating the penalty of unit constraint violation for resource $k$ and is dynamically adjusted in the search procedure. At each iteration, each $b_k$ is modified by a factor $1+\beta$ where $\beta > 0$. If the current solution is feasible, each $b_k$ is divided by $1+\beta$, otherwise multiplied by $1+\beta$. 
5.4.3 Neighborhood structures

Two local moves derive the neighborhood of the current solution. The first move is to remove one node from the current path. The second move is to insert one node into the current path. For each node $i$ in the current path except for the destination, and the algorithm selects some nodes that have not been visited by the path directly connected to node $i$ and attempt to insert them after node $i$. The move leading to the lowest penalized objective value produces the best neighboring solution.

5.4.4 Tabu List and Aspiration criterion

To prevent cycling and avoid local optimum, some recent moves are banned within several iterations denoted as $\theta$. They are stored in the tabu list for short-term memory. If one node is removed from the current path, it can be inserted into another path after $\theta$ iterations. If one node is already inserted into a path, it is prohibited to remove this node for the next $\theta$ iterations. A move declared tabu would be accepted when it produces a lower penalized objective value than the previous search's best solution.

5.4.5 Acceleration Strategies

Two acceleration strategies are developed to enhance computational efficiency: the subgraph and the self-adaptive insertion, which differs from the existing research on navigation algorithms.

5.4.5.1 Subgraph Construction

To obtain the shortest path for a particular OD pair, only part of the map is useful since the regions far from the origin and destination are not relevant. Therefore, the algorithm constructs a subgraph by implementing the algorithm to reduce the problem scale and
improve computational efficiency. Figure 5.7 illustrates the procedure of creating a subgraph.

**Figure 5.7 Subgraph Demonstration**

First, the origin and destination are connected by a straight line with a length $L$, as the associated rectangle's diagonal. It enlarges the rectangle by extending its diagonal to both sides by an equal length with a multiplier $\gamma$. Hence, a rectangle subgraph where the length of the diagonal is $(1+2\gamma)L$ is built. The brown rectangle refers to the full graph, and the green area represents the subgraph. Note that solving the shortest path on the subgraph might slightly sacrifice the solution quality, but this can indeed enhance computational efficiency dramatically according to the numerical results.

5.4.5.2 *Self-adaptive Insertion*

Since it is time-consuming if all nodes connected to a specified node $i$ are considered as alternatives to be inserted into the current path after node $i$, it is necessary to propose a self-adaptive insertion approach to improve the efficiency without hurting the solution quality. $L_i$ denotes a set of nodes connected to node $i$, and $L'_i \subset L_i$ refers to a subset that contains the alternative nodes for insertion. The nodes in $L'_i$ are sorted by distance from node $i$ in ascending order. The self-adaptive approach adjusts $|L'_i|$ dynamically based on search performance. a lower limit $D$ and an upper limit $Q$ of $|L'_i|$, and two thresholds $B_1$ and $B_2$ are defined to characterize the self-adaptive adjustment. $|L'_i|$ is initialized by $D$. When
better solutions cannot be found within $B_1$ iterations, $|L'_i|$ is doubled. If better solutions are found continuously for $B_2$ iterations, $|L'_i|$ is half decreased. When an adjustment results in $|L'_i| > Q$ or $|L'_i| < D$, it will not be performed.

5.5 Numerical Study

In this section, numerical studies are conducted to analyze the performance of the proposed algorithm. First, the study investigates the impact of different initial-solution algorithms on computational performance and illustrates each algorithm's adaptability. Secondly, the study evaluates and compares the algorithm's computational performance with CPLEX, LSA, and a state-of-the-art algorithm to validate the necessity to make predictions. The experiments consider two dynamic resources: fuel consumption and driving risk (e.g., risk score). Then, the algorithm is tested on a real network extracted from Manhattan, and the result is compared to Google's solutions. Finally, the result verifies the rationality of the subgraph. The experiments are implemented on a computer with a 2.60 GHz CPU, Microsoft Visual Studio, and CPLEX Solver.

5.5.1 Experimental Data

The random networks in the experiments are formed as follows. All nodes lie in a square region, where both horizontal coordinate $x_i$ and vertical coordinate $y_i$ of each node $i$ are uniformly distributed in $[0, 100]$. A parameter $AD$ is used to determine whether nodes $i$ and $j$ are directly connected. If $|x_i - x_j| < AD$ and $|y_i - y_j| < AD$, then an edge $(i, j)$ is constructed. For each edge $(i, j)$, $D_{ij}$ is defined as the Euclidean distance. $T_{ij0}$ is a random value uniformly distributed in $[2, 4]$. Each $T_{ijt}$ $(t > 0)$ equals $T_{ij0} + Z$ with a probability of 80%. Otherwise, it is equal to $T_{ij0} - Z$, where $Z$ is uniformly distributed in $[0, 2]$. $R_{i/j0}$ is also
a random value uniformly distributed in $[2, 4]$, which is the only resource in LSA. For simplicity, it is assumed that each $R_{tij}$ ($t > 0$) is derived by $R_{1ij0} + Z$ and $R_{1ij0} - Z$ with the same probability. The default parameters in the numerical experiments are as follows: $\alpha=0.5$, $\beta=0.01$, $\theta=3$, $b_1=b_2=5000$, $D=4$, $Q=60$, $B_1=B_2=5$, $UL=1000$.

### 5.5.2 Comparison among Initial-Solution algorithms

This comparison explores the impact of different initial-solution algorithms on computational performance from efficiency and solution quality perspectives. The tabu search algorithms with LSA, Dijkstra, and DSA are denoted as TS1, TS2, and TS3, respectively. Three networks in different scales are randomly generated in this experiment. Necessary information is provided in Table 5.2.

#### Table 5.2 Basic Information of Three Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Arcs</th>
<th>$AD$</th>
<th>$T$</th>
<th>$W_1$</th>
<th>$W_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Network</td>
<td>700</td>
<td>25538</td>
<td>13</td>
<td>150</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Medium Network</td>
<td>2000</td>
<td>182358</td>
<td>12</td>
<td>150</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Large Network</td>
<td>5000</td>
<td>801704</td>
<td>10</td>
<td>150</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

The size of the network depends on the number of nodes. Five OD pairs are chosen in each network as instances, and the computation time (CPU) and the travel cost are compared for each of the three algorithms. A gap between TS1 and other algorithms is defined to indicate the improvement of CPUs compared to TS1. For example, the gap between TS2 and TS1 is calculated by $(CPU(TS1)-CPU(TS2)) / CPU(TS1)$. Similarly, the gap of travel costs between TS3 and other algorithms is calculated to show much improvement of travel costs can be made compared to TS3. The results are reported in Table 5.3-4.5.
### TABLE 5.3 Comparison of Initial-Solution Algorithms: Small Network

<table>
<thead>
<tr>
<th>OD</th>
<th>TS1 CPU (s)</th>
<th>TS1 Travel Cost</th>
<th>TS2 CPU (s)</th>
<th>TS2 Travel Cost</th>
<th>TS3 CPU (s)</th>
<th>TS3 Travel Cost</th>
<th>Gap of CPU</th>
<th>The gap in Travel Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.90</td>
<td>0.65</td>
<td>37.39</td>
<td>0.65</td>
<td>37.39</td>
<td>0.643</td>
<td>38.18</td>
<td>0.2%</td>
<td>1.7%</td>
</tr>
<tr>
<td>35,40</td>
<td>0.63</td>
<td>30.00</td>
<td>0.63</td>
<td>30.00</td>
<td>0.471</td>
<td>31.65</td>
<td>0.8%</td>
<td>25.4%</td>
</tr>
<tr>
<td>100, 500</td>
<td>0.85</td>
<td>44.43</td>
<td>0.83</td>
<td>44.43</td>
<td>0.735</td>
<td>45.33</td>
<td>2.2%</td>
<td>13.7%</td>
</tr>
<tr>
<td>220, 250</td>
<td>1.30</td>
<td>50.94</td>
<td>1.26</td>
<td>54.05</td>
<td>0.576</td>
<td>57.60</td>
<td>3.0%</td>
<td>55.7%</td>
</tr>
<tr>
<td>370, 560</td>
<td>0.83</td>
<td>41.27</td>
<td>0.82</td>
<td>41.27</td>
<td>0.650</td>
<td>43.06</td>
<td>1.6%</td>
<td>21.5%</td>
</tr>
<tr>
<td>Average</td>
<td>0.85</td>
<td>40.80</td>
<td>0.84</td>
<td>41.43</td>
<td>0.615</td>
<td>43.16</td>
<td>1.5%</td>
<td>23.6%</td>
</tr>
</tbody>
</table>

### TABLE 5.4 Comparison of Initial-Solution Algorithms: Medium Network

<table>
<thead>
<tr>
<th>OD</th>
<th>TS1 CPU (s)</th>
<th>TS1 Travel Cost</th>
<th>TS2 CPU (s)</th>
<th>TS2 Travel Cost</th>
<th>TS3 CPU (s)</th>
<th>TS3 Travel Cost</th>
<th>Gap of CPU</th>
<th>The gap in Travel Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>60,80</td>
<td>1.43</td>
<td>24.00</td>
<td>1.33</td>
<td>24.00</td>
<td>1.24</td>
<td>25.46</td>
<td>6.9%</td>
<td>13.0%</td>
</tr>
<tr>
<td>90,120</td>
<td>1.46</td>
<td>30.81</td>
<td>1.44</td>
<td>30.81</td>
<td>1.34</td>
<td>33.29</td>
<td>1.2%</td>
<td>8.4%</td>
</tr>
<tr>
<td>150,160</td>
<td>1.95</td>
<td>42.57</td>
<td>1.82</td>
<td>42.57</td>
<td>1.80</td>
<td>44.56</td>
<td>6.6%</td>
<td>7.8%</td>
</tr>
<tr>
<td>150, 200</td>
<td>1.03</td>
<td>23.03</td>
<td>0.98</td>
<td>23.03</td>
<td>0.97</td>
<td>24.73</td>
<td>4.3%</td>
<td>5.6%</td>
</tr>
<tr>
<td>300, 460</td>
<td>1.38</td>
<td>27.02</td>
<td>1.25</td>
<td>27.02</td>
<td>1.20</td>
<td>28.05</td>
<td>9.2%</td>
<td>12.9%</td>
</tr>
<tr>
<td>Average</td>
<td>1.45</td>
<td>29.49</td>
<td>1.37</td>
<td>29.49</td>
<td>1.31</td>
<td>31.22</td>
<td>5.6%</td>
<td>9.5%</td>
</tr>
</tbody>
</table>

### TABLE 5.5 Comparison of Initial-Solution Algorithms: Large Network

<table>
<thead>
<tr>
<th>OD</th>
<th>TS1 CPU (s)</th>
<th>TS1 Travel Cost</th>
<th>TS2 CPU (s)</th>
<th>TS2 Travel Cost</th>
<th>TS3 CPU (s)</th>
<th>TS3 Travel Cost</th>
<th>Gap of CPU</th>
<th>The gap in Travel Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>35,45</td>
<td>9.77</td>
<td>33.84</td>
<td>7.57</td>
<td>33.84</td>
<td>5.15</td>
<td>37.04</td>
<td>22.3%</td>
<td>47.3%</td>
</tr>
<tr>
<td>240,620</td>
<td>5.03</td>
<td>35.15</td>
<td>4.45</td>
<td>35.15</td>
<td>3.61</td>
<td>38.16</td>
<td>11.5%</td>
<td>28.1%</td>
</tr>
<tr>
<td>320,420</td>
<td>2.95</td>
<td>21.31</td>
<td>2.70</td>
<td>21.31</td>
<td>2.68</td>
<td>22.10</td>
<td>8.6%</td>
<td>9.0%</td>
</tr>
<tr>
<td>620,850</td>
<td>2.44</td>
<td>14.97</td>
<td>2.31</td>
<td>14.97</td>
<td>1.98</td>
<td>15.04</td>
<td>5.5%</td>
<td>18.8%</td>
</tr>
<tr>
<td>1000, 2722</td>
<td>7.21</td>
<td>39.83</td>
<td>5.79</td>
<td>39.83</td>
<td>4.60</td>
<td>42.27</td>
<td>19.8%</td>
<td>36.3%</td>
</tr>
<tr>
<td>Average</td>
<td>5.48</td>
<td>29.02</td>
<td>4.57</td>
<td>29.02</td>
<td>3.60</td>
<td>30.92</td>
<td>13.5%</td>
<td>27.9%</td>
</tr>
</tbody>
</table>
In terms of the gap of CPUs, TS3 is the best, TS1 is the worst, and TS2 is in the middle. Meanwhile, TS1 is slightly better than TS2, and TS3 is the worst regarding the gap in travel cost. It is interesting to check the efficiency superiority of TS1, TS2, and TS3 for different network sizes. The average CPU and travel cost of each algorithm in different networks are shown in Figure 5.8-4.9.

![Figure 5.8 Average CPU of Each Algorithm in Each Scale](image1)

**Figure 5.8 Average CPU of Each Algorithm in Each Scale**

![Figure 5.9 Average Travel Cost of Each Algorithm in Each Scale](image2)

**Figure 5.9 Average Travel Cost of Each Algorithm in Each Scale**

According to Figure 5.8, as the problem scale enlarges, the CPU of TS1 increases faster than TS2, and the CPU of TS3 grows at the lowest rate. This finding suggests that the efficiency superiority of TS3 turns out to be more significant in large-scale problems. It
can be seen in Figure 5.9 that the superiority of the solution quality of TS1 and TS2 to TS3 remains relatively stable at all scales.

The results indicate that initial-solution algorithms should be selected according to the problem scale. More specifically, if the problem scale is small, the algorithm tends to choose TS1 due to its capability to deliver the best solution quality while keeping an acceptable deviation on efficiency. TS3 should be ideal for a large-scale problem due to its significant efficiency and acceptable variation in solution quality.

5.5.3 Comparison with other solution approaches

The algorithm is first compared with CPLEX and LSA and then compared with a state-of-the-art algorithm to further investigate the proposed algorithms' performance. In this section, TS2 is adopted to generate the travel path.

5.5.3.1 Comparison with CPLEX

The comparison of the algorithm with commercial software CPLEX uses a random network consisting of 80 nodes and 504 edges. The experiment has parameters $AD=15$, $T=30$, $W_1=W_2=30$, $UL=100$, and randomly select 20 OD pairs. For each OD pair, the total travel cost and the computation time obtained by the algorithm and by CPLEX are listed in Table 5.6.
The average cost of TS and CPLEX are the same (36.26), but the CPU time of TS (0.02) is much faster than CPLEX (0.87). Both the proposed algorithm and CPLEX obtain the optimal solutions for these instances. The average computation time of CPLEX is almost 48 times longer than the proposed algorithm, indicating a remarkable time-saving effect by the algorithm while keeping the same solution quality.

5.5.3.2 Comparison with LSA

Some conventional dynamic programming algorithms (e.g., LSA) applied in urban navigation can provide the solution efficiently with current traffic information. However, they fail to consider the impact of time-varying traffic information on routing decisions, which may not lead to a satisfactory route decision.

The experiment assesses the solutions generated by TS2 and LSA in the medium size network in Section 4.5.2 to verify this. For simplicity, only the fuel consumption is considered as a resource, and the corresponding upper limit is 20. The system randomly chooses 20 OD pairs. For each OD instance, TS2 uses current information and historical information, while LSA only uses current information to generate a travel path. The
expected travel cost of the two solutions when reaching the destination is computed. For each instance, the total cost gap of TS2 compared with LSA, CPUs, and travel paths are listed in Table 5.7.

Table 5.7 Comparison with LSA

<table>
<thead>
<tr>
<th>OD</th>
<th>LSA Cost</th>
<th>LSA CPU(s)</th>
<th>Travel Path</th>
<th>TS Cost</th>
<th>TS CPU(s)</th>
<th>Travel Path</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>160,220</td>
<td>38.5</td>
<td>0.30</td>
<td>160,1685,1698,784,681,549,220</td>
<td>37.9</td>
<td>1.70</td>
<td>160,318,1698,784,681,1271,1220</td>
<td>1.5%</td>
</tr>
<tr>
<td>1880,120</td>
<td>32.8</td>
<td>0.19</td>
<td>1880,533,1382,865,1990,120</td>
<td>31.1</td>
<td>1.68</td>
<td>1880,533,514,1938,1427,1247,120</td>
<td>4.9%</td>
</tr>
<tr>
<td>846,160</td>
<td>29.3</td>
<td>0.06</td>
<td>846,1794,189,1993,160</td>
<td>27.3</td>
<td>1.20</td>
<td>846,1794,1364,189,1993,160</td>
<td>6.8%</td>
</tr>
<tr>
<td>60,80</td>
<td>25.5</td>
<td>0.13</td>
<td>60,5,725,369,80</td>
<td>24.6</td>
<td>1.22</td>
<td>60,5,869,725,369,80</td>
<td>3.6%</td>
</tr>
<tr>
<td>150,200</td>
<td>23.5</td>
<td>0.03</td>
<td>150,335,711,200</td>
<td>22.4</td>
<td>0.97</td>
<td>150,335,1959,200</td>
<td>4.8%</td>
</tr>
<tr>
<td>290,310</td>
<td>24.6</td>
<td>0.04</td>
<td>290,1381,1150,1420,310</td>
<td>23.9</td>
<td>1.20</td>
<td>290,1381,1150,1984,1420,310</td>
<td>2.9%</td>
</tr>
<tr>
<td>1021,440</td>
<td>36.1</td>
<td>0.23</td>
<td>1021,703,727,1288,554,664,440</td>
<td>34.7</td>
<td>1.72</td>
<td>1021,703,782,151,1288,554,664,440</td>
<td>3.9%</td>
</tr>
<tr>
<td>86,241</td>
<td>27.2</td>
<td>0.03</td>
<td>86,643,555,861</td>
<td>25.1</td>
<td>1.54</td>
<td>86,778,643,1349,861,1040,241</td>
<td>7.8%</td>
</tr>
<tr>
<td>335,776</td>
<td>33.8</td>
<td>0.13</td>
<td>335,343,1867,1733,1410,776</td>
<td>29.4</td>
<td>1.61</td>
<td>335,1025,1371,672,852,1099,1410,776</td>
<td>12.9%</td>
</tr>
<tr>
<td>336,868</td>
<td>18.9</td>
<td>0.02</td>
<td>336,1685,1738,868</td>
<td>17.9</td>
<td>0.90</td>
<td>336,1685,713,868</td>
<td>5.3%</td>
</tr>
<tr>
<td>1067, 1895</td>
<td>29.7</td>
<td>0.18</td>
<td>1067,1630,350,1769,1895</td>
<td>27.8</td>
<td>1.56</td>
<td>1067,1510,1010,1947,869,1415,1633,1895</td>
<td>6.3%</td>
</tr>
<tr>
<td>741,943</td>
<td>16.3</td>
<td>0.02</td>
<td>741,52,643,943</td>
<td>14.1</td>
<td>0.87</td>
<td>741,122,1111,943</td>
<td>13.4%</td>
</tr>
<tr>
<td>335,1410</td>
<td>28.0</td>
<td>0.06</td>
<td>335,343,1867,1733,1410</td>
<td>25.1</td>
<td>1.39</td>
<td>335,122,343,1257,1733,1410</td>
<td>10.2%</td>
</tr>
<tr>
<td>334,982</td>
<td>20.0</td>
<td>0.04</td>
<td>334,888,214,982</td>
<td>19.0</td>
<td>1.07</td>
<td>334,888,250,982</td>
<td>4.6%</td>
</tr>
<tr>
<td>1359,662</td>
<td>26.2</td>
<td>0.04</td>
<td>1359,367,1974,937,662</td>
<td>24.4</td>
<td>1.21</td>
<td>1359,1628,1974,937,662</td>
<td>6.7%</td>
</tr>
<tr>
<td>480,919</td>
<td>18.1</td>
<td>0.03</td>
<td>480,1789,1466,919</td>
<td>16.6</td>
<td>0.81</td>
<td>480,1482,1841,919</td>
<td>8.6%</td>
</tr>
<tr>
<td>510,620</td>
<td>26.1</td>
<td>0.04</td>
<td>510,1274,594,1230,620</td>
<td>24.3</td>
<td>1.16</td>
<td>510,1695,594,1230,620</td>
<td>6.8%</td>
</tr>
<tr>
<td>390,861</td>
<td>28.5</td>
<td>0.03</td>
<td>390,644,1652,1349,861</td>
<td>25.8</td>
<td>1.33</td>
<td>390,50,864,343,1111,52,861</td>
<td>9.5%</td>
</tr>
<tr>
<td>1,100</td>
<td>33.1</td>
<td>0.16</td>
<td>1,010,633,1603,1303,100</td>
<td>32.2</td>
<td>1.43</td>
<td>0,214,633,1603,649,100</td>
<td>2.9%</td>
</tr>
<tr>
<td>980,1500</td>
<td>30.8</td>
<td>0.26</td>
<td>980,737,1010,1229,854,1500</td>
<td>27.5</td>
<td>1.66</td>
<td>980,737,364,805,317,704,1500</td>
<td>10.7%</td>
</tr>
<tr>
<td>Average</td>
<td>0.10</td>
<td></td>
<td></td>
<td></td>
<td>1.70</td>
<td></td>
<td>6.7%</td>
</tr>
</tbody>
</table>

The result shows that the average gap is 6.7%, which implies that the impact of time-varying traffic information should not be neglected. It is practical to integrate the current and historical traffic information. The CPU of TS2 and LSA are 1.311s and 0.101s, respectively, both acceptable in urban navigation. These findings also verify that the proposed algorithm properly combines two types of information.
5.5.3.3  **Comparison with a state-of-the-art Algorithm**

This comparison compares the proposed approach and Wang et al.’s study [105]. The authors formulate their problem as an integer programming model to minimize the expected travel time. For a fair comparison, this problem is made identical to theirs. The $\alpha=0$ so that only the travel time is incorporated in the objective function. Moreover, the heuristic is implemented on the same network, containing 123 nodes and 342 edges, with the travel time, distance, and fuel consumption available on each edge. The experiment considers the same resources as Wang et al.’s: fuel consumption and distance, with identical constraint limitations. The planning horizon $T$ is 100, $UL=100$. The same OD instances as in Wang et al.’s paper are used, and the total travel time, total distance, total fuel consumption, and average computation time by the algorithm are calculated. Table 5.8 compares the results of the algorithm with theirs.

**Table 5.8 Comparison with Wang L et al.’s Algorithm**

<table>
<thead>
<tr>
<th>OD</th>
<th>Wang L et al.’s Algorithm</th>
<th>TS</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Travel Time</td>
<td>Distance</td>
<td>Fuel Consumption</td>
</tr>
<tr>
<td>1.83</td>
<td>57.70</td>
<td>36</td>
<td>4.04</td>
</tr>
<tr>
<td>8,113</td>
<td>54.10</td>
<td>40</td>
<td>4.42</td>
</tr>
<tr>
<td>50,94</td>
<td>41.80</td>
<td>30</td>
<td>3.47</td>
</tr>
<tr>
<td>15,102</td>
<td>45.60</td>
<td>27</td>
<td>3.34</td>
</tr>
<tr>
<td>72,80</td>
<td>30.80</td>
<td>25</td>
<td>2.83</td>
</tr>
<tr>
<td>61,112</td>
<td>39.40</td>
<td>30</td>
<td>3.64</td>
</tr>
<tr>
<td>56,95</td>
<td>46.40</td>
<td>35</td>
<td>3.54</td>
</tr>
<tr>
<td>7,123</td>
<td>50.50</td>
<td>37</td>
<td>4.20</td>
</tr>
<tr>
<td>39,90</td>
<td>50.90</td>
<td>44</td>
<td>4.74</td>
</tr>
<tr>
<td>19,80</td>
<td>51.50</td>
<td>31</td>
<td>3.82</td>
</tr>
<tr>
<td>Average</td>
<td>46.87</td>
<td>34</td>
<td>3.80</td>
</tr>
</tbody>
</table>
The two algorithms obtain the same travel path in each instance. Besides, the mean CPU is 0.019s, and Wang L et al.’s is the 20s. The proposed algorithm can run faster than Wang L et al. without loss of the solution quality.

5.5.4 Verification of the Subgraph

Two acceleration strategies, the subgraph and the self-adaptive insertion approach, are proposed. The former requires the verification of rationality, while the latter is more straightforward to be understood. The experiment analyzes the impact of the subgraph scale on the trade-off between solution quality and computational efficiency. To this end, a sensitivity analysis on parameter $\gamma$ will be implemented. The real network in Section 5.5.4 is used to obtain the travel cost and CPU of 7 OD pairs on the full graph with $\alpha=0.5$. Then, $\gamma$ is adjusted from 0 to 2, and the change of travel cost and CPU is measured by the percentile gap based on the results from both subgraph and full graph. The average number of nodes and edges in the subgraph and the average cost gap and CPU gap for each $\gamma$ are reported in Figure 5.10.

![Figure 5.10 The Impact on Algorithm Performance by Subgraph Scale](image)

The graph shows that the travel cost decreases dramatically as the subgraph scale grows, and a considerable time-saving effect shaped by $\gamma$ is observed. In particular, the best travel path in a subgraph is identified with almost 20% improvement in efficiency than the full
graph when $\gamma=0.3$, which illustrates that a well-performed navigation solution can be delivered efficiently with a partially loaded map.

5.5.5 Real Network Demonstration

Previous sections show the superiority of the algorithm in randomly generated networks. This section demonstrates the SRM methodology for risk score calculation and LSTM autoencoder for travel time prediction. It examines the algorithm's performance in a real road network extracted from Manhattan, NY, USA, by OSMNX [140], as shown in Figure 5.11.

![Real Network Extracted from Manhattan, NY, USA](image)

The Manhattan map contains 4,579 nodes and 9,877 edges, which can be regarded as a large-scale problem. The experiment extracts the upper west side of Manhattan, which has 369 nodes and 809 edges, to visualize the result more straightforwardly. On each edge, the road characteristics such as speed limit, number of lanes, one-way, and length are obtained from OSMNX [140]. Historical crashes are obtained from NYC Crash Mapper [141]. At different times, travel time is acquired from Google API [142], and the dynamic resources are randomly derived.
5.5.5.1 Risk score computation

The upper west side of Manhattan is chosen to show how the risk score is obtained. Figure 12 shows the NN model structure for crash probability calculation with various types of input features.

![Neural Network Structure](image)

**Figure 5.12 The proposed Neural Network for crash probability calculation**

There are three types of inputs associated with historical crashes for the NN model: Temporal features, road characteristics, and traffic conditions. Since the crash database only has positive data (crash) in the database, it is necessary to use the idea of Positive and Unlabeled (PU) Learning [143] to generate the unlabeled samples by randomly changing the features of positive samples. A positive dataset is $\chi_p = \{x^i_p\}_{i=1}^{n_p}$ and an unlabeled dataset is $\chi_u = \{x^i_u\}_{i=1}^{n_u}$. It is assumed that all unlabeled samples are negative samples $\chi_n = \{x^i_n\}_{i=1}^{n_n} \sim \chi_u = \{x^i_u\}_{i=1}^{n_u}$ and are 1:1 blended into the positive samples to simplify the problem. In this way, the classifier can be trained in an ordinary supervised learning
fashion. In training, training and testing datasets are partitioned by 7:3, and Table IX presents the training and testing confusion matrix.

**Table 5.9 Training vs. Testing Confusion Matrix**

<table>
<thead>
<tr>
<th></th>
<th>Actual non-conflict</th>
<th>Actual conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N=40731</td>
<td>18800</td>
<td>1566</td>
</tr>
<tr>
<td>Predicted non-conflict</td>
<td>92.3%</td>
<td></td>
</tr>
<tr>
<td>Predicted conflict</td>
<td>73.1%</td>
<td></td>
</tr>
<tr>
<td>77.4%</td>
<td>90.5%</td>
<td></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N=17457</td>
<td>8001</td>
<td>727</td>
</tr>
<tr>
<td>Predicted non-conflict</td>
<td>91.7%</td>
<td></td>
</tr>
<tr>
<td>Predicted conflict</td>
<td>71.8%</td>
<td></td>
</tr>
<tr>
<td>76.5%</td>
<td>89.6%</td>
<td></td>
</tr>
</tbody>
</table>

The rows of the matrices are the predicted class, and the columns are the actual class. The diagonal cells represent correctly classified observations. The off-diagonal cells represent observations that are incorrectly classified. The last column of the plot shows the percentages of correct and incorrect predictions (positive predictive value and false discovery rate, respectively). The bottom row shows the percentages of classified and unclassified examples in each class. It is noted that the overall prediction accuracy of training data is 76.9%, and testing data is 76.6%, which yields good prediction results. The above analysis evaluates the risk of a roadway from a dynamic perspective by predicting the risk probability in real-time for each edge based on time-varying attributes. The study continues the experiment by predicting the number of crashes for each edge to assess the roadway from a static perspective. Follow the SPF procedure in [112], Figure 5.13 compares the actual and estimated crash counts for 2020 using historical crash data from 2012-2019 and predicts 2021.
The Freeman-Tukey coefficient of determination ($R^2 = 0.91$) suggests that the predictive model can adequately explain crash counts' variation. The above analysis evaluates the risk of a roadway from both dynamic and static perspectives. The fuzzy logic model combines the two measures into a more comprehensive measure, risk score. $\tilde{CC}$ and $\tilde{RP}$ denote the fuzzy set of estimated crash counts and risk probabilities, respectively. The fuzzy output set, risk score, is denoted as $\tilde{RS}$. Gaussian membership functions are used for the fuzzification of input sets and defuzzification of the output set. Each fuzzy set has five membership functions that describe the probabilities of linguistic variables “very low” (VL), “low” (L), “medium” (M), “high” (H), and “very high” (VH), respectively. Table 5.10 shows the matrix representation of all fuzzy rules for $\tilde{RS}$.
Table 5.10 The Fuzzy Logic Reasoning Rule Matrix

<table>
<thead>
<tr>
<th>𝐶𝐶</th>
<th>VL</th>
<th>L</th>
<th>M</th>
<th>H</th>
<th>VH</th>
</tr>
</thead>
<tbody>
<tr>
<td>VL</td>
<td>VL</td>
<td>VL</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>L</td>
<td>VL</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>M</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>VH</td>
</tr>
<tr>
<td>VH</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>VH</td>
<td>VH</td>
</tr>
</tbody>
</table>

Row captions in the matrix contain the values of the risk probability can take, column captions have the values of the crash count can take, and each cell is the resulting risk score when the input variables take the values in that row and column. For instance, the cell (4, 2) can be interpreted as: if the crash count is high, and the risk probability is low, then the risk score is medium. From the results of crash and conflict estimates, the range of membership functions for crash counts is set to [0,60] since the maximum number of predicted crashes in 2018 is 60 (Figure 5.13), and the range of membership functions for conflict probability is set to [0,1]. The range of membership functions for risk score is set to [0,100], with 100 being extremely risky. It is also a straightforward way to display the risk level for the driver. By defuzzifying the aggregated shape of pairwise comparisons for all the rules, risk scores are obtained. Figure 5.14 illustrates the reasoning process of 3 sample fuzzy rules for an input pair (conflict probability = a%, crash count = b).

![Figure 5.14 Illustration of the fuzzy reasoning process](image-url)
Finally, the SRM methodology is implemented to color the sample road map based on the risk score calculated by historical crashes and real-time risk probabilities. Figure 5.15 displays the Crash Data on a road network and heatmaps the roadway based on the calculated risk scores.

Figure 5.15 Crash Data and risk heatmap on a sample roadway

Road features are extracted from OSMNX; historical crash data are obtained from NYC Crash Mapper. With essential information and data, the SRM methodology can calculate roadways' risk scores at different locations and times. For example, the average risk score on a clear Sunday morning is relatively low, while on a fog peak-hour Monday morning, it is significantly higher.

5.5.5.2 Travel time projection

Following the procedure in 5.3.2, the first step is to decide the best hyperparameter values to use before coming to the final model. WMAPE is used as the performance metric. And experiments are designed to find how many epochs and how many months to use for the one-step-ahead forecast can lead to the minimal testing WMAPE. Five replications are run
for each (epoch, step) combination to get the mean and std WMAPE and show the training and testing performance in Figure 5.16.

Figure 5.16 Training and testing WMAPE associated with training epochs and steps

The trend of training errors is going downwards while testing errors fluctuate. The lowest testing WMAPE sits at 3,000 training epochs and six steps. Therefore, those two values are adopted to train the final predictive model on the full historical travel time. Figure 5.17 shows the overall structure of the LSTM model for the final model.

Figure 5.17 The overall structure of the LSTM Autoencoder model
The input is a 6*9,877 matrix, where 6 represents the number of input time steps, and 9,877 is the number of edges. The final model can predict one-step-ahead travel time for all edges at the same time. Some of the prediction results are shown in Figure 5.18.

![Figure 5.18 Training and testing results of sample edges](image)

Blue circles are the travel time for these edges, and yellow circles represent the fitted results. Red circles represent the forecasted result for the future, which is unknown at the moment of planning. As more data points available over time, multi-step-ahead of travel times can be predicted by recursively using this model.

5.5.5.3 **Data-driven Route planning**

The study tests the routing decision based on the shortest path, fastest path, and most rapid path with safety constraints. Figure 5.19 shows the origin, destination, and nodes along different paths.
Figure 5.19 Route planning according to different objectives and constraints

In the example, the planning horizon $T$ is 60 minutes. The departure time is 14:30 on a Thursday. The origin is node 93, and the destination is node 64. Future travel time and risk on each roadway and time step are pre-calculated at departure. The algorithm provides three paths according to different objectives and constraints. Path1 is the shortest path obtained by setting $\alpha = 1$, path2 is the fastest path obtained by setting $\alpha = 0$, and path3 is obtained by capping the risk score to 26. Table 5.11 lists the detailed information of each path.
Table 5.11 Routing Decision Based on Different Objectives and Constraints

<table>
<thead>
<tr>
<th>Path</th>
<th>Google</th>
<th>TS3</th>
<th>Avg Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (min)</td>
<td>Dist (km)</td>
<td>Travel Path</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.60</td>
<td>93, 1004, 2111, 2113, 2112, 1838, 316, 64</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.75</td>
<td>93, 3624, 3244, 69, 3633, 64</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0.85</td>
<td>NA</td>
</tr>
</tbody>
</table>

On this sample map, the shortest path and the fastest path are the same as the Google map. An alternative route that limits the average risk to no more than 26 is also found. Ten OD pairs are randomly chosen as instances in this test to examine the algorithm's performance further. For each instance, the algorithm attempts to find the travel path with the least cost: the weighted sum of travel time and travel distance by TS3 ($\alpha=0.5$). The resource constraints are relaxed since the two resources considered here are not available in the Google Map. Finally, each path's total travel time and distance are evaluated using the data and displayed in Table 5.12.
Table 5.12 Real Network Experiments

<table>
<thead>
<tr>
<th>OD</th>
<th>Google Time (min)</th>
<th>Dist (km)</th>
<th>Travel Path</th>
<th>TS3 Time (min)</th>
<th>Dist (km)</th>
<th>Travel Path</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4541, 2278</td>
<td>5</td>
<td>2.9</td>
<td>4541, 3393, 3392, 95, 97, 163, 1137, 4142, 4143, 1133, 1134, 2786, 2279, 2278</td>
<td>5</td>
<td>2.9</td>
<td>4541, 3393, 3392, 95, 97, 163, 1137, 4142, 4143, 1133, 1134, 2786, 2279, 2278</td>
<td>0.34</td>
</tr>
<tr>
<td>1406, 2283</td>
<td>10</td>
<td>4</td>
<td>1406, 1956, 2985, 2294, 2283</td>
<td>10</td>
<td>3.4</td>
<td>1407, 2985, 2294, 2283</td>
<td>0.25</td>
</tr>
<tr>
<td>2841, 4088</td>
<td>6</td>
<td>1.6</td>
<td>2841, 2288, 4095, 4088</td>
<td>6</td>
<td>1.6</td>
<td>2841, 2288, 4095, 4088</td>
<td>0.21</td>
</tr>
<tr>
<td>1787, 2294</td>
<td>7</td>
<td>2.7</td>
<td>1787, 102,138, 4464, 2296, 2294</td>
<td>7</td>
<td>2.7</td>
<td>1787, 102,138, 4464, 2296, 2294</td>
<td>0.45</td>
</tr>
<tr>
<td>3118, 800</td>
<td>11</td>
<td>4</td>
<td>3118, 2574, 2034, 2706, 802, 800</td>
<td>10</td>
<td>4</td>
<td>3118, 2574, 2034, 274, 2024, 4150, 802, 800</td>
<td>0.36</td>
</tr>
<tr>
<td>3874, 3324</td>
<td>10</td>
<td>4</td>
<td>3874, 1617, 401, 377, 620, 888</td>
<td>10</td>
<td>3.9</td>
<td>3874, 1617, 2224, 389, 377, 620, 888</td>
<td>0.46</td>
</tr>
<tr>
<td>2537, 756</td>
<td>7</td>
<td>2.7</td>
<td>2537, 2538, 3256, 756</td>
<td>7</td>
<td>2.7</td>
<td>2537, 2538, 3256, 756</td>
<td>0.21</td>
</tr>
<tr>
<td>1220, 1402</td>
<td>6</td>
<td>2.1</td>
<td>1220, 1415, 1412, 2446, 1402</td>
<td>6</td>
<td>2.1</td>
<td>1220, 1415, 1412, 2446, 1402</td>
<td>0.44</td>
</tr>
<tr>
<td>3499, 3834</td>
<td>5</td>
<td>1.8</td>
<td>3499, 464, 459, 3557, 453, 3836, 3834</td>
<td>5</td>
<td>1.8</td>
<td>3499, 464, 459, 3557, 453, 3836, 3834</td>
<td>0.24</td>
</tr>
</tbody>
</table>

Four instances where the proposed algorithm slightly outperforms the Google map in terms of travel distance or travel time are highlighted. For other OD pairs, a tie between the proposed algorithm and Google Map is found. The result shows the satisfactory performance of the proposed algorithm. Therefore, it can be applied in urban navigation.

As stated in 5.1, navigation software can update the shortest path frequently. Since the proposed algorithm can outperform the software for a certain run, it is natural to perform the algorithm with the same update rate as Google Map. Thus the quality of the final solution will outperform the software. Besides, the average CPU of the algorithm is 0.33, which presents an acceptable efficiency. Moreover, if the experiment is performed on a subgraph instead of the full graph, the algorithm can run much faster, as discussed in 5.5.4.
5.6 Conclusion

This chapter investigates a dynamic shortest path problem (DSPP) considering time-varying travel time and resources obtained by data-driven methodologies. The study formulates the problem as a mixed-integer programming problem to minimize travel costs under risk constraints. The risk score and travel time in the planning horizon are estimated by the SRM methodology and LSTM with Autoencoder, respectively. A modified tabu search algorithm with three candidate approaches for the initial solution concerning different problem scales is proposed to boost the DSPP algorithm's performance. The searching algorithm starts with constructing the neighborhood by two local moves and then move from the current solution to the best neighboring solution at each iteration. Besides, subgraph and self-adaptive insertion methods are proposed to enhance computational efficiency. The proposed algorithm's superiority is verified by comparing with CPLEX, LSA, and a state-of-the-art algorithm. It also competes against the Google Map in a real network in Manhattan, NY, US. The numerical results suggest that the proposed algorithm can achieve good efficiency without losing the solution quality.
Reference


[23] Federal Motor Carrier Safety Administration, Report to Congress on the Large Truck Crash Causation Study: Author Washington, DC.


[38] T. A. Dingus et al., “The 100-car naturalistic driving study, Phase II-results of the 100-car field experiment,”
[59] J. Xu, K. M. Kockelman, and Y. Wang, “Modeling crash and fatality counts along mainlanes and 1 frontage roads across texas: 2 the roles of design, the built environment, and weather 3,” in 93rd Annual Meeting of the Transportation Research, p. 24.
[99] N. Y.S. DOT, New York State Department of Transportation Traffic Data Viewer.
[101] Y. Zhang, Z.-J. Max Shen, and S. Song, "Lagrangian relaxation for the reliable shortest path problem with correlated link travel times," Transportation Research Part B:


A. Faro and D. Giordano, "Algorithms to find shortest and alternative paths in free flow and congested traffic regimes," *Transportation Research Part C: Emerging


E. Galbrun, K. Pelechrinis, and E. Terzi, "Urban navigation beyond shortest route:


