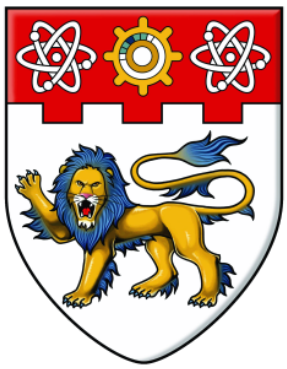


NANYANG TECHNOLOGICAL UNIVERSITY

MAXIMIZING THE PROBABILITY OF ARRIVING ON TIME: A STOCHASTIC SHORTEST PATH PROBLEM

CAO ZHIGUANG

INTERDISCIPLINARY GRADUATE SCHOOL
ENERGY RESEARCH INSTITUTE @ NTU (ERI@N)

2017

MAXIMIZING THE PROBABILITY OF ARRIVING ON TIME: A STOCHASTIC SHORTEST PATH PROBLEM

CAO ZHIGUANG

Interdisciplinary Graduate School
Energy Research Institute @ NTU (ERI@N)

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2017

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

. . 06/MAR/2017 . .
Date

……..Cao Zhiguang . . . . .
Student Name

Abstract

Traffic and transportation are crucial to the sustainable development of most metropolitan cities, where the stochastic shortest path (SSP) problem for vehicle routing is a challenging topic. For this problem, optimization-based methods and multiagent-based methods are usually adopted to compute the optimal path for a single vehicle and multiple cooperative vehicles, respectively. Most of these methods take into account various uncertainties in traffic and yields route recommendations dynamically, leading to a better solution to the development of sustainable transportation systems. At the same time, various criteria for the optimal paths have been proposed for the SSP problem. Among them, the probability tail (PT) model aims to find a path that maximizes the probability of arriving on time. It is promising in that, it integrates travel time, risk (associated with the variance in the probability) and deadline. Therefore, this thesis focuses on solving this arriving on time problem, for a single vehicle and multiple cooperative vehicles. Regarding the former, an optimal path for a single vehicle is independently computed using the travel time data of the road links. Regarding the latter, the optimal paths for multiple vehicles are cooperatively computed by exploring their intentions.

To circumvent the unrealistic assumptions in the current solutions to the arriving on time problem for a single vehicle, such as Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines, a data-driven approach is developed. More specifically, the PT model based SSP problem is first reformulated as a cardinality minimization problem by directly utilizing travel time data on each road link. Then, this minimization problem is approximately solved via relaxing the cardinality by ℓ_1-norm and its variants, and further formulating it as a mixed integer linear programming (MILP) problem. Consequently, this arriving on time problem becomes solvable via an existing MILP solver, e.g., branch and bound (B&B) method.

To improve the computation efficiency of the data-driven approach, the property of total unimodularity (TU) that exists in the equality constraint of the MILP-based arriving on time problem is further explored. This nice property can guarantee integer solutions by solving the

corresponding linear programming (LP) problem if there is no inequality constraint. In view of this, the partial Lagrange multiplier method is employed to relax the undesired inequality constraints in the MILP problem by shifting them to the objective function. After that, this relaxed problem is solved using the subgradient method in an iterative manner, and only LP problems need to be solved in each iteration, which saves computation cost significantly.

To increase the accuracy of finding the real optimal path (i.e., considering that ℓ_1-norm relaxation is an approximation to the cardinality minimization), a practical Q-learning method is developed. In particular, this Q-learning method aims to maximize the expected reward regarding whether the vehicle can arrive on time or not on a specified path. At the same time, the expected reward has the meaning of probability of arriving on time from a current location to destination. To address the issues of continuous deadlines and large-scale road networks, a practical value function approximation method is further proposed. Specifically, it adopts dynamic neural network to directly learn the optimal Q-value in each iteration, which which represents the probability of arriving at the destination on time.

To address the arriving on time problem for multiple cooperative vehicles, a decentralized multiagent-based approach is proposed. It aims to increase the chances of arriving on time for all relevant vehicles. In this approach, two types of agents are involved, i.e., vehicle agent and infrastructure agent. Each vehicle agent follows the route guidance by the local infrastructure agent, and the infrastructure agent collects intentions (i.e., destinations and preferred deadlines) from local vehicle agents. The infrastructure agent formulates the guidance as a route assignment problem, the objective of which is to minimize the cardinality of any possible delays for all local vehicle agents. At the same time, the route assignment problem for each infrastructure agent is solved by existing solvers. Additionally, the proposed route guidance approach is also improved in the aspects of travel time prediction for the assigned road link, computational efficiency of solving route assignment, and acquirement of real-time traffic condition.

Extensive experiments on both artificial and real road networks are conducted to evaluate all the above proposed methods. The experimental results confirm the desirable advantages of the proposed methods over others. Therefore, the research in this thesis provides important contributions to the development of intelligent transportation systems.

Acknowledgments

Upon the completion of this Ph.D thesis, I would like to express my great thanks to those who have been supporting me in the last three years. It is their professional guidance and kind patience that brings me the chance for this Doctoral degree.

Among those people, I am extremely grateful to my main supervisor Assoc Prof. Zhang Jie for his continuous, patient and invaluable guidance. It has been a fruitful and enjoyable experience to work with him. I especially appreciate his "push", which I always believe can bring me more achievements beyond my expectation. In the meantime, I want to thank Assoc Prof. Guo Hongliang greatly, with whom I discuss the research problem almost every day. And I really benefit a lot from those discussion. Additionally, I would like take this chance to express sincere appreciation to my co-supervisors Assoc Prof. Dusit Niyato and Prof. Huang Guang-bin, and my mentor Prof. Ong Yew Soon. All the discussion with them makes me feel enlightened, which is deeply acknowledged.

I appreciate the research and finance support from the BMW-NTU Future Mobility Research Lab, who is especially generous with my overseas conferences in Germany and USA. I also want to thank my colleagues, Sun Haoqi, Han Wei, Cheng Zihao, Liu Tianchi, Jim Cherian and Ghosh Banishree for the fruitful and pleasant discussion, and their friendships as well. Without them, the research cannot be such an enriching and enjoyable experience.

Finally, my immense gratitude goes to my family, especially my wife, my parents and other family members, who were always there cheering me up and stood by me through all my up-and-downs during the past three years.

Table of Contents

Abstract i
Acknowledgments iii
Table of Contents v
Table Captions ix
Figure Captions xi
Abbreviations xiii

1 Introduction **1**
1.1 Background and Motivation 2
1.2 Stochastic Shortest Path Problem 3
1.2.1 SSP Problem for a Single Vehicle 4
1.2.2 SSP Problem for Multiple Cooperative Vehicles 6
1.3 The Objectives and Proposed Approaches 7
1.3.1 A Data-Driven Approach 8
1.3.2 Improving Computation Efficiency via Partial Lagrange Multiplier . . . 8
1.3.3 Increasing Accuracy via Q-learning 8
1.3.4 A Multiagent-based Method 9
1.4 Organization 10

2 Literature Review **13**
2.1 SSP Problem for a Single Vehicle 14
2.1.1 Least Expected Travel Time 14
2.1.2 Mean-Risk Model 16
2.1.3 User-defined Utility or Penalty Function 17
2.1.4 Faster Criterion 18
2.1.5 Probability Tail (PT) Model 19
2.2 SSP Problem for Multiple Cooperative Vehicles 21
2.3 Summary 23

3 Arriving on Time for a Single Vehicle **25**

3.1 Problem Formulation and Reformulation 26

3.1.1 Original Problem Formulation 26

3.1.2 Problem Reformulation as Cardinality Minimization 27

3.2 Solution Method for Cardinality Minimization 28

3.2.1 ℓ_1-norm Minimization to Solve Cardinality Minimization 29

3.2.2 Iterated Weighted ℓ_1-norm Algorithm 29

3.2.3 Reweighted ℓ_1-norm Algorithm 30

3.2.4 Further Reformulated as an MILP Problem 31

3.2.5 Analysis of Accuracy and Computation Complexity 33

3.3 Experimentation 34

3.3.1 Test Scenario: A Simple Network 36

3.3.2 Description of Symbols and Settings 37

3.3.3 Case Study 1: Single Independent Distribution 38

3.3.4 Case Study 2: Blended Distributions 40

3.3.5 Case Study 3: Correlated Distributions 41

3.3.6 Computation Time 43

3.4 Parameter Analysis 44

3.4.1 Distribution Parameters 45

3.4.2 Ratio of Training Size over Testing Size 46

3.4.3 O-D Pair 47

3.4.4 Graph Scale and Data Size 49

3.4.5 Deadline Coefficients 50

3.5 Testing on Real World Road Network 51

3.6 Summary 55

4 Improving the Computation Efficiency **57**

4.1 The Two Models and Total Unimodularity 58

4.1.1 The Two Models 58

4.1.2 Total Unimodularity of the Incidence Matrix 59

4.2 Partial Lagrange Multiplier Method 60
4.2.1 Partial Lagrange Multiplier 60
4.2.2 Solution to the Partial Lagrange Multiplier 61
4.2.3 Improvement 63
4.2.4 Generalization 64
4.3 Theoretical Performance Analysis 65
4.3.1 Convergence Analysis 65
4.3.2 Scalability Analysis 68
4.4 Experimental Analysis 69
4.4.1 Small Scale: Three Different Graphs 70
4.4.2 Medium Scale: Graph of Singapore Main Roads 74
4.4.3 Large Scale: Graph of All Roads in Beijing 75
4.4.4 Implementation on Real Navigation System 78
4.5 Summary 80

5 Improving the Accuracy **83**
5.1 Background on Q-learning 84
5.2 Basic Q-Learning for Discrete Deadlines 85
5.2.1 MDP Formulation for PT Model 86
5.2.2 Q-Value Representation and Path Planning 86
5.3 Q-Learning for Continuous Deadlines 87
5.3.1 Value Function Update Scheme 88
5.3.2 Function Approximation through Neural Network 88
5.3.3 Core Deployment 89
5.3.4 Other Practical Considerations 91
5.4 Experimentation 92
5.4.1 Artificial Network with Discrete Deadlines 92
5.4.2 Real Networks with Continuous Deadlines 94
5.5 Summary 98

6 Arriving on Time for Multiple Cooperative Vehicles **101**
6.1 Multiagent-based Route Guidance . . . 102
6.2 Route Assignment . . . 104
6.2.1 Route Assignment Formulation . . . 104
6.2.2 Bound Analysis . . . 106
6.2.3 Pseudo-Code Summary . . . 106
6.3 Further Improvements . . . 108
6.3.1 Refinement of Predicted Travel Time . . . 108
6.3.2 Improvement on Computational Efficiency . . . 109
6.3.3 Performance Improvement via Communication . . . 110
6.4 Experimentation . . . 111
6.4.1 Road Networks and Parameter Settings . . . 111
6.4.2 Comparative Performance . . . 112
6.4.3 Improved Performance . . . 117
6.5 Summary . . . 120

7 Conclusions and Future Work **121**
7.1 Conclusions . . . 122
7.2 Future Work . . . 124
7.2.1 Further Study of Arriving on Time for a Single Vehicle . . . 124
7.2.2 Further Study of Arriving on Time for Multiple Cooperative Vehicles . 126

Publications **127**

References **129**

Table Captions

3.1 Case 1: Accuracy for Independent Single Distribution (%) 39
3.2 Case 2: Accuracy for Combined Independent Probability Distribution (%) . . . 41
3.3 Case 3: Accuracy for Correlated Probability Distribution (%) 42
3.4 Computation Time for the Enumeration Method and ℓ_1-Norm Based Algorithms 43
3.5 Averaged Running Time for Different Standard Deviations 46
3.6 Averaged Running Time for Different Rtsots 47
3.7 Averaged Running Time for Different O-D Pairs 48
3.8 Averaged Running Time for Different Graph Scale and Data Size 50
3.9 Averaged Running Time for Different Deadline Coefficients 51
3.10 Computation Time for the ℓ_1-Norm Based Algorithms and State-of-the-Art Algorithm 54

4.1 Computation Time Complexity for the Real World Road Network 79

5.1 Settings of the Three Real Road Networks 94
5.2 Accuracy of Time-Dependent Q-learning (%) 97

6.1 Average Computation Time (s) 119

Figure Captions

1.1 Uncertain Factors for the Traffic on Roads 3

2.1 Probability Density Functions for Two Paths under LET Criterion 16

3.1 A 65-Node, 123-Arc Road Network. 36
3.2 Accuracy with Respect to Different Standard Deviations 45
3.3 Accuracy under Different Values of Rtsots 46
3.4 Accuracy with Respect to Different O-D Pairs 48
3.5 A 100-node, 220-arc road network. 49
3.6 Accuracy with Respect to Different Graph Scale and Data Size 50
3.7 An area of the Munich city, Germany 51
3.8 Accuracy for the ℓ_1-Norm Based Algorithm and State-of-the-Art Algorithm . . 53

4.1 Three Graphs of Small Scale 70
4.2 Average $f_{best}^{(k)}$ Over the Iterations 71
4.3 Computation Time with Respect to the B&B Method and the PLM_IPM Method 72
4.4 Computation Time with Respect to the PLM_IPM Method and the PLM_Dijkstra Method 73
4.5 Comprehensive Results on the Graph of Medium Scale 74
4.6 Beijing Road Network 76
4.7 The Average Minimum Number of Road Links between O-D for Each Scenario 76
4.8 Average $f_{best}^{(k)}$ Over the Iterations for Different Time Slots 77
4.9 Computation Time with Respect to Different Time Slots 77
4.10 Singapore Road Network 79

5.1 Artificial Network with Discrete Deadlines 93
5.2 Accuracy Results on the Three Real Road Networks with Continuous Deadlines 95
5.3 Accuracy Changes with Data Size and Iterations 96

5.4 Computation Time 98

6.1 Two Types of Agents and Intention Collection 103
6.2 Illustration of Refinement of Predicted Travel Time Function. 109
6.3 Two Testing Road Networks 111
6.4 Different Types of Deadlines on Singapore Map. 113
6.5 Different Types of Deadlines on New York Map. 114
6.6 Different Penetration Rates. 115
6.7 Different Compliance Rates and Different Percentages of Tight Deadlines. . . . 116
6.8 Refinement of Prediction and Improvement via Communication. 118

Abbreviations

SSP	Stochastic Shortest Path
LET	Least Expected Travel Time
SD-OSP	Spatially Dependent Online Shortest Path
TD-OSP	Temporally Dependent Online Shortest Path
TSD-OSP	Temporally Spatially Dependent Online Shortest Path
SSPD	Stochastic Shortest Path Problem with Delay Excess Penalty
SPOTAR	Shortest Path problem with On-Time Arrival Reliability
MILP	Mixed Integer Linear Programming
rtsots	Ratio of Training Size over Testing Size
O-D	Origin-Destination
PT Model	Probability Tail Model
B&B	Branch and Bound
PLM	Partial Lagrange Multiplier
IPM	Interior Points Method

Chapter 1

Introduction

Cars can facilitate people's travel, but can also cause heavy burden to traffic, which may in turn cause the delay and decrease the quality of people's life. To reduce the traffic delay, stochastic shortest path (SSP) problems are extensively studied. In this chapter, the existing solutions to the SSP problems regarding a single vehicle and multiple cooperative vehicles are briefly analyzed, respectively. After that, the proposed approaches in this thesis are listed to solve the limitations and disadvantages of the existing solutions. At last, the organization of the whole thesis is presented.

1.1 Background and Motivation

The last twenty years have seen a great increase in the number of cars as the technology develops and the society evolves forward. Nowadays, it is common for people to commute by driving a car and many families even own more than one car, which is very popular in the USA, Europe and China [1]. It is reported that in 2010, the global number of cars surpassed 1.015 billion, which was jumped from about 980 million in the year of 2009. It is also predicted that by 2035, the number of cars on the road worldwide will be up to 1.7 billion [2]. Cars can facilitate people's travel, but can also cause heavy burden to traffic, which may in turn decrease the quality of people's life. In fact, a great amount of drivers or other commuters are experiencing traffic congestion because of the increasing traffic level [3]. The Urban Mobility Report [4] pointed out that 498 urban areas in USA would have suffered an additional 865 million hours of delay in 2011. It also mentioned that in the same year, Americans spend 14.5 million hours every day stuck in traffic, trying to commute or move goods to market, and 2.9 billion gallons of fuel that summed up to $121 billion was wasted. Moreover, it is estimated that the costs of congestion in Britain, France, and Germany, had amounted to $200 billion in 2013. And this figure is expected to increase to $300 billion by 2030 [5]. Additionally, in Singapore, according to a study in 2012 by TomTom, a provider of in-car location and navigation products and services, it was revealed that the travel time of some congested roads was increased by 147% compared with the non-congestion situation [6], which caused inconvenience to people's life. Therefore, it is extremely important to reduce traffic delays and allow people to arrive on time, for the sake of both economy, environment and people's comfort.

Typically, there are two major ways to alleviate traffic delay: 1) constructing more road infrastructures in the crowded area to hold more vehicles, such as large overpass; 2) collecting more traffic data and exploring them more efficiently to provide high quality route guidance through navigation system [7]. Apparently, the latter is more economical and viable due to lower investment required in comparison with the former. On the other hand, although nowadays it is not challenging for a navigation system to find an optimal path in terms of shortest distance, shortest distance does not always guarantee reliable and acceptable travel time because traffic

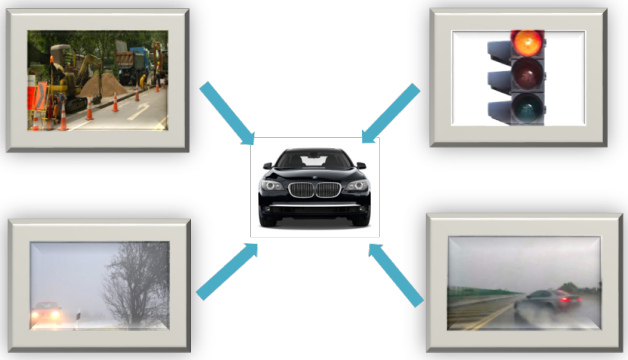


Figure 1.1: Uncertain Factors for the Traffic on Roads

is always random [8, 9, 10, 11, 12, 13, 3]. There are many uncertain factors on roads, e.g., unexpected road work, undesirable traffic lights, bad weather and traffic accident (see Fig. 1.1), which may cause longer travel time on a path of short distance. Therefore, more and more attentions are given to the stochastic shortest path (SSP) problem with respect to the vehicle routing, which aims to find an optimal path considering uncertainties on roads [14]. It attracts broad and extensive attentions from both industry and academia [15, 16, 17].

1.2 Stochastic Shortest Path Problem

The SSP problem for vehicle routing aims to find an optimal route in the random traffic, and the optimality criteria can vary due to the uncertain nature of traffic and user demands. Moreover, those criteria also depend on whether they are specified for a single vehicle independently, or multiple vehicles cooperatively. In particular, the major difference between SSP problems regarding an independent vehicle and multiple cooperative vehicles is whether they collect and explore the intentions (e.g., origin, destination and desired deadline to arrive at the destination) of others when calculating an optimal path for each vehicle [18, 19, 20].

1.2.1 SSP Problem for a Single Vehicle

Optimization methods have often been employed to solve the SSP problem for a single independent vehicle, where the optimal path solution for a vehicle usually does not explicitly rely on the intentions of other individuals [21]. In most of the optimization methods, the driver simply inputs the origin and destination[1], then a complete path would be returned to the driver [22]. The optimal path is usually achieved by utilizing the stochastic traffic data of road links, and minimizing certain costs or maximizing certain utilities [23, 24]. Moreover, in the SSP problem for a single independent vehicle, there are five typical criteria for the optimal path [17]:

- **Least expected travel time (LET)** - a path is optimal if it guarantees the minimum expected travel time.
- **Mean-risk model** - a path is optimal if it minimizes the sum of a linear combination of mean and variance regarding the travel time.
- **User-defined utility or penalty function** - a path is optimal if it maximizes the desirable utilities or minimizes relevant penalties for undesirable results.
- **Faster criterion** - a path is optimal if it guarantees a higher probability of reaching destination earlier than all the alternatives.
- **Probability tail model (PT model)** - a path is optimal if it maximizes the probability of reaching destination before the user-defined deadline.

The five optimal criteria have their own merits and demerits. The LET criterion is widely studied and applied mainly due to its high efficiency of computation. This is from the fact that LET-based SSP problem can be directly solved by the deterministic shortest path finding algorithms (e.g., the Dijkstra algorithm), where the weight on each arc refers to the mean of travel time [25], and the road network is expressed as a directed graph. However, the LET path may fail to meet drivers' expectation if there is a large variance (i.e., associated with the risk) of travel time. On the other hand, the criterion of mean-risk model is able to alleviate the risk issue since it incorporates the variance into the objective [26]. However, there are some other

[1] It may also include the desired deadline, which depends on the real user demand.

limitations with respect to the mean-risk model. In particular, there is no physical meaning of the weighted combination of expectation and variance, especially it lacks a dominate rule to decide the combination coefficients. Therefore, drivers may not be able to understand and gain actual expectation from the optimization solution. The criterion of user-defined utility or penalty function is flexible in providing various preferences to drivers, as those functions can be linear, quadratic, exponential and step, to express the desired and undesired expectations of drivers [27]. However, there is no dominated utility and penalty regarding this criterion, and the drivers are likely to be lost since they do not know which one is the most appropriate. In addition, normal drivers may not be able to understand the real meaning of these linear, quadratic, exponential and step functions. The faster criterion seems desirable since the chosen path is better than all the alternatives in the metric of probabilistic comparison [28]. However, to get the optimal solution, all possible comparisons are always needed, which renders it prohibitively inefficient, especially for enumerating all the probabilistic comparisons. The PT model is practical and promising in that it integrates travel time, risk (i.e., variance or standard deviation in the probability) and deadlines. Moreover, in many cases of route planning, driver utilities are always governed by deadlines [15, 29, 30], e.g., business meeting, fire rescue, organ delivery and catching up flight, all of which need arriving on time with the maximum chance. These critical cases, which are mainly driven by deadlines, render the SSP problem based on the other four criteria impractical [15]. Therefore, this thesis focuses on the PT model based SSP problem for a single independent vehicle, where the utilities are mainly governed by deadlines[2].

Unfortunately, the principle of optimality breaks down in the PT model based SSP problem, precluding dynamic programming approaches, which is NP-hard in nature [31]. A number of algorithms have been proposed to address this problem [32, 33, 29, 34], but are limited by strong assumptions. Those strong assumptions include Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines. However, they may not hold in real traffic, thus prevent the wide applications of those algorithms. Therefore, this thesis aim to address those limitations in the existing solutions. Note that, the

[2]The author does not claim that the PT model is dominantly better than the other four criteria. However, regarding the deadline driven scenarios, the PT model is much more promising.

PT model based SSP problem is also referred as the arriving on time problem by taking into account the definition of the PT model.

1.2.2 SSP Problem for Multiple Cooperative Vehicles

Quite a few multiagent-based computational methods have been applied to solve the SSP problem for multiple cooperative vehicles [35, 36, 37, 38, 39], because the agent metaphor for modeling a participant or decision-maker can capture complex constraints connecting all problem-solving phases [40, 41, 42], especially in cooperative vehicle routing [43]. In particular, a transportation system can be modeled as a large, distributed and dynamic multiagent system where vehicles represented as agents move on the road network following their own routes, which are determined by themselves or infrastructure agents at road intersections [42]. More importantly, intentions (e.g., origin, destination, time of route request and desired deadline) of the vehicles can be specifically and cooperatively explored in the context of multiagent, so as to efficiently respond to traffic dynamics and determine optimal route guidance for all relevant vehicle agents [44]. Here is a simple scenario to justify the importance of intention exploration in multiagent-based approaches. Assuming that 1,000 vehicle agents all request routes from the same origin to the same destination at the same time, the solution based on each independent vehicle may return the same route to all of them, which may cause traffic congestion when they are en-route. To the contrary, multiagent-based solutions are likely to distribute them to different routes to avoid the potential traffic congestion, by cooperatively exploring their intentions [45, 46].

In multiagent-based route guidance, LET paths are always provided to vehicle agents [47, 48, 49, 50, 51, 42, 52]. They consider LET paths mainly because the LET path for an individual vehicle is likely to result in less total travel time for all vehicles in the whole transportation system, which is environment friendly, considering that fuel consumption and air pollution are directly related with total travel time of vehicles [51, 42]. However, as mentioned earlier, drivers usually have deadlines in real traffic, and even the same driver may have different deadlines in various scenarios [15]. For instance, if they want to catch up important appointments,

their deadlines might be tight; if they go shopping, deadlines might be loose. Simply seeking LET paths for all drivers is not necessary and may cause some drivers with tight deadlines to miss their deadlines due to the influence from other drivers, especially those with loose deadlines. This will then increase the drivers' frustration and impatience, and, as a consequence, the accident rate [53]. Therefore, it is significant to consider arriving on time for all relevant vehicles cooperatively. On the other hand, although the PT model is promising as described in Section 1.2.1, current solutions always independently compute a path before the vehicle departs. Since traffic is always dynamic, optimality of a pre-computed path may not hold any more when all vehicles are en-route due to the influences from each other [54]. It is thus desirable to leverage both the advantages of decentralized multiagent-based approach (that deals with traffic dynamics by considering intentions of vehicle agents) [55] and the PT model (that considers different deadlines), by incorporating the latter into multiagent-based route guidance.

1.3 The Objectives and Proposed Approaches

In view of the above limitations and challenges for the SSP problem, the objectives of this thesis come to: (1) generalize the solution to the PT model based SSP problem regarding a single vehicle (i.e., circumvent the unrealistic assumptions in existing solutions) while preserving satisfactory performance with respect to the accuracy and computational efficiency; (2) incorporate the arriving on time criterion into the SSP problem regarding multiple cooperative vehicles, which should handle the traffic dynamics well and increase the chances of arriving on time for all relevant vehicles. To achieve these, this thesis develops novel optimization methods to maximize the probability of arriving on time for a single vehicle. Specifically, it firstly uses cardinality minimization to reformulate the arriving on time problem, then employs partial Lagrange multiplier method to improve computation efficiency of the reformulated problem, and finally adopts Q-learning method to increase the accuracy of finding the real optimal path. Additionally, a novel multiagent-based method is also proposed to address the arriving on time problem for multiple cooperative vehicles.

1.3.1 A Data-Driven Approach

The general solutions to the PT model based SSP problem suffer from several unrealistic assumptions: Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines [31, 56]. To get rid of these assumptions, this thesis proposes a data-driven approach [14, 17], which directly utilizes the travel time data of each road link. In particular, it formulates the optimal path finding as a cardinality minimization problem, which aims to minimize the frequency of being late on the optimal path regarding the desired deadline. This minimization problem is approximately solved by relaxing the cardinality via using ℓ_1-norm, which is further formulated as a mixed integer linear programming (MILP) problem. Thus, the arriving on time problem is solved using existing MILP solvers, e.g., the branch and bound (B&B) method [57]), and the desired path is naturally included in the MILP solution.

1.3.2 Improving Computation Efficiency via Partial Lagrange Multiplier

The above data-driven approach circumvents the unrealistic assumptions, but the existing MILP solver, i.e., the B&B method, is computationally inefficient if the sizes of travel time data and road network scale up. Therefore, this thesis takes the advantage of the total unimodularity (TU) [58] in the equality constraints of the reformulated MILP problem, which can return an integer solution even if the integer constraint is relaxed (i.e., it will become a linear programming (LP) problem), under the condition that there is no additional inequality constraint(s) [16]. Therefore, the partial Lagrange multiplier method [59] is employed, in which it shifts the inequality constraint to the objective function, and solves the corresponding problem using the sub-gradient method [60]. Consequently, an LP problem is solved in each iteration, which guarantees an integer solution, thus significantly improving the computation efficiency.

1.3.3 Increasing Accuracy via Q-learning

The ℓ_1-norm relaxation is only an approximation to the cardinality minimization. In order to further improve the accuracy of finding the real optimal path, while keeping satisfactory

computation efficiency, this thesis proposes a practical Q-learning method [61]. It is designed to maximize the probability of arriving on time regarding the user-defined deadline, which is approximated as the ratio between the number of times in which the vehicle reaches the destination before deadline, i.e., *success*, and the total number of travels. Specifically, each success event is considered as a *reward*, each pair of intersection and time-to-deadline (i.e., the remaining time to deadline) are considered as a *state*, and the driving direction at each intersection is considered as an *action*. Thus, the travel time data of road links could be directly utilized to calculate the reward, and the complete optimal path can be found by deciding a best action at each intersection. To handle the continuous deadlines and large-scale road networks, a practical value function approximation method [62] is developed based on the dynamic neural network [63] to directly learn the optimal Q-value, which represents the probability of arriving at the destination on time. Thus, the accuracy of returning the real optimal path would be improved.

1.3.4 A Multiagent-based Method

To adopt the PT model into the SSP problem for multiple cooperative vehicles, this thesis proposes a decentralized multiagent-based approach [53], which includes two types of agents: vehicle agent and infrastructure agent. Particularly, infrastructure agents locally collect intentions of concerned vehicle agents and formulate the route guidance as a route assignment problem [64], the objective of which is to increase the chances of arriving on time for all relevant vehicles. Moreover, the predicted travel time on each assigned road link is refined by iteratively linearizing a non-linear function to achieve more accurate approximation; the computational efficiency is enhanced by introducing additional variables and reformulating the route assignment as an MILP problem; and the overall performance is improved by allowing communication between neighboring infrastructure agents.

Note that, generally, the solution to the SSP problem regarding multiple cooperative vehicles can achieve better performance from the perspective of the whole transportation system than that of a single independent vehicle [65, 66]. This is because the former collaboratively calculates the optimal routes by exploring the collective information from individual vehicles, which

can better respond to the traffic dynamics when they are en-route [67]. However, although the V2X communication technique significantly advances [68], the solutions of SSP for multiple cooperative vehicles are still not ready for wide application. The major reason is that not every driver would like to share his/her intention to others, or follow the route guidance assigned by some infrastructures, especially in the case of making sacrifice (even tiny) to others [43]. Therefore, the solution to the SSP problem regarding a single independent vehicle is still in real demand by recent navigation systems, while the solution to multiple cooperative vehicles is likely to be applied in (near) future navigation systems.

In view of all the proposed approaches, the major contribution of the thesis is summarized as follows: (1) it relaxes the strong assumptions in the existing solutions to a single vehicle, such as Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines. At the same time, novel methods are developed to improve the accuracy and computation efficiency for the proposed approach. (2) Regarding the SSP problem for multiple cooperative vehicles, to the best knowledge of the author, it is the first time to integrate the arriving on time criterion into the multiagent based route guidance, which aims to increase the chance of arriving on time cooperatively for all relevant vehicles.

1.4 Organization

The organization of the whole thesis is as follows: **Chapters 1** and **2** introduce the research topic of SSP problems and discuss related work. Then, three methods are proposed to solve the arriving on time problem for a single independent vehicle, and improve its performance in **Chapters 3**, **4** and **5**, respectively. After that, a multiagent-based approach is proposed to solve the arriving on time problem for multiple cooperative vehicles in **Chapter 6**. At last, the conclusions and future work are given in **Chapter 7**. More specifically, the rest of this thesis is organized as follows:

- **Chapter 2** reviews related work on the general SSP problems for independent single vehicle and cooperative multiple vehicles, respectively. Regarding the former, it mainly discusses five criteria for the optimal path, among which the main focus is given to the

PT model (i.e., arriving on time). Regarding the latter, it mainly reviews the multiagent-based route guidance, which is popular in the cooperative vehicle route guidance.

- **Chapter 3** presents the data-driven approach to solve the PT model based SSP problem for an independent vehicle, which circumvents the three unrealistic assumptions described in Chapter 2. More specifically, this approach reformulates the path finding problem considering PT model as a cardinality minimization problem by directly utilizing the travel time data of each road link, the objective of which is to minimize the frequency of being late on the optimal path with respect to the user-defined deadline. Then, this cardinality minimization is approximately solved through ℓ_1-norm relaxation and its variants, which is further reformulated as an MILP problem. Thus, the arriving on time becomes solvable through the existing MILP solver. This data-driven approach is tested on an artificial network and part of the Munich road network, and the results show that the proposed approach can solve the arriving on time problem without the three unrealistic assumptions.

- **Chapter 4** proposes a partial Lagrange multiplier method, which aims to improve the computation efficiency of the PT model reformulated in Chapter 3. The proposed method explores the feature of total unimodularity that exists in the equality constraint of the above MILP problem, which guarantees integer solutions by solving the corresponding LP problem if there are no inequality constraints. Thus, the Lagrange multipliers are adopted to relax the existing inequality constraints by shifting them to the objective function, and only LP problems are solved afterwards, which guarantees polynomial computation, significantly saving computation time. To show desirable generalization, the proposed method is also applied to solve the SSPD model (i.e., the SSP problem with delay excess penalty) based problem. This method is tested on various artificial and real road networks, and the results verify the desired improvements of the proposed method regarding the computation efficiency.

- **Chapter 5** develops a practical Q-learning method, which mainly aims to improve the accuracy of finding the real optimal path, considering that the ℓ_1-norm relaxation is only approximation to the cardinality minimization in Chapter 3. This Q-learning method

is intended to maximize the probability of arriving on time, which is expressed as the ratio between the number of times in which, the vehicle reaches the destination before deadline, i.e., success, and the total number of travels. To address the issues of continuous deadlines and large-scale road networks, a value function approximation method is devised based on the dynamic neural network, to directly learn the probability of arriving on time regarding different deadlines and locations, which improves the accuracy of the returned solution. Simultaneously, desirable computation efficiency is also achieved. This Q-learning method is applied to three real road networks, i.e., Munich, Singapore and Beijing, and the results confirm the improvements on the accuracy.

- **Chapter 6** devises a decentralized multiagent-based method, which solves the arriving on time problem for multiple cooperative vehicles. In this approach, two types of agents are involved, i.e., vehicle agent and infrastructure agent, where each vehicle agent follows the route guidance by the local infrastructure agent, and the infrastructure agent formulates the local route guidance as an route assignment problem, which aims to increase the chances of arriving on time for all relevant vehicle agents. Moreover, the route assignment problem at each infrastructure agent is solved efficiently through existing solvers and the overall performance is enhanced through communications between neighboring infrastructure agents. This method is implemented into a traffic simulator based on some areas of two cities, i.e., Singapore and New York, and the results justify that the proposed multiagent-based method can increase the chances of arriving on time for cooperative vehicles.

- **Chapter 7** concludes the present research and outlines the future work for better solutions to the arriving on time problem. Briefly, regarding the single independent vehicle, there is still room to improve the accuracy and computation efficiency. In addition, more practical issues should be considered, e.g., transition time at intersections (including the time waiting for traffic lights), passing through fixed locations before reaching destination, and reducing turnings (at intersections) for the whole route. And extensive testing on real vehicles should be conducted. Regarding the multiple cooperative vehicles, incentive schemes should be investigated in order to encourage vehicles to follow the provided route guidance.

Chapter 2

Literature Review

In this chapter, an overview of related research on the general stochastic shortest path (SSP) problem regarding vehicle routing is presented, which helps to justify the direction and importance of the research work. In particular, Section 2.1 reviews and analyzes five optimal criteria pertaining to the SSP problem for a single vehicle, where more focus is given to the PT model; Section 2.2 reviews related work on the SSP problem for multiple cooperative vehicles; and Section 2.3 summarizes the limitations and unsolved issues in existing solutions. More specifically, the limitations regarding the SSP problem for a single vehicle include several unrealistic assumptions, such as Gaussian distribution of travel time, independence among travel time on different road links, and relatively large deadlines. Regarding the SSP problem for multiple cooperative vehicles, arriving on time is not considered in existing solutions. Consequently, this thesis is developed to address those problems.

2.1 SSP Problem for a Single Vehicle

The SSP problem for a single vehicle has been studied widely in the fields of operations research and traffic engineering, especially in the application of emergency response and disaster management [15]. In this problem, the objective is to determine an optimal path considering the uncertainties on roads. The corresponding solutions usually rely on the traffic data detected by roadside sensors [69, 70] or recorded by GPS devices of vehicles [71], and do not explicitly explore the intentions of other vehicles [21]. At the same time, in real applications, criteria for the optimal path are not unique because traffic is usually random, and different criteria are likely to result in different routing performance [8, 72, 73].

2.1.1 Least Expected Travel Time

In the SSP problem for a single vehicle, least expected time (LET) is often adopted as a criterion for path finding [9, 10, 11]. In particular, a path is optimal only if it guarantees minimum expected travel time. The travel time of each road link is assumed to always follow a certain type of distribution, e.g., normal, log-normal or bi-normal, where the mean travel time of each road link can be expressed by a fix value. In this case, finding the optimal path with the least expected travel time is equal to solving a deterministic shortest path problem on a graph (i.e, the underlying road network), where the weight for each arc (i.e., the road link) is its expected travel time. Therefore, the A^* or Dijkstra algorithm can be employed as a solution to find the optimal path. And there have been many researchers studying this problem and its variants [9, 10, 11].

Hooks and Mahmassani [9] address a problem of finding the LET path in a stochastic transportation network, where the travel times on all road links are denoted by random variables. To solve this problem in a time-dependent manner, two steps are proposed. Particularly, in the first step, considering each departure time during the peak hour, they compute the LET paths which originate from all origins to a fixed destination. In the second step, they calculate the lower bounds for the expected travel times of those paths obtained in the first step. After that,

the exact solution to the optimal path is easily derived based on this lower bound. Dean [10] solves a time-dependent LET path problem with waiting policy. He first studies the problem of searching the minimum-cost paths in a time-varying network, where the travel time of each road link is denoted by a function of departure time. He concludes that the waiting ability at some junctions may guarantee desired paths with less cost. This ability is directly associated with the time span that the driver may wait and the corresponding waiting cost at each intersection. In the case of discrete time, the general dynamic programming methods can be employed to efficiently solve this time-dependent SSP problem, which takes into account the waiting constraints. Waller and Ziliaskopoulos [11] address an online LET path problem which considers correlation between travel time of the road links. The correlation is characterized by spatial and temporal dependency, respectively. With respect to the spatial dependency, they propose an SD-OSP (Spatially Dependent Online Shortest Path) algorithm that takes into account one-step dependency regarding the location. With respect to the temporal dependency, similarly, they develop a TD-OSP (Temporally Dependent Online Shortest Path) algorithm that considers one-step dependency regarding the time. To solve the spatial and temporal road link cost dependencies together, they come up with a TSD-OSP (Temporally Spatially Dependent Online Shortest Path) algorithm by combining both the SD-OSP and TD-OSP algorithms. Then this solution is adopted to find the online optimal path in an iterative manner, which considers correlated traffic on different road links.

A major reason for researchers to adopt the LET path as optimal is that, under this criterion, the SSP problem or its sub-problem can be transformed into a deterministic path finding problem, which can be further solved by classic path finding algorithms. However, a route with minimum expected travel time may have a comparatively large variance of travel time, which is always associated with high risk. This can be illustrated by the two paths displayed in Fig. 2.1, where path 1 has a smaller mean with respect to travel time, and it is better than path 2 according to the LET criterion. However, path 1 has a higher variance, and compared with path 2, the driver on path 1 rarely traverses it with the expected travel time. Moreover, traveling on path 1 is also likely to result in longer travel time. Therefore, the LET criterion might be unreliable.

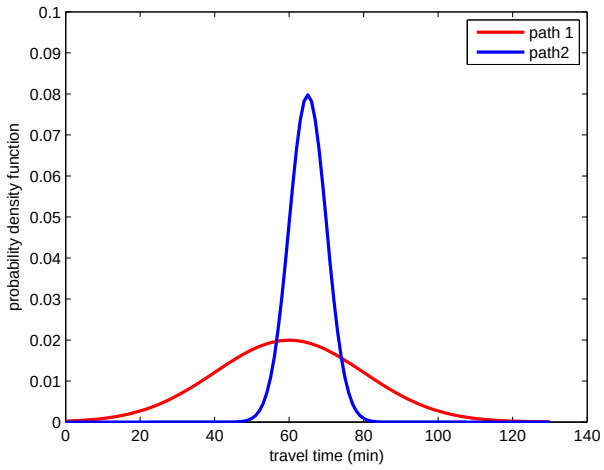


Figure 2.1: Probability Density Functions for Two Paths under LET Criterion

2.1.2 Mean-Risk Model

The mean-risk model is developed to solve the risk issue that appeared in the LET path [26, 74, 75]. This criterion seeks a path which minimizes the weighted combination of mean and variance of travel time (i.e., minimize $E(\vec{\mathbf{x}}) + \gamma Var(\vec{\mathbf{x}})$) regarding the path $\vec{\mathbf{x}}$. To find the optimal solution, it can be converted into a deterministic shortest path problem with arc weight being the combination of corresponding mean and variance of travel time.

Nikolova [26] discusses the mean-risk model in an application of driving to the airport under stochastic traffic. It is concluded through experimentation that the path purely minimizing expected travel time does not necessarily result in a satisfactory solution due to the large travel time variance (i.e., risk) on the LET path. Nikolova also theoretically proves that the objective to minimize the mean plus weighted standard deviation of a route can address the risk issue well. Then a quasi-convex optimization technique is proposed, which guarantees log-polynomial computation complexity to find a solution. Hutson and Shier [74] aim to find an optimal path in a stochastic road network, the optimality of which is defined as certain combination of mean and variance of travel cost, namely $z(P) = f(m(P)) + g(v(P))$. In particular, $f(\cdot)$ is an increasing convex function, $g(\cdot)$ is an increasing concave function, and P is a path. Both the mean and variance of travel cost (i.e., travel time) are additive functions if they are

assumed as independent distributions for relevant road links. The traditional labeling algorithm is not able to solve this problem since Bellman's principle of optimality does not hold in this case [76]. To address this issue, Hutson and Shier propose the concept of "extended dominance" based on the objective function, which satisfies a certain form of Bellman's optimality principle. Thus, a modified labeling algorithm could be adopted to compute the optimal path.

To some extent, the mean-risk model is able to address the risk issue in the SSP problem. However, it still has an obvious limitation: normal drivers may not be able to understand the physical meaning of this model and they probably have no idea about what the suitable weight coefficient is regarding the mean and variance of travel time.

2.1.3 User-defined Utility or Penalty Function

With respect to this criterion, the researchers usually define some utility or penalty functions of the travel cost (e.g., travel time). These functions can be linear, quadratic, exponential or step, which aim to maximize the desirable utilities or minimize the relevant penalty for undesirable results [77, 27, 78, 79, 80, 81].

Murthy and Sarkar [77, 27, 78] aim to solve a class of shortest path problems that maximize an expected user-defined utility. This utility function describes a situation where the utility of a path decreases at certain rate, until travel time on this path exceeds a predefined threshold. To find such an optimal path, they develop an efficient algorithm to achieve the exact solution, rather than using the monte-carlo based simulation technique to get an approximated solution. More specifically, they propose two different path pruning techniques incorporated in an enumeration scheme, which can find the desirable path with efficient computation. Nikolova, Brand and Karger [80] present an efficient approach for optimal route planning dealing with uncertainty, the objective of which is to minimize the value of an user-defined penalty function. In that approach, they adopt a decision theoretic framework to determine the optimal path. In particular, considering an origin O and a destination D in a road network, they seek an O-D path with the least expected travel time cost. And the road link travel time is presented by random variables and the cost is calculated by a nonlinear function of total travel

time. Then, based on the combinatorial optimization technique, they demonstrate an efficient solution through dynamic programming with polynomial computation. Cheng, Kosuch and Lisser [81] aim to address an SSP problem with delay excess penalty (SSPD) that minimizes the sum of a deterministic cost and an expected delay cost. More specifically, the delay cost is a function with respect to a user-defined deadline. To solve the problem without losing generality, they employ a branch-and-bound (B&B) framework to evaluate all the feasible paths between the O-D pair. In so doing, it helps to determine the lower bounds of optimum by solving a corresponding problem with a relaxed objective function. After that, this problem in turn is addressed by using a stochastic projected gradient algorithm, where an active set method is adopted to speed up the computation.

Pertaining to this criterion, a number of types of utility functions or penalty functions have been proposed. However, none of them can dominate each other, and drivers may not know which criterion is more appropriate. In addition, normal drivers may not even understand the physical meaning of those linear, quadratic, exponential or step functions.

2.1.4 Faster Criterion

The faster criterion is developed based on comparison, and a path is optimal if it guarantees faster arrival than all the other alternatives [28, 82, 83]. Moreover, the comparison is usually conducted based on some probabilistic metrics.

Sigal, Protsker and Solberg [28] aim to solve an SSP problem under the faster criterion. In particular, they conclude that it is often inadequate to only consider the expected travel time. On the other hand, it is desirable to make probabilistic statements regarding the comparison of candidate paths. In view of this, they define the optimal path as the one with highest probability that it is faster than all the other alternatives regarding the same O-D. More specifically, Sigal, Protsker and Solberg propose the concept of "path optimality index" to evaluate the quality of candidate paths. However, the way to compute the "path optimality index" in their approach is prohibitively time-consuming, because each candidate path needs to be explicitly compared with all the alternatives. Fastenrath and Becker [83] develop a new method to find the optimal

path considering the faster criterion, where they assume that only two traffic situations exist on each road link: congest or congestion-free. To compare two paths, they first figure out which path has a higher probability of being faster than the other, taking into account the situations of congestion and congestion-free. Then the winner is preserved to further compare with another candidate path in a similar way, and the loser will be discarded. This process is repeated until only one path is remained, which is considered as optimal.

The faster criterion seems desirable since the chosen path at the last step is better than all the other candidates according to a probabilistic metric. However, to get such an optimal solution, all possible comparisons are needed, which are prohibitively time-consuming. Thus, the inefficiency in computation renders this criterion impractical in real applications.

2.1.5 Probability Tail (PT) Model

To overcome the above limitations and demerits, the probability tail (PT) model, which considers travel time, risk and user-defined deadline, is proposed as an optimality criterion [84]. It defines the optimal path to be the one that maximizes the probability of arriving at destination before deadline (i.e., arriving on time). This criterion is reasonable in that it is consistent with people's route planning behavior. For example, people would like to request that "I plan to arrive at the airport in 40 minutes, please find a path with the maximum chance". Many researchers investigate this problem and various solutions have been subsequently achieved as in [15, 8, 32, 31, 56, 85].

Fan, Kalaba and Moore [15] address an SSP problem based on the PT model, where the direct goal is to achieve a policy which determines the next road intersection to visit instead of a prior complete path. To solve this problem, they use a class of *unknown* functions to indicate the maximum probability of arriving at destination before deadline, which are estimated by the Picard method of successive approximations. Thus the maximum probability is able to be evaluated without enumerating all possible paths. To evaluate the computation complexity of this approach, they introduce the Laplace transform and its numerical inversion to reduce the computation for evaluating the convolution integrals, which result from the successive approximation steps. Nie and Wu [32] solve a shortest path problem with on-time arrival reliability

(SPOTAR). They propose that SPOTAR can be addressed efficiently via identifying all local reliable paths, which are non-dominated under the stochastic dominance of first order. Then they demonstrate that SPOTAR could be formulated and solved via general dynamic programming. Further, Nie and Wu demonstrate that the dynamic problems are decomposable pertaining to certain desirable arrival times. Consequently, they conclude that introducing the time dimension (i.e., deadline) may not increase the computation complexity, although more data are required to specify the time-varying distributions. Nikolova et al. [31] consider a problem of finding the shortest path in a graph with independent randomly distributed arc lengths. The goal is to maximize the probability that the path length does not exceed a given threshold value t (i.e., it can also be a deadline if the length is travel time cost, thus same with the PT model). They develop an efficient algorithm using the quasi-convex maximization, which is based on the assumption that travel time follows an independent Gaussian distribution and t is not less than the mean of the smallest-mean path. In particular, when the departure time is closer to the deadline, so that any shortest path has the mean greater than t, the objective function will not be quasi-convex any more. In that case, the computation to get an optimum would be much time-consuming. Lim et al. [56] present a stochastic motion planning algorithm and its application to traffic navigation. The algorithm aims to determine a path between two points that optimizes a cost function of the delay probability distribution, which is equal to finding a path that maximizes the probability of reaching destination before deadline. For such a problem, classic shortest path algorithms do not work because the optimal substructure property does not hold. Consequently, they develop a novel approach based on the quasi-convex optimization technique to address this issue, which incorporates a cutting technique by exploring the means and variances of travel time for all relevant paths. Lim et al. evaluate the algorithm using both simulations and real-world driving data, which are gathered from a set of taxis equipped with GPS sensors and a wireless network. In comparison with the work in [31], the proposed algorithm by Lim et al. [56] saves much computation cost.

In comparison with other criteria, the PT model is more attractive. This thesis focuses on this criterion in the SSP problem for a single vehicle, basically due to two aspects: (1) it is easy for a normal driver to understand, and the general meaning of this model is to maximize the chance of arriving at destination before a user-defined deadline; (2) it is more consistent

with real world travel planning, i.e., people usually request that "I want to reach the airport by 08:00 pm. Please find a most reliable path". Among the aforementioned work to solve this arriving on time problem, the methods in [31] and [56] seem more desirable since they can efficiently find the accurate solutions. However, their high computation efficiency relies on several strong assumptions: (1) the travel time for each road link must follow a normal distribution; (2) the travel time distributions on different road links are independent from each other; (3) the deadline should be large enough, i.e., at least larger than the mean of the path with the least expected travel time. The first two assumptions are made to simplify the computation. However, in real world, the travel time on road links may not necessarily follow a normal distribution, and they might not be independent either. In fact, there are many studies in the literature challenging these assumptions, e.g., travel time following a skewed distribution is identified in [86]; travel time fitting a gamma distribution is concluded in [87]; a log-normal fitting for travel time is derived in [88], [89], [90] and [91]; a bi-normal distribution of travel time is claimed in [83], where one is for congestion situation and the other is for non-congestion situation; correlation for travel time on adjacent road links is also claimed in [89] and [90]. The third assumption is made to guarantee the quasi-convexity of the problem so as to speed up the computation. However, in real-world travel planning, people may not know whether their values of deadline are sufficiently large or not, especially when they are traveling on unfamiliar paths.

2.2 SSP Problem for Multiple Cooperative Vehicles

In the SSP problem for multiple cooperative vehicles, multiagent-based approaches have often been employed, where the intentions of vehicles can be explored in a sophisticated and cooperative manner [35, 36, 37]. An extreme but convincing motivation for intention exploration is the simultaneous route requests by a large number of vehicles, who request routes from the same origin to the same destination. The solution based on a single independent vehicle will allocate the same path to all of them, which is likely to cause traffic congestion. To the contrary, the solution based on the intention exploration may allocate them to different routes, which helps to avoid the potential congestion.

Most of the existing multiagent-based approaches rely on the LET paths for route guidance, which aims to reduce the total travel time of all vehicles in the whole transportation system. In one of the early multiagent-based approaches for route guidance [47], a global server agent constantly collects intentions of routes from all vehicle agents. It then computes a predicted LET path for each individual vehicle agent by cooperatively exploring the collected intentions, and all vehicle agents update their routes at each intersection accordingly. Based on the work in [47], Li et al. [48] predict the potential traffic on each road link at different time slots according to the collected intentions, then a corresponding LET path is constantly computed for each individual vehicle. Similarly, a modified A* algorithm [50] incorporates a repulsion scheme into the expression of weights on all road links. Then each vehicle agent recursively computes a LET path in a centralized manner, to avoid the situation where too many vehicle agents rush into a same route. Liang et al. [51] propose a personalized rerouting strategy by first ranking vehicles and then calculating a predicted LET path for a vehicle according to the decision of those who are ranked higher. In another centralized approach [66], each vehicle agent is assumed to know real-time traffic condition on all road links, and dynamically travels along the latest LET path. Centralized approaches often suffer from low computational efficiency.

Jiang et al. [42, 92] propose a decentralized pheromone-based vehicle rerouting approach, in which whenever congestion is predicted by a local infrastructure agent, the concerned vehicle agents will update their routes by choosing one of the best k LET paths. In another decentralized approach, Weerdt et al. [93] aim to minimize the delays at charging stations for electric vehicles, which incorporates the intentions into the probabilistic arrival time of those vehicles to each road link. To find the best location to charge, they maximize the expected utility function for each vehicle. Nevertheless, the results are only verified by the measurement of total expected travel time rather than whether vehicles arrive on time. Decentralized multiagent-based approaches have the capability of adaptively updating routes according to dynamic traffic.

However, the dominated multiagent-based methods do not consider specific demands of vehicle agents, i.e., preferred deadlines. As described previously, drivers may often have different deadlines for various traveling scenario, and it is important to take into account the criterion

of arriving on time in the multiagent-based route guidance. Nevertheless, most of the existing approaches incorporating the PT model presented in Section 2.1.5 independently pre-calculate a path for each individual vehicle before it departs, without considering the intentions of others. Since traffic is always dynamic, optimality of a pre-computed path may not hold any more when all vehicles are en-route due to the influences from each other [66]. It is thus desirable to leverage both the advantages of a decentralized multiagent-based approach and the PT model, and facilitate all the vehicles to arrive on time regarding their respective desirable deadlines.

The author would like to note that, the SSP problem for multiple cooperative vehicles is similar to the stochastic vehicle routing problem (SVRP) [94], considering that the optimal route for each vehicle can be cooperatively determined in both of the two problems. However, they are different in nature. The objective of the general SVRP is to minimize a vehicle fleet and the sum of travel cost, where the vehicle fleet are supposed to depart from their current locations to pick up a set of customers or goods at different locations, and then deliver them to their destinations [95]. Usually, in SVRP, each vehicle has a specified capacity that cannot be exceeded, and the amount of customers or goods at each location might be random, based on which the vehicle cooperatives with each other. Those characters essentially differentiate it from the general SSP problem for multiple cooperative vehicles. Although numerous solutions (e.g., exact, heuristic and meta-heuristic) to the SVRP have been proposed in the literature and are commercially available [95, 96, 97, 98], they are beyond the interest of this thesis. Therefore, the literature for the SVRP are not elaborated in this section.

2.3 Summary

With respect to the SSP problem for a single vehicle, the PT model is promising in that it integrates travel time, risk and deadline. However, the typical solutions to this problem are limited by several unrealistic assumptions. Consequently, a data-driven approach will be developed to eliminate these unrealistic assumptions in Chapter 3; its computation efficiency will be improved in Chapter 4; and its accuracy will be further increased in Chapter 5.

With respect to the SSP problem for multiple cooperative vehicles, a decentralized multiagent-based approach is desired. Although intentions of vehicle agents are cooperatively explored in the existing multiagent-based solutions, purely pursuing LET paths for all vehicle agents may cause undesirable results. Especially, in the scenario where deadlines are more valued, vehicle agents with tight deadlines are more likely to be delayed due to the influence of others. Therefore, the PT model will be adopted into the multiagent-based route guidance in Chapter 6, where a decentralized route assignment is proposed, with the objective to increase the chances of reaching destination before deadline for all vehicle agents.

Chapter 3

Arriving on Time for a Single Vehicle

In this chapter[1]*, a data-driven approach is proposed to solve the arriving on time problem for a single vehicle. In order to circumvent the three unrealistic assumptions in the existing solutions, the arriving on time problem is reformulated as a cardinality minimization problem. After that, the ℓ_1-norm is adopted to relax the cardinality, which further converts the original path finding problem into an MILP problem. Thus, this arriving on time problem becomes solvable via using an existing MILP solver (the B&B method). In particular, this approach does not need to assume a fixed distribution of travel time for each road link, which can also work well when travel time on different road links is correlated with each other. In addition, the proposed approach can efficiently handle different deadlines as well.*

[1]This chapter is developed based on my publications:1- Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. A Data-Driven Method for Stochastic Shortest Path Problem. 17th IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 1045-1052, 2014; 2- Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. Finding the Shortest Path in Stochastic Vehicle Routing: A Cardinality Minimization Approach. IEEE Transactions on Intelligent Transportation Systems (T-ITS), vol. 17, no. 6, pp. 1688-1702, 2016.

The remainder of this chapter is organized as follows. In Section 3.1, the arriving on time problem is mathematically described and formulated, and then is further transformed into a cardinality minimization problem. In Section 3.2, the ℓ_1-norm is employed to solve the cardinality minimization problem, two variants of which are also proposed. Then all of them are further reformulated as MILP problems. In Section 3.3, extensive experiments are performed on a simple network to justify the three advantages of the proposed approach. In Section 3.4, analysis for important parameters that may affect the performance of the proposed approach are provided. In Section 3.5, the proposed approach is further tested on a real road network with real traffic data. Section 3.6 states the summary for this Chapter.

3.1 Problem Formulation and Reformulation

In this section, a road network is first defined as a graph, based on which, the PT model based SSP problem (i.e., the arriving on time problem) is formulated. Note that, to find the optimal path, the proposed approach mainly relies on the travel time samples of road links, e.g., a vehicle took 100 seconds to traverse a road link, and another vehicle took 120 seconds to traverse the same road link. Generally, it is not easy to directly obtain those travel time samples (i.e., travel speed samples are much common). However, they can be achieved by analyzing the GPS trajectory data when vehicles travel on those road links [71, 99, 100].

3.1.1 Original Problem Formulation

A road network is modeled as a graph. Let $G = (V, A_r)$ be a directed graph, where $V = \{1, 2, \ldots, n\}$ represents the set of nodes and $A_r \subseteq \{(v, w) : v, w \in V, v \neq w\}$ represents the set of arcs, which also refer to the road links. More specifically, (v, w) means an arc from node v to node w. Then the SSP problem to maximize the probability of arriving at the destination d from the origin o not later than deadline T (i.e., T is actually the remaining time to deadline, and deadline is adopted for simplification in the whole thesis) can be formulated

as follows:

$$\begin{aligned} \max_{\vec{\mathbf{x}}} \quad & Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq T) \\ \text{s.t.} \forall v \in V : & \sum_{w \in V,(v,w) \in A_r} x(v,w) - \sum_{w \in V,(w,v) \in A_r} x(w,v) = \begin{cases} 1, \text{if } v = o, \\ -1, \text{if } v = d, \\ 0, \text{otherwise}, \end{cases} \end{aligned} \tag{3.1}$$

where $\vec{\mathbf{w}}$ denotes travel time samples for each arc, which can be obtained from GPS trajectories of vehicles [71, 99, 100]; $\vec{\mathbf{x}} \in \{0,1\}^{|A_r|}$, and its each component refers to an arc, e.g., $(w,v) \in A_r$ is on the concerned path if $x(v,w)$=1, and not on it if $x(v,w)$=0. Then this problem can be compactly written as follows [14]:

$$\max_{\vec{\mathbf{x}}} Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq T) \;\Big|\; \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}};\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}, \tag{3.2}$$

where $\mathbf{M} \in \mathbb{R}^{n \times |A_r|}$ is node-arc incidence matrix; $b \in \mathbb{R}^n$, whose elements are zeros except the sth and tth element, which are 1 and -1 (i.e., for o and d, respectively).

In general, the optimization problem in Eq. (3.2) is not convex or quasi-convex if further assumptions on travel time distribution, correlation and deadlines are not made. This means that there is no efficient method to solve this problem optimally. Alternatively, this chapter seeks to approximate the probability by using frequency based on the cardinality minimization problem. The proposed approach first rewrites the "maximizing" problem in Eq. (3.2) as "minimizing", which is expressed as follows:

$$\min_{\vec{\mathbf{x}}} Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} > T) \;\Big|\; \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}};\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}. \tag{3.3}$$

3.1.2 Problem Reformulation as Cardinality Minimization

Definition 1 *Cardinality is the number of non-zero elements in a vector or matrix [101], e.g., if* $\vec{\mathbf{x}} = [x_1, x_2, x_3] = [0, 0, 4]$*, then the cardinality of* $\vec{\mathbf{x}}$ *is 1.*

In the routing problem, the objective is to minimize probability of arriving at destination later than deadline (i.e., as in Eq. (3.3)). Therefore, it is equivalent to minimizing number of times

of arriving at destination later than deadline if size of travel time samples for each path is sufficiently large. For instance, if a driver travels 1000 times on two paths (e.g., path 1 and path 2) from o to d, and the frequency of arriving at d after deadline is 20 on path 1 and 10 on path 2, then path 2 is considered to be optimal. Thus, it formulates original problem as a cardinality minimization problem, which aims to minimize the cardinality of vector, i.e., $C(\cdot)$, defined as follows:

$$\begin{aligned} C(\vec{\mathbf{x}}) &= (c_1, c_2, \ldots, c_S) \\ &= \Big([\vec{\mathbf{w}}_1^\top \vec{\mathbf{x}} - T]^+, [\vec{\mathbf{w}}_2^\top \vec{\mathbf{x}} - T]^+, \ldots, [\vec{\mathbf{w}}_S^\top \vec{\mathbf{x}} - T]^+\Big), \end{aligned} \tag{3.4}$$

where $[\cdot]^+ = \max\{0, \cdot\}$; each component in $\vec{\mathbf{w}}_i$ denotes the i_{th} travel time sample on corresponding arc (i.e., it can simply assume that $\vec{\mathbf{w}}_i$ is the travel time of a vehicle who traverse the road network for the i_{th} time); S is size of travel time samples on each arc. Note that the cardinality minimization approach does not require any specific travel time distribution, correlation or deadline. Consequently, the objective to minimize probability of arriving at destination later than deadline can be reformulated as follows:

$$\min_{\vec{\mathbf{x}}} Card(C(\vec{\mathbf{x}})) \;\Big|\; \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}};\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}, \tag{3.5}$$

where $\vec{\mathbf{x}}$ is decision variable, denoting the optimal path. Thus, original problem to find an optimal path that maximizes probability of arriving at destination not later than deadline becomes a cardinality minimization problem. It is important to highlight that in this problem, travel time samples on arcs are adopted instead of probability distribution. When sampling size S is large enough, the frequency of not being late can be used to accurately approximate the probability of arriving on time.

3.2 Solution Method for Cardinality Minimization

In this section, the ℓ_1-norm minimization (and its variants) and mixed integer linear programming are presented. The former is used to transform the cardinality minimization problem while the latter is used to solve the transformed problem.

3.2.1 ℓ_1-norm Minimization to Solve Cardinality Minimization

Generally, cardinality minimization is neither classified as convex nor quasi-convex optimization, and a typical approach to solve it is to relax the problem by ℓ_1-norm. The relaxed problem can be solved efficiently, where ℓ_1-norm is usually known as convex envelop of the function $Card(\vec{\mathbf{x}})$ [102, 103].

ℓ_1-norm of a vector is denoted by $\|\cdot\|_1$, which is absolute sum of all its elements. For instance, ℓ_1-norm of the vector $\vec{\mathbf{x}}$ can be expressed as follows:

$$\|\vec{\mathbf{x}}\|_1 = |x_1| + \cdots + |x_n|, \tag{3.6}$$

where n is size of $\vec{\mathbf{x}}$. Accordingly, minimization of ℓ_1-norm for $\vec{\mathbf{x}}$ can be expressed as follows:

$$\min_{\vec{\mathbf{x}}} \|\vec{\mathbf{x}}\|_1 \quad \Big| \quad \vec{\mathbf{x}} \in \mathcal{F}, \tag{3.7}$$

where $\mathcal{F}$ is a feasible set.

3.2.2 Iterated Weighted ℓ_1-norm Algorithm

Inspired by the typical ℓ_1-norm minimization, another approximation of $Card(\vec{\mathbf{x}})$ can be expressed as follows [104]:

$$Card(\vec{\mathbf{x}}) = \lim_{\epsilon \to 0} \sum_{i=1}^{n} \frac{\log(1 + \frac{|x_i|}{\epsilon})}{\log(1 + \frac{1}{\epsilon})}, \tag{3.8}$$

with $\epsilon \geq 0$ and $\vec{\mathbf{x}} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top$. Without loss of generality, $\vec{\mathbf{x}}$ is decomposed and written as follows:

$$\vec{\mathbf{x}} = \vec{\mathbf{x}}_+ - \vec{\mathbf{x}}_-, \text{ where } \vec{\mathbf{x}}_+, \vec{\mathbf{x}}_- \geq 0. \tag{3.9}$$

Thus it has $Card(\vec{\mathbf{x}}) = Card(\vec{\mathbf{x}}_+) + Card(\vec{\mathbf{x}}_-)$. Then, it reformulates the cardinality minimization as follows:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{n} \log(1 + \frac{x_i}{\epsilon}) \Big| \vec{\mathbf{x}} \in \mathcal{F}, \vec{\mathbf{x}} \geq 0. \tag{3.10}$$

This approximation is closer to the real cardinality. Although Eq. (3.10) is still not a convex problem, it can be solved by 'difference of convex' programming [105]. Therefore, it further linearizes the objective function at current $x_i^{(k)}$:

$$\sum_{i=1}^{n} \log(1 + \frac{x_i}{\epsilon}) \approx \sum_{i=1}^{n} \log(1 + \frac{x_i^{(k)}}{\epsilon}) + \sum_{i=1}^{n} \frac{x_i - x_i^{(k)}}{\epsilon + x_i^{(k)}}. \tag{3.11}$$

This objective function can be solved in an iterative manner, and can be further expressed as follows [104]:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{n} w_i x_i \;\Big|\; \vec{\mathbf{x}} \in \mathcal{F}, \vec{\mathbf{x}} \geq 0, \tag{3.12}$$

where $w_i = 1/(\epsilon + x_i)$, which will be used as weight in next iteration until it converges to a local solution.

3.2.3 Reweighted ℓ_1-norm Algorithm

In Eq. (3.12), weight is updated in each iteration. Alternatively, to obtain a better weight for ℓ_1-norm, it can use the merit function [106].

Definition 2 *For any $\epsilon > 0$, a merit function $F_\epsilon(\vec{\mathbf{x}})$: $R_n \to R$ is separable and coercive of the form $F_\epsilon(\vec{\mathbf{x}}) = \sum_{i=1}^{n} \psi_i(|x_i| + \epsilon)$ for approximating the cardinality, where the functions ψ_i are strictly concave, i.e., strictly increasing and twice differentiable.*

Accordingly, the cardinality minimization problem can be more closely approximated as follows:

$$\min_{\vec{\mathbf{x}}} F_\epsilon(\vec{\mathbf{x}}) \;\Big|\; \vec{\mathbf{x}} \in \mathcal{F}. \tag{3.13}$$

The merit functions are not unique. A typical merit function to approximate the cardinality is defined as follows [106]:

$$F_\epsilon(\vec{\mathbf{x}}) = \sum_{i=1}^{n} \log(|x_i| + \epsilon) + \sum_{i=1}^{n} \log(|x_i| + \epsilon)^p, \tag{3.14}$$

where $0 < p < 1$. Similarly, to solve this merit function, the linearization technique can still be used. Then, it can reach the same formulation as in Eq. (3.12). The only difference is the way to compute the weight, which is described as follows [106]:

$$w_i = \frac{1 + (|x_i| + \epsilon)^p p}{|x_i| + \epsilon}, i = 1, \ldots, n. \tag{3.15}$$

3.2.4 Further Reformulated as an MILP Problem

The three algorithms in Sections 3.2.1, 3.2.2, and 3.2.3 can be further reformulated as typical mathematical programming, e.g., MILP. Only ℓ_1-norm minimization is taken as an example. For iterated weighted ℓ_1-norm algorithm and reweighted ℓ_1-norm algorithm, the only difference is that they are computed in an iterative manner with different weights as stated in Eq. (3.12) and Eq. (3.15), respectively.

The optimization problem in Eq. (3.5) is aimed to be solved, whose cardinality is defined in Eq. (3.4). Since ℓ_1-norm can be used to minimize the cardinality, it means that the sum of $\left([\vec{\mathbf{w}}_1^\top \vec{\mathbf{x}} - T]^+, [\vec{\mathbf{w}}_2^\top \vec{\mathbf{x}} - T]^+, \ldots, [\vec{\mathbf{w}}_S^\top \vec{\mathbf{x}} - T]^+\right)$ can be minimized instead. To make this objective function more compact, a non-negative intermediate variable ξ_i and some inequality constraints are introduced. Incorporating all above considerations, Eq. (3.5) can be reformulated as follows:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \xi_i \left| \begin{array}{l} \vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T \leq \xi_i;\ \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0;\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}, \end{array} \right. \tag{3.16}$$

where ξ_i is the intermediate variable associated with $[\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T]^+$ in Eq. (3.4), and $\vec{\mathbf{x}}$ is decision variable which refers to optimal path. By analyzing the optimization in Eq. (3.16), it is found that the ℓ_1-norm minimization can be easily transformed to an MILP problem, which is expressed as follows:

$$\min_{\vec{\mathbf{y}}} \vec{\mathbf{c}}^\top \vec{\mathbf{y}} \left| \begin{array}{l} \mathbf{A}\vec{\mathbf{y}} \leq \mathbf{B};\ \mathbf{A}_\mathrm{e}\vec{\mathbf{y}} = \mathbf{B}_\mathrm{e}; \\ \vec{\mathbf{l}} \leq \vec{\mathbf{y}} \leq \vec{\mathbf{u}};\ \vec{\mathbf{y}}_{(1:|A_r|)} \in \mathbb{Z}^{|A_r|}, \end{array} \right. \tag{3.17}$$

where

$$\vec{\mathbf{y}} = \begin{bmatrix} x_1 & \cdots & x_{|A_r|} & \xi_1 & \cdots & \xi_S \end{bmatrix}^\top, \tag{3.18}$$

$$\vec{\mathbf{c}} = \begin{bmatrix} 0_1 & \cdots & 0_{|A_r|} & 1_1 & \cdots & 1_S \end{bmatrix}^\top, \tag{3.19}$$

$$A = \begin{bmatrix} W_{11} & W_{12} \cdots W_{1|A_r|} & -1 & 0 & \cdots & 0 \\ W_{21} & W_{22} \cdots W_{2|A_r|} & 0 & -1 & \cdots & 0 \\ \vdots & \vdots \ddots \vdots & \vdots & \vdots & \ddots & \vdots \\ W_{S1} & W_{S2} \cdots W_{S|A_r|} & 0 & 0 & \cdots & -1 \end{bmatrix}, \tag{3.20}$$

$$\mathbf{B} = \begin{bmatrix} T & \cdots & T \end{bmatrix}^\top, \tag{3.21}$$

$$\mathbf{A}_\mathrm{e} = \begin{bmatrix} \mathbf{M} & \mathbf{0}_{|V|\times S}) \end{bmatrix}^\top, \tag{3.22}$$

$$\mathbf{B}_\mathrm{e} = \vec{\mathbf{b}}, \tag{3.23}$$

$$\vec{\mathbf{l}} = \mathbf{0}_{(|A_r|+S)\times 1}, \tag{3.24}$$

$$\vec{\mathbf{u}} = \begin{bmatrix} \mathbf{1}_{1\times|A_r|} & \boldsymbol{\infty}_{1\times S} \end{bmatrix}^\top. \tag{3.25}$$

Note that W_{ij} in Eq. (3.20) is the i_{th} travel time sample on arc j and $\mathbf{B}$ in Eq. (3.21) has the size of $S \times 1$.

In this MILP problem, $\vec{\mathbf{y}}$ is decision variable, and $\vec{\mathbf{x}}$ in Eq. (3.18) is optimal path. To solve this MILP problem efficiently, the *intlinprog* function in Matlab (version: R2014a) is used, which is based on the B&B method [107]. The steps are stated as follows [108]:

(i) *Initially reduce the problem size:* Linear program pre-processing is performed on dual problem, the purpose of which is to eliminate redundant variables or constraints.

(ii) *Solve the relaxed linear programming:* Interior point method [109] is employed to determine an optimal solution to the relaxed problem, which guarantees polynomial computation complexity.

(iii) *Tighten the LP relaxation of the mixed-integer problem:* Mixed-integer program preprocessing is conducted to analyze linear inequalities and determine whether some bounds can be tightened.

(iv) *Further tighten the LP relaxation:* Cut generation is implemented to restrict feasible region of the linear programming relaxations, to ensure that solutions are closer to integers.

(v) *Compute the integer-feasible solutions:* Heuristics are used to find feasible points for the branch and bound step below so that an upper bound on the objective functions can be determined [110].

(vi) *Systematically search for the optimal solution:* Branch and bound method [111] is constructed as a sequence of sub-problems that attempt to converge to a solution of the MILP, where sub-problems give a sequence of upper and lower bounds on the solution.

The *intlinprog* function can solve the MILP problem in any of the steps. In that case, $intlinprog$ does not execute the latter steps once it finds optimal solution. More details about these steps can be found in [108].

3.2.5 Analysis of Accuracy and Computation Complexity

Before analyzing performance of the proposed approach, its flow is summarized as follows: It originally aims to determine the path that maximizes probability of arriving at destination before deadline, but that optimization problem is neither convex nor quasi-convex without strong assumption, i.e, independent normal distribution for the travel time, which is usually contrary to the conclusions by many researchers. To solve this problem without such strong assumption, it uses frequency of arriving at destination before deadline to approximate corresponding probability. However, the frequency can be exactly represented by cardinality. Further, the typical way to solve cardinality minimization is ℓ_1-norm minimization or its variants. It is pointed out that, they only provide approximated solutions, because they are trying to approximate cardinality by ℓ_1-norm, and different variants may have different degrees of approximation. Despite these differences, each of the ℓ_1-norm based algorithms can be further exactly reformulated as a MILP problem, which can be solved by standard solvers.

Based on this summary, there are two critical issues that should be discussed with emphasis because they impact the performance of the proposed approach, i.e., accuracy and computation

complexity. It uses ℓ_1-norm based algorithms to approximately solve cardinality minimization problem, which results in the accuracy issue. Take iterated weighted ℓ_1-norm algorithm as an example. When ϵ is infinitely close to 0, Eq. (3.8) can provide 100%-accuracy solution to cardinality minimization. However, Eq. (3.8) cannot be solved directly. Thus, it is linearized in Eq. (3.11), which can be solved iteratively. Moreover, it may converge to a sub-optimal solution as iteration number becomes large [112], which also depends on the input data. However, higher accuracy usually comes at the cost of more iterations. Thus, in an actual application, it can seek the satisfactory balance.

The other important issue comes from MILP problem, which mainly affects computation time. As stated in Section 3.2.4, it employs the B&B framework to solve the MILP problem. In this framework, it iteratively branches at a new point, and the worst computation complexity to finish the branching is $\Theta(2^{|A_r|})$ [113], where $|A_r|$ is number of arcs. In each branch, there is a bounding problem, which involves a relaxed linear programming, and its averaged computation complexity is $\Theta((|A_r|+S)^3)$ [114], where S is number of inequality constraints in Eq. (3.17). Therefore, the worst complexity for the whole algorithm will be $\Theta(2^{|A_r|}(|A_r|+S)^3)$. However, the real computation time also depends on input data because it may affect the number of branching in MILP problem.

Since both accuracy and computation time depend on input data (e.g., travel time samples on each road link) and those data are always random in traffic, in the following sections, it first tests the proposed approach on an artificial road network with artificial data. Then, it analyzes various parameters that may influence the accuracy and computation time. Finally, it further tests the proposed approach in the real road network of partial Munich city with real traffic data and compare it with the state of the art approach.

3.3 Experimentation

In this section, performance of the proposed approach is evaluated through simulation. Particularly, it first divides travel time samples on all road links into training data set and testing data. Then it uses the proposed approach to compute an optimal path based on training

data, and compares it with exact solution. The exact solution is obtained by enumerating all possible paths and computing corresponding probabilities on testing data. The accuracy is then measured by comparing the exact solution with solutions derived from the proposed approach.

Note that, by 'training data', it is referred to the historic travel time samples on road links, e.g., collected from 5 days backwards till the current moment, and by 'testing data', it refers to the future traffic data, e.g., 24 hours from the current moment onwards. For a typical prediction problem in machine learning, it always uses training data to build a classifier and measures the accuracy based on the ground truth label of testing data. Similarly, in the stochastic shortest path problem, it plans to predict an optimal path for the upcoming driving. However, the metric of this optimal solution is the probability rather than a class label. In real applications, it is more reasonable to find the ground truth optimal path out of both the training data and testing data, instead of only testing data, and then compare it with the path obtained out of training data by the proposed approach. Moreover, also note that the accuracy obtained based only on testing data is usually not higher than that of on whole data set. This is the case in this Chapter, i.e., results in Table 3.1, Table 3.2 and Table 3.3. However, in order to show the lower bound accuracy of the proposed approach, it only uses testing data to find the ground truth optimal path and compare it with the solution computed by the proposed approach.

Moreover, the impacts of probability distributions, correlation and different deadlines are evaluated, which constitute three major advantages of the proposed approach over existing algorithms. In particular, the experiment are classified into three cases with respect to travel time on arc. In Case 1, it considers independent probability distributions. In Case 2, it considers combination of independent probability distributions. Then, in Case 3, it considers correlated probability distributions. While Case 1 and Case 2 help to show that the proposed approach is able to handle diverse or even unknown distributions, Case 3 can show that the proposed approach is able to address the correlation issue with respect to travel time. In all cases, the tests are performed with various of deadlines to show that the proposed approach works well for different deadlines.

It is highlighted that, for the sake of convenience of justifying the proposed approach, all experiments in Sections 3.3−3.5 are performed on a general PC with Intel Core i7-3540M processor

and 8.00 GB RAM. However, in real use cases, computation of the optimal path would be done by a powerful central routing engine server, which means that the computation speed in real use cases can be faster than that of the experiments in Sections 3.3−3.5. Additionally, all the optimal paths would be computed in an online manner because the traffic data on each arc may change dynamically.

3.3.1 Test Scenario: A Simple Network

The proposed approach is first tested on a simple road network to validate accuracy of the solution and to gain useful insights of the solution. Consider the 65-node, 123-arc network in Fig. 3.1, which is a directed graph. This is a fairly representative spatial network with the following features. Firstly, the graph contains cycles, and some arcs between two nodes are bi-directional. Secondly, there are some clusters in this graph which are connected with each other. The two features make the graph closely imitating real traffic networks.

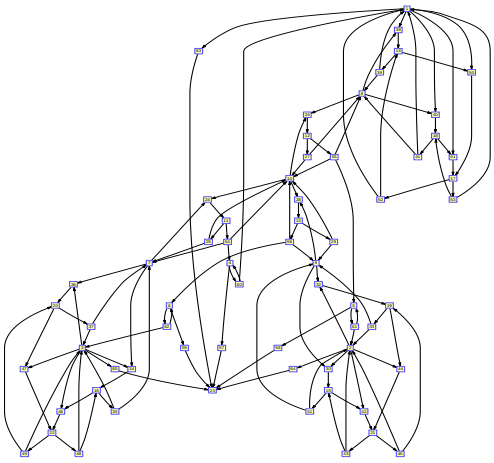


Figure 3.1: A 65-Node, 123-Arc Road Network.

The proposed approach is data-driven, which needs travel time samples on each arc as input (i.e., W_{ij} in Eq. (3.20)), and it adopts some random distribution functions to generate them. In the following experiments, it considers Normal, Bi-Normal, Gamma and Log-normal distributions and their combinations. It randomly selects an O-D pair out of ten: $\{$(3, 7), (7, 3), (1, 8), (8, 1), (10, 2), (2, 10), (1, 10), (10, 1), (3, 10), (10, 3)$\}$.

3.3.2 Description of Symbols and Settings

- $S -$ size of training data for travel time samples on each arc. It takes values of 100, 500, and 1000.
- $S_1 -$ size of testing data for travel time samples on each arc. It takes values of 40, 200 and 400.
- $\alpha -$ deadline coefficient with respect to T: $T = T_1 + \alpha * (T_2 - T_1)$, where T is deadline, T_2 is the minimum longest travel time for all paths connecting origin and destination, and T_1 is the shortest travel time with respect to the same path. It takes α=0.2, 0.4, 0.6, 0.8, 1.0 and 1.2.
- $N_E -$ number of times to run the proposed approach to reach target accuracy, which is 1000 in this experiment.
- $N -$ denotes Normal distribution.
- $Bi -$ denotes Bi-normal distribution.
- $G -$ denotes Gamma distribution.
- $L -$ denotes Log-normal distribution.
- $N + Bi -$ denotes Normal distribution combined with Bi-normal distribution.
- $N + G -$ denotes Normal distribution combined with Gamma distribution.
- $N + L -$ denotes Normal distribution combined with Log-normal distribution.
- O-D pair $-$ denotes origin and destination pair.
- $rtsots -$ denotes ratio of training size over testing size.
- ℓ_1-*norm* $-$ denotes ℓ_1-norm minimization algorithm.
- *iterated* ℓ_1-*norm* $-$ denotes iterated weighted ℓ_1-norm minimization algorithm.
- *reweighted* ℓ_1-*norm* $-$ denotes reweighted ℓ_1-norm minimization algorithm.
- Both *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms apply 10 iterations.

3.3.3 Case Study 1: Single Independent Distribution

For each arc, S training data and S_1 testing data are generated according to the four probability distributions. For each distribution, the data are generated with different parameters, and independent from each other. Then the S training data for each arc will be incorporated into Eq. (3.20) to compute the optimal path using three ℓ_1-norm based algorithms. Then, this path will be compared with the real optimal path which is obtained based on the S_1 testing data for each arc. To show that this result is only the lower bound accuracy, the accuracy measured on both training and testing data together is also computed for the ℓ_1-*norm* algorithm, as stated previously at the beginning of Section 3.3. Moreover, this process will repeat N_E times and the four corresponding accuracy results are shown in Table 3.1, structure of which is stated as follows:

(i) 1^{st} column stands for types of distributions;

(ii) 2^{nd} column stands for values of deadline coefficient α;

(iii) In each of the $3^{rd} - 5^{th}$ columns, there are four sub-columns. The first sub-column is accuracy for the ℓ_1-*norm* based on both training and testing data, the second to the fourth sub-column are respectively accuracy for ℓ_1-*norm*, *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* based on only testing data.

From Table 3.1, it is observed that the minimum accuracy for three ℓ_1-norm based algorithms is above 92%, and most of them are higher than 95%. The overall accuracy is sufficiently high considering the following facts. Firstly, optimal path is computed from the training data, while it evaluates the optimal path by testing data, and higher accuracy means better prediction. Secondly, ℓ_1-norm based algorithms are only approximations of cardinality minimization.

Comparing three ℓ_1-norm based algorithms with each other, it is observed that *reweighted* ℓ_1-*norm* always has higher accuracy than that of *iterated* ℓ_1-*norm*. Furthermore, *iterated* ℓ_1-*norm* has higher accuracy than that of ℓ_1-*norm*. Although there are some exceptions, which are highlighted with bold font in Table 3.1, they are already sufficiently high. Therefore, it is concluded that *reweighted* ℓ_1-*norm* is better than *iterated* ℓ_1-*norm*, and *iterated* ℓ_1-*norm* is

Table 3.1: Case 1: Accuracy for Independent Single Distribution (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
N	0.2	95.8, 95.5, 96.3, 97.2	96.8, 96.6, 96.9, 97.7	98.6, 98.3, 98.7, 99.1
	0.4	97.6, 97.3, 97.9, 98.1	99.5, 99.3, 99.5, 99.8	98.8, 98.6, 98.9, 99.3
	0.6	97.0, 96.9, 97.2, 97.9	98.1, 97.8, 97.9, 98.5	98.9, 98.7, **99.4**, **99.2**
	0.8	96.7, 96.3, 96.9, 98.4	97.7, 97.6, 98.2, 98.9	99.0, 98.5, 98.9, 99.3
	1.0	95.9, 95.6, 97.7, 98.2	96.3, 95.7, 97.9, 98.4	95.9, 95.7, 96.9, 98.4
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
Bi	0.2	92.9, 92.7, 93,8, 96.1	96.0, 95.7, 96.3, 97.5	97.5, 97.1, 97.4, 97.9
	0.4	95.5, 95.1, 95.8, 96.3	98.9, 98.3, 98.6, 99.2	98.9, 98.5, 98.6, 98.9
	0.6	96.0, 95.4, 96.2, 97.3	98.6, 98.1, 98.7, 99.1	98.8, 98.3, 98.8, 99.1
	0.8	93.9, 93.0, 94.5, 96.2	97.6, 97.0, 98.1, 98.6	97.3, 96.8, 97.1, 97.5
	1.0	93.2, 92.7, 93.8, 95.3	94.1, 93.3, 95.1, 95.8	96.2, 95.0, 95.8, 98.3
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
G	0.2	96.0, 95.7, 96.2, 97.1	97.8, 97.4, 97.9, 98.2	99.1, 98.7, 98.9, 99.4
	0.4	97.8, 97.4, 97.6, 98.2	99.3, **99.5**, **99.2**, **99.1**	99.5, **99.4**, **99.2**, 99.7
	0.6	97.8, 97.2, 97.5, 98.2	98.5, 98.2, 98.7, 99.1	99.1, 99.3, 99.5, 99.7
	0.8	95.1, 94.8, 95.3, 96.2	97.9, 97.3, 97.9, 98.2	98.1, 97.9, 98.4, 98.9
	1.0	96.0, 95.8, 96.4 96.9	96.2, 95.9, 96.5, 97.3	96.4, 96.0, 96.7, 97.3
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
L	0.2	95.9, 95.6, 96.3, 98.1	97.8, 97.1, 97.9, 98.4	98.1, 97.5, 97.9, 98.4
	0.4	95.4, 95.0, 95.8, 97.2	98.3, 98.1, 98.3, 98.9	99.3, 99.1, 99.3, 99.6
	0.6	97.2, 96.8, 97.2, 97.8	98.5, 98.3, 98.7, 99.0	98.9, 98.6, 98.8, 99.3
	0.8	94.5, 94.0, 95.5, 96.2	97.2 96.9, 97.1, 97.8	98.6, 98.0, 98.4, 99.0
	1.0	94.2, 93.9, 94.6, 96.7	94.8, 94.1, 94.9, 97.1	95.3, 94.6, 95.4, 96.7
	1.2	100, 100, 100, 100	100, 100 , 100, 100	100, 100, 100, 100

better than ℓ_1-*norm* in terms of accuracy. It is straightforward to understand this because of their different degrees of cardinality approximation.

It is observed that when deadline coefficient α is larger than 1, i.e., 1.2, which corresponds to the situation that there exists at least one path guaranteeing arriving on time with 100% probability, the proposed approach can always correctly find the actual optimal path. It is expected since ℓ_1-norm minimization in the proposed approach is to minimize total delay with respect to deadline T. If T is large enough, there always exists at least one path with zero probability of being late. When α is not larger than 1, i.e., 0.2, 0.4, 0.6, 0.8, 1.0, which corresponds to the situation that there is no path that can guarantee arriving on time with 100% probability, the accuracy always falls between 92%-100%. Regarding data size, it is observed that accuracy increases when data size S increases from 100 to 500. This is because that more data sampled, the closer frequency is to real probability. However, limited similar increase is

observed when S increases from 500 to 1000. One possible reason is that data size of 500 is already large enough and further more data does not enhance the accuracy.

Another notable result is that four different probability distributions share a similar pattern for the accuracy under different deadlines and data sizes. The main reason is that the proposed approach is directly based on data, and Eq. (3.17) only takes the sampled data into account, and it is not affected by the probability distribution used.

Last but not least, comparing the first sub-column with the second sub-column of the $3^{rd} - 5^{th}$ columns, it is observed that, most of the accuracy results measured on both training data and testing data together are higher than that of only on testing data. This is because in-sample test usually achieves higher accuracy than the out-of-sample test [115]. Although there are some exceptions, which are highlighted with underlines, they are still acceptable since accuracy in first sub-column and second sub-column are both high.

Based on above analysis, it can be basically concluded that the proposed approach is able to handle different independent distributions with different deadlines. Especially, when data size is large (i.e., S=500), the proposed approach can achieve high accuracy.

3.3.4 Case Study 2: Blended Distributions

For each arc, combinations of probability distributions are adopted to generate S training data and S_1 testing data. It first uses the sequence in incidence matrix $\mathbf{M}$ to order the arcs. Then, at each time, odd arcs use a probability distribution, and even arcs will use a different distribution. The combinations of probability distributions are set as follows: N+Bi, N+G, and N+L. It is also assumed that data on different arcs is independent from each other. Similar with Case Study 1, corresponding results are shown in Table 3.2.

From Table 3.2, it is observed that the minimum accuracy for three ℓ_1-norm based algorithms is above 89%, and most of them are higher than 93%. It is sufficiently high considering that the traffic data are mixture of different distributions. Besides, it is observed that *reweighted* ℓ_1*-norm* always has higher accuracy than that of *iterated* ℓ_1*-norm*, and *iterated* ℓ_1*-norm* has higher

Table 3.2: Case 2: Accuracy for Combined Independent Probability Distribution (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
	0.2	94.1, 93.6, 94.4, 95.2	97.3, 97.0, 97.6, 98.1	96.5, 96.2, 97.1, 97.8
	0.4	95.6, 95.0, 95.4, 95.9	99.1, 98.8, 99.1, 99.4	98.6, 98.2, 98.3, 99.1
N	0.6	96.4, 95.9, 96.8, 97.2	98.9, 98.8, 98.9, 99.2	98.9 **98.9**, **99.2**, **98.9**
+	0.8	96.1, 95.4, 96.3, 97.0	97.2, 96.4, 96.9, 98.0	97.6, 97.1, 97.4, 98.9
Bi	1.0	96.2, 95.5, 96.2, 96.9	95.2, 94.7, 95.2, 96.7	96.3 95.5, 96.3, 97.1
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
	0.2	93.1, 91.1, 92.6, 94.7	95.1, 94.6, 95.4, 96.2	96.0, 95.5, 96.3, 97.8
	0.4	93.4, 92.8, 93.8, 94.9	98.3, 97.6, 98.1, 98.5	98.2, 98.4, 98.9, 99.3
N	0.6	95.1, 94.0, 95.5, 96.3	96.9, 96.5, 97.2, 97.9	97.9, 97.5, 97.8, 98.5
+	0.8	91.2, 89.4, 92.4, 94.7	95.2, 94.8, 95.4, 96.1	96.3, 95.8, 96.2, 97.3
G	1.0	92.1, 89.5, **92.9**, **92.4**	91.3, 90.4, 91.9, 93.4	91.5, 90.6, 91.3, 92.7
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
	0.2	94.7, 94.2, 95.1, 96.0	97.2, 96.6, 97.1, 98.0	98.0, 97.3, 97.8, 98.1
	0.4	96.1, 95.7, 96.6, 97.2	98.3, 97.8, 98.3, 98.4	98.1, 97.7, 98.2, 99.0
N	0.6	96.0, 95.6, 96.3, 97.1	98.0, 97.5, 97.8, 98.9	98.5, 98.0, 98.6, 98.9
+	0.8	93.1, 92.5, 93.2, 94.7	96.3, 95.7, 96.0, 96.7	95.5, 95.4, 96.1, 96.8
L	1.0	92.0, 91.4, 92.7, 93.3	93.9, 93.2, 94.3, 95.0	91.4, 90.9, 92.1, 94.2
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100

accuracy than that of ℓ_1-*norm*. There are also very few exceptions highlighted with bold font, which are acceptable. Similarly, the accuracy results for ℓ_1-*norm* measured on both training and testing data together are higher than that of only on testing data. Although there is exception highlighted with underline, it is also acceptable. Comparing the results with Case 1, accuracy for Case 2 shares similar pattern under different deadlines and data size. From Table 3.2, the data-driven approach can accommodate blended distributions with different deadlines because the proposed approach is data driven, which is not affected by distribution types.

3.3.5 Case Study 3: Correlated Distributions

For each arc, S training data and S_1 testing data are first generated according to the four probability distributions. Then, it randomly chooses some adjacent arc pairs, the travel time on which are correlated with each other, i.e., the data on an arc is proportional to the other. Similar with the first two cases, corresponding results are shown in Table 3.3.

From Table 3.3, it is observed that the minimum accuracy for three ℓ_1-norm based algorithms is above 94%, and most of them are higher than 96%. The overall accuracy is sufficiently high

Table 3.3: Case 3: Accuracy for Correlated Probability Distribution (%)

	α	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
N	0.2	98.0, 97.6, 98.2, 98.7	97.9, 97.9, 98.3, 98.9	99.3, 98.8, 99.2, 99.5
	0.4	98.6, 98.1, 98.4, 99.2	99.3, 99.0, 99.2, 99.6	99.2, 99.0, 99.2, 99.7
	0.6	98.3, 98.0, 98.2, 99.0	99.5, 99.3, 99.6, 99.8	99.0, 98.8, 99.0, 99.4
	0.8	97.6, 97.1, 97.5, 98.2	98.6, 98.1, 98.4, 99.0	99.3, 99.0, 99.2, 99.5
	1.0	97.2, 96.6, 96.8, 97.4	97.2, 96.9, 97.2, 98.1	96.9, 96.5, 97.1, 98.2
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
Bi	0.2	96.8, 96.2, 96.8, 97.5	97.3, 96.8, 97.3, 98.5	98.1, 97.7, 98.0, 98.6
	0.4	97.6, 97.4, 97.8, 98.7	99.4, 99.2, 99.3, 99.6	99.5, **99.3**, **99.0**, **99.1**
	0.6	96.5, 96.2, 96.9, 97.5	98.4, 98.1, 98.5, 99.1	99.3, 99.5, 99.7, 99.7
	0.8	97.2, 96.9, 97.4, 98.3	98.9, 98.3, 98.9, 99.2	99.4, 99.3, 99.5, 99.5
	1.0	96.2, 95.9, 96.1, 97.4	96.4, 96.1, 96.7, 97.9	98.0, 96.9, 97.2, 98.1
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
G	0.2	98.2, 97.6, 98.2, 98.7	98.5, 98.2, 98.4, 98.9	98.9, 98.5, 98.8, 99.5
	0.4	98.6,**98.4**, **98.1**, **98.3**	99.5, 99.3, 99.4, 99.8	99.7, **99.7**, **99.3**, **99.4**
	0.6	98.7, 98.5, 98.7, 99.2	99.3, 99.2, 99.4, 99.5	99.5, 99.5, **99.7**, **99.6**
	0.8	97.5, 97.2, 97.8, 98.3	98.6, 98.2, 98.7, 99.3	98.9, 98.5, 98.9, 99.3
	1.0	96.5, 95.9, 96.2, 97.3	98.1, 97.4, 98.1, 98.7	97.8, 97.4, 98.2, 99.0
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100
L	0.2	97.0, 96.6, 96.7, 97.2	98.4, 98.1, 98.3, 98.8	98.7, 98.1, 98.5, 98.9
	0.4	97.6, 97.4, 97.7, 98.1	98.9, 98.6, 99.1, 99.3	99.3, 99.2, 99.3, 99.7
	0.6	97.9, 97.3, 97.5, 97.9	99.3, 99.1, 99.3, 99.4	98.9, 98.8, 98.9, 99.3
	0.8	96.2, 95.8, 96.4, 97.5	96.5, 95.9, 96.2, 97.3	98.2, 97.8, 98.2, 98.8
	1.0	95.3, 94.6, 95.3, 95.9	96.1, 95.3, 95.7, 96.5	96.3, 95.7, 96.1, 96.8
	1.2	100, 100 , 100, 100	100, 100 , 100, 100	100, 100, 100, 100

considering that existing methods cannot address the correlation in probability distribution. Comparing the results with Case 1 and Case 2, Case 3 share similar pattern with them. However, accuracy for Case 3 is slightly higher than that of Case 2 on average. This is because, in general, compared with correlated distribution, the blended distributions always generate larger variance for travel time samples on a path. Since accuracy results in Case 2 and Case 3 are obtained based on a learning scheme, training data and testing data might not be similar with each other if the variance is too large, which generally is not desirable to achieve high accuracy. Thus, higher accuracy for Case 3 is observed compared with Case 2. Similar explanation can also be applied in analysis of distribution parameters in Section 3.4.1. From Table 3.3, the proposed approach can address correlation issue well.

3.3.6 Computation Time

In all previous cases, to determine whether the proposed approach can achieve an optimal path, it uses enumeration method to compute the actual optimal path. To evaluate the overall computation complexity, it records all the running time for the above experiments. Additionally, the average running times for different sizes of travel time data and different algorithms are shown in Table 3.4.

Table 3.4: Computation Time for the Enumeration Method and ℓ_1-Norm Based Algorithms

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
enumeration method	0.6419	0.6389	0.6124
ℓ_1-*norm*	0.0546	0.2042	0.4052
iterated ℓ_1-*norm*	0.3412	0.7308	1.7123
reweighted ℓ_1-*norm*	0.4020	0.8736	1.9052

From Table 3.4, it is observed that the average running time for ℓ_1-*norm* is always shorter than that of enumeration method under different sizes of travel time data. The important reason is that MILP can be solved more efficiently than enumeration method. In this table, computation time for enumeration method is relatively constant because it depends mainly on the network size. Although running time always increases with size of travel time data for ℓ_1-*norm* algorithm, a very large size of travel time data is not needed to obtain satisfactory solutions based on conclusions for the 3 cases that studied. This means that it can obtain an optimal path faster than enumeration method.

Comparing *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms with enumeration method, it is observed that when size of training data is 100, the running times for *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms are shorter than that of enumeration method. When data size increases to 500 and 1000, it has a reverse effect. However, one fact is that when network size increases, the running time and storage space will become prohibitively large for enumeration method (see Section 3.4.4). By contrast, the computation time for *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms will also increase as network scales up. Nevertheless, they are

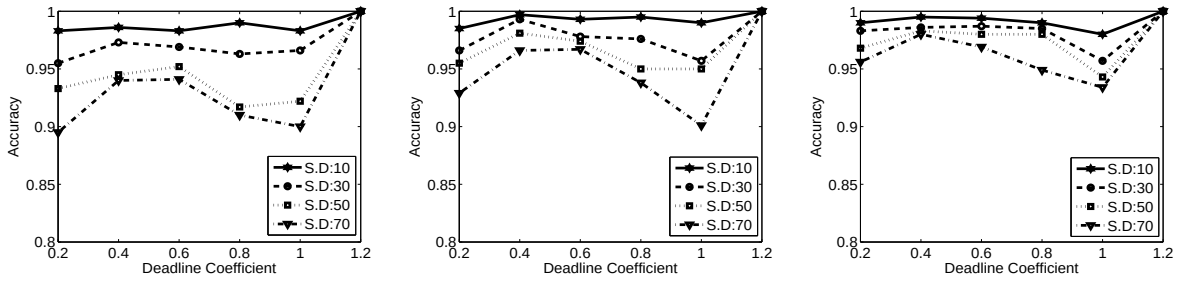
still acceptable since the two algorithms are still more efficient with respect to computation time in that case (see Section 3.4.4).

Comparing three ℓ_1-norm based algorithms with each other, it is observed that *iterated* ℓ_1*-norm* and *reweighted* ℓ_1*-norm* algorithms have higher computation time. This is because the two algorithms involve 10 iterations for each optimal path. For ℓ_1*-norm*, it inherently solves only one MILP problem. For *iterated* ℓ_1*-norm* and *reweighted* ℓ_1*-norm*, they compute 9 relaxed MILP (linear programming in nature) problems before the last real MILP. Since the computation complexity for linear programming is much less than that of MILP, it is much less than 10 times of computation time of ℓ_1*-norm* algorithm.

The accuracy of ℓ_1*-norm* algorithm in all 3 cases is sufficiently high (although not the highest) and *iterated* ℓ_1*-norm* and *reweighted* ℓ_1*-norm* algorithms have much higher computation time, although they have comparatively higher accuracy. Therefore, it is concluded that ℓ_1*-norm* has better overall performance.

3.4 Parameter Analysis

In Section 3.3, simulations are performed to evaluate the accuracy and computation time of the proposed approach, which involve various parameters. In this section, influences on the performance of the proposed approach are investigated, i.e., accuracy and computation time. The *iterated* ℓ_1*-norm* and *reweighted* ℓ_1*-norm* algorithms are comparatively complicated, but they are both developed based on ℓ_1*-norm*. Therefore, it only takes the latter as an example. For all experiments in this section, at each time, it randomly generates travel time samples for each arc. To obtain accuracy and averaged running time, it repeats the experiment for 1000 times. Moreover, note that in this section, the accuracy are measured only based on testing data, which usually results in the lower bound, as justified in Section 3.3.



3.2.a data size: 100 3.2.b data size: 500 3.2.c data size: 1000

Figure 3.2: Accuracy with Respect to Different Standard Deviations

3.4.1 Distribution Parameters

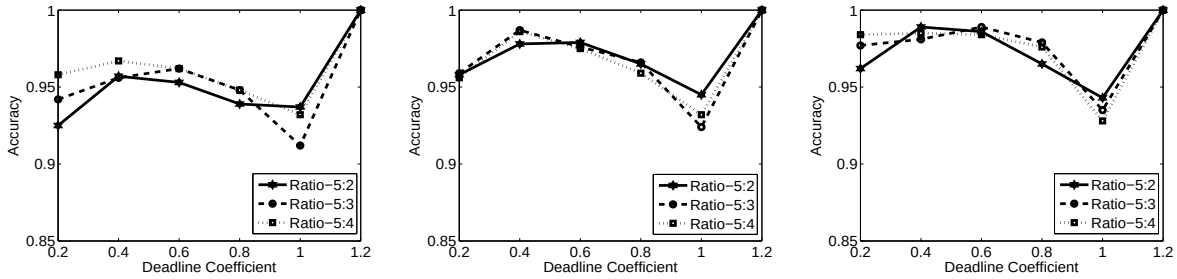
Generally, two parameters are very important in the probability distributions, i.e., mean and standard deviation. To investigate their influence on the performance, only normal distribution is taken as an example. The proposed approach varies the standard deviation, i.e., 10, 30, 50 and 70, while keeping other parameters fixed (e.g., O-D pair and ratio between the sizes of training and testing data). The accuracy results are shown in Fig. 3.2.

From Fig. 3.2, it is observed that most of accuracy is still above 90% (except one instance at 89%) for standard deviation of 70. It is also observed that as standard deviation increases, accuracy decreases under same deadline coefficient and data size. It is not difficult to understand that if standard deviation is 0, the accuracy by ℓ_1-*norm* will be 100% because it becomes a deterministic problem. If standard deviation is large, the similarity between training data and testing data greatly decreases, and accordingly, the accuracy will be detrimental. Additionally, the averaged running time is shown in Table 3.5 for different standard deviations and data sizes.

From Table 3.5, it is observed that computation time increases as standard deviation increases from 10 to 50. This is due to the fact that when standard deviation is 0, there is only one effective inequality constraint in the MILP problem. Thus, computation complexity is small. However, computation complexity may not always increase as standard deviation continues to increase, which can be observed when standard deviation changes from 50 to 70. This is

Table 3.5: Averaged Running Time for Different Standard Deviations

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
10	0.0403	0.1993	0.4018
30	0.0546	0.2042	0.4052
50	0.0599	0.2436	0.4872
70	0.0571	0.2630	0.4659



3.3.a data size: 100 3.3.b data size: 500 3.3.c data size: 1000

Figure 3.3: Accuracy under Different Values of Rtsots

because that the input data also affects computation time, but there is an upper bound for the complexity to solve MILP problem (see Section 3.2.5).

3.4.2 Ratio of Training Size over Testing Size

In the proposed approach, the accuracy is computed based on prediction, which is similar to the prediction problem in machine learning. In machine learning, ratio of training size over testing size ($rtsots$) is an important parameter to evaluate the accuracy. Accordingly, its influence on the proposed approach is investigated. It varies the value of $rtsots$ from 5:2 to 5:4 while keeping other parameters fixed. The results are shown in Fig. 3.3.

From Fig. 3.3, it is observed that there is no clear trend that accuracy will be affected by $rtsots$. This can be explained by combining two points. Firstly, the problem is similar to the prediction in machine learning, and according to the machine learning theory, more training data and less testing data usually lead to higher accuracy. This means that, in the problem, higher $rtsots$

may achieve better accuracy. Secondly, on the other hand, the question is not exactly same as the machine learning problem, because metric of this stochastic optimal solution is probability. When testing data size is comparatively large, it approximates the probability better and share a similar pattern with training data (i.e., similar probability of not being later than deadline). It means that lower $rtsots$ may achieve comparatively higher accuracy, especially when traffic data size is large enough. When these two points compromise with each other, a clear pattern regarding accuracy and $rtsots$ may not be observed. Meanwhile, it also records the averaged running time with respect to different $rtsots$ in Table 3.6.

Table 3.6: Averaged Running Time for Different Rtsots

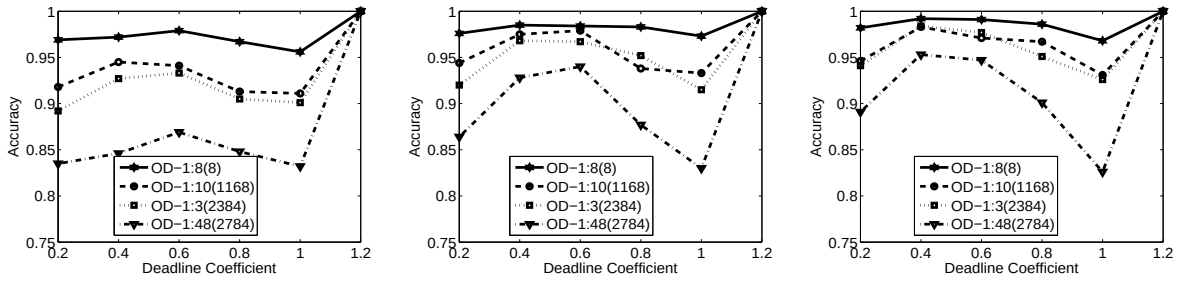
	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
5:2	0.0395	0.1836	0.3769
5:3	0.0382	0.1963	0.3721
5:4	0.0419	0.1880	0.3694

As expected, obvious patterns between $rtsots$ and computation time are not observed in Table 3.6, because optimal path is obtained from training data while it is validated on testing data. Thus, the running time will not be influenced by $rtsots$.

3.4.3 O-D Pair

The locations of O-D pair are also important parameters that may impact accuracy. The influence is investigated by changing O-D pair while keeping other parameters unchanged. It chooses following O-D pairs: 1-8, 1-10, 1-3 and 1-48, and there are respectively 8, 1168, 2384 and 2784 different connected paths between them. The results are shown in Fig. 3.4.

From Fig. 3.4, it is not observed that for most cases, as number of connected paths increases, the accuracy decreases except deadline coefficient of 0.8 in Fig. 3.4(b) and 0.6 in Fig. 3.4(c). It is reasonable considering that if there is only one possible path between O-D pair, the accuracy will always be 100%. Since the proposed approach is an approximation and traffic data is all random, if there are a large number of candidate paths, the chance that the proposed solution



3.4.a data size: 100 3.4.b data size: 500 3.4.c data size: 1000

Figure 3.4: Accuracy with Respect to Different O-D Pairs

is the actual optimal path will decrease. However, in real world road networks, there may not be too many connected paths between the concerned O-D pair for driver to choose. Therefore, accuracy will not be degraded significantly. Moreover, the averaged running time for different O-D pairs is recorded in Table 3.7.

From the first three O-D pairs in Table 3.7, it is observed that when number of connected paths between O-D pair becomes large, averaged running time accordingly increases. It is reasonable because if there is only one path between the pair, running time should be very short. If the number is large, it will take longer time to search the best path. However, computation time may not constantly increase, because MILP solver adopts efficient algorithms to find optimal solution although the worst case (exponential computation complexity) is possible to happen.

Table 3.7: Averaged Running Time for Different O-D Pairs

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
1-8 (8)	0.0280	0.1352	0.3102
1-10 (1168)	0.0390	0.1958	0.3798
1-3 (2384)	0.0452	0.2210	0.3941
1-48 (2784)	0.0443	0.2324	0.3886

3.4.4 Graph Scale and Data Size

To analyze the influence brought by graph scale and data size, the proposed approach is implemented on another comparatively large artificial network with 100 nodes and 220 arcs. For this large network, as shown in Fig. 3.5, it randomly selects O-D pairs out of (40, 43) and (40, 51), whose numbers of connected paths are respectively 2,403,060 and 4,529,052. Regarding other settings, they are the same as previous simulations for the small road network in Fig. 3.1. Accordingly, the accuracy for these two graphs are plotted in Fig. 3.6.

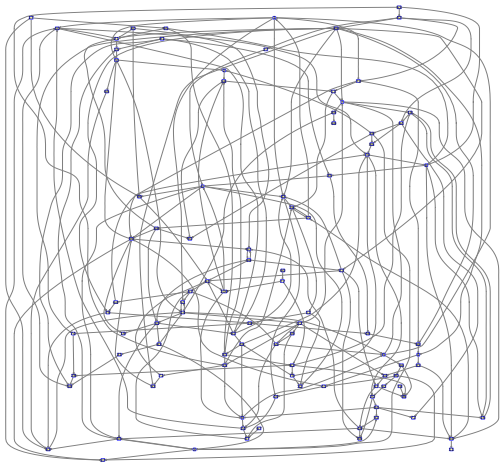
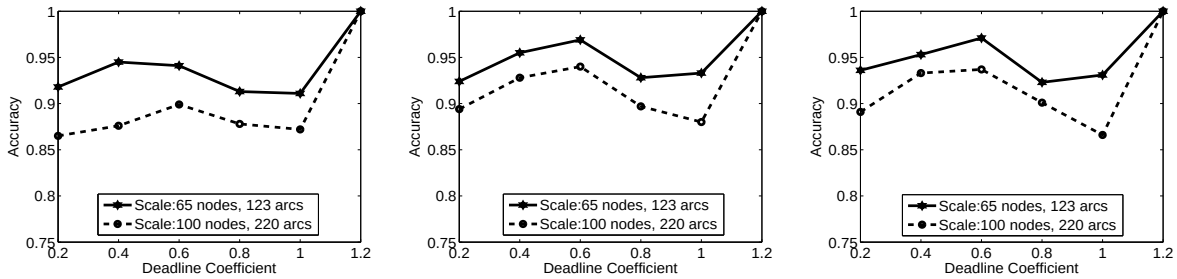


Figure 3.5: A 100-node, 220-arc road network.

From Fig. 3.6, it can be seen that graph scale affects accuracy and small graph can achieve comparatively higher accuracy. Note that the average number of connected paths for small graph is around 2,000 while large graph is around 3,000,000. Moreover, the reason for different accuracy can be explained by the same analysis for O-D pair. As for the influence caused by data size, it has been analyzed previously, and the accuracy could improve as data size becomes large, i.e., from 100 to 500. However, it becomes saturated, i.e., from 500 to 1000. Additionally, more obvious impact on the proposed approach caused by graph scale and data size is computation time, which is recorded in Table 3.8.

From Table 3.8 it can be seen that enumeration method becomes prohibitively time consuming as graph scales up, which uses more than 4,000 seconds to compute optimal path.



3.6.a data size: 100 3.6.b data size: 500 3.6.c data size: 1000

Figure 3.6: Accuracy with Respect to Different Graph Scale and Data Size

Table 3.8: Averaged Running Time for Different Graph Scale and Data Size

	S=100, S_1=40	S=500, S_1=200	S=1000, S_1=400
enumeration for large graph	4670.5	4830.1	4781.5
ℓ_1-*norm* for large graph	0.7207	0.9816	1.3198
enumeration for small graph	0.6980	0.6932	0.6970
ℓ_1-*norm* for small graph	0.0539	0.2267	0.3996

The obvious reason is that the number of connected paths between O-D pair is huge on this large graph and enumeration does not work efficiently. By contrast, computation time of ℓ_1-*norm* on large graph is only around 1 second, which is considerably efficient because it uses smart optimization techniques to search the desired path. However, considering ℓ_1-*norm* alone, computation time also increases as graph and data size scale up. This is reasonable as analyzed in Section 3.2.5, the proposed approach is based on MILP, which is solved by branch-and-bound method. Additionally, the worst complexity for branch-and-bound method is $\Theta(2^{|A_r|}(|A_r|+S)^3)$, which increases as network and data size scale up. However, as observed from all previous experiments, the worst case seldom happens.

3.4.5 Deadline Coefficients

Deadline coefficient impacts accuracy, which can be justified from Fig. 3.2 - Fig. 3.4 and Fig. 3.6. Especially, for coefficient larger than 1, i.e., 1.2, accuracy always reaches 100%. The

Table 3.9: Averaged Running Time for Different Deadline Coefficients

	100, 40	500, 200	1000, 400
0.2	0.0240	0.1561	0.2983
0.4	0.0367	0.1683	0.2836
0.6	0.0344	0.1625	0.3041
0.8	0.0436	0.1438	0.3249
1.0	0.0359	0.1583	0.3039
1.2	0.0408	0.1469	0.3329

detailed reason can be found in the case studies presented in Section 3.3. As to coefficients between 0 and 1, there is no regular pattern for the accuracy. Moreover, to show the impact on computation time, it also records the averaged running time in Table 3.9 for different deadline coefficients while keeping other parameters unchanged. From Table 3.9 it is observed that, computation time does not change with deadline coefficients regularly. This is reasonable because in the MILP problem, deadline coefficient impacts only inequality constraints, but does not change the number of these constraints. By contrast, it may impact the branching operation in MILP, which can be evaluated only by the worst case scenario.

3.5 Testing on Real World Road Network

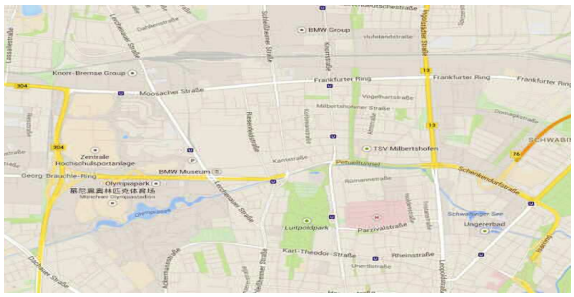


Figure 3.7: An area of the Munich city, Germany

In this section, the proposed approach is evaluated using an area of Munich city, which is shown in Fig. 3.7. The underlying graph includes 170 nodes and 277 arcs. The travel time samples

on each road link are extracted from real GPS trajectories of BMW vehicles[2]. The numbers of those samples on each road link are not same, the minimum of which is 880 and the maximum is 41,256. Other settings are similar to that in Section 3.3. Since the input data is from real GPS trajectories, there are no assumptions of probability distribution or correlation. Morevoer, it synthesizes the input data as follows:

(i) Sort the data for each arc according to the time when they were generated;

(ii) Divide the data for each arc into two parts: the first 80% forms training data set, and the remaining forms testing data set;

(iii) For each arc, randomly sample S data from first part as training data and S_1 data from second part as testing data, and set $S : S_1 = 5 : 3$;

(iv) Repeat 1000 times to obtain accuracy measure.

Note that it measures accuracy based only on testing data instead of training data and testing data together, which usually results in lower bound accuracy, as justified in Section 3.3. Additionally, it implements the state-of-the-art algorithm in the same network. The algorithm assumes that travel time for each road link follows an independent normal distribution. If there exists at least one path whose expected travel time is no longer than deadline, this problem can be solved by quasi-convex optimization technique efficiently. Otherwise, it can only find an optimal solution by an inefficient enumeration method. Its details are described in [56]. For better comparison, this method is adopted according to the following settings:

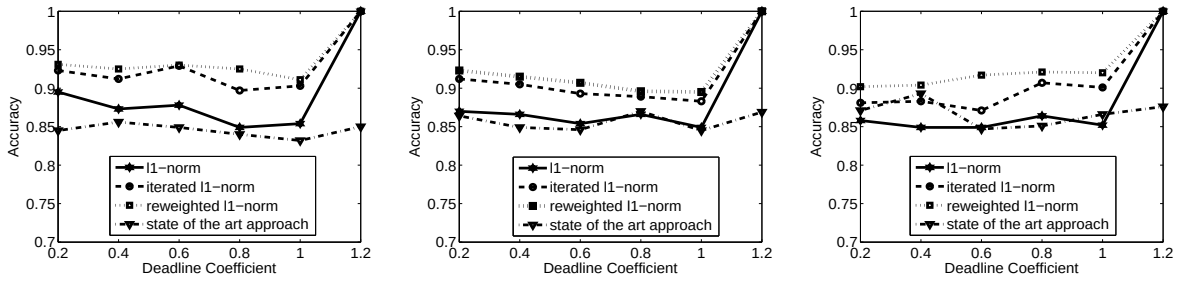
(i) Fit the best normal distribution for each arc based on the S training data;

(ii) Use state-of-the-art algorithm to find the optimal path;

(iii) Validate this optimal path using the testing data, and repeat 1,000 times to obtain the accuracy measure.

Based on above settings, the accuracy results are obtained, which are shown in Fig. 3.8. Comparing the three sub-figures, it is observed that, as expected, *reweighted ℓ_1-norm* algorithm

[2]The dataset is provided by the BMW Group.



3.8.a data size: 100 3.8.b data size: 500 3.8.c data size: 1000

Figure 3.8: Accuracy for the ℓ_1-Norm Based Algorithm and State-of-the-Art Algorithm

always achieves higher accuracy than that of *iterated* ℓ_1*-norm* algorithm, and *iterated* ℓ_1*-norm* obtains higher accuracy than that of ℓ_1*-norm* algorithm. The reason has been previous analyzed. Considering the accuracy for data size of 100, ℓ_1*-norm* always has better accuracy than that of the state-of-the-art algorithm. As data size increases to 500 and 1000, accuracy of the state-of-the-art algorithm for some coefficients exceeds that of ℓ_1*-norm* algorithm. This is because the state-of-the-art algorithm can work well only for normal distribution. When size of the data is small, such as 100, the fitting between normal distribution and real data is usually poor. As data size becomes large, the sampled data will become close to the distribution. However, its accuracy is significantly affected by probability distribution and data size. By contrast, the ℓ_1*-norm* based algorithms are not strictly impacted by them, which has been previously analyzed. However, regarding the accuracy for large deadline coefficients, i.e., 1.2 or larger, the proposed approach always achieves 100% accuracy, which clearly outperforms the state-of-the-art algorithm. Moreover, it is also important to note that, the accuracy measured for ℓ_1*-norm* algorithm in Fig. 3.8 is only the lower bound.

One important observation is that, the overall accuracy for three ℓ_1*-norm* based algorithms is not as high as those in Sections 3.3 and 3.4. This is because it made some assumptions there to analyze the factors that may impact the performance of the proposed approach (e.g., keeping the same distribution, standard deviation or O-D pairs). By contrast, the traffic data in this section is collected from real vehicle trajectories, and there are consequently no such assumptions. However, considering that the real traffic data is always random and most of accuracy is above 85%, the ℓ_1*-norm* based algorithms achieve good performance for the real world road

Table 3.10: Computation Time for the ℓ_1-Norm Based Algorithms and State-of-the-Art Algorithm

α	0.2	0.4	0.6	0.8	1.0	1.2
ℓ_1-*norm*	0.454	0.435	0.465	0.476	0.472	0.464
iterated ℓ_1-*norm*	1.541	1.730	1.712	1.860	1.687	1.702
reweighted ℓ_1-*norm*	1.602	1.873	1.914	1.937	1.837	1.821
state-of-the-art	0.808	0.721	0.554	0.443	0.453	0.450

network and traffic. Additionally, the deadline is an important parameter for the state-of-the-art algorithm. To show the impact on computation time caused by deadline coefficients, the averaged running time is recorded in Table 3.10. From Table 3.10, it is observed that ℓ_1-*norm* algorithm is more efficient than *iterated* ℓ_1-*norm* algorithm, and *iterated* ℓ_1-*norm* algorithm is more efficient than *reweighted* ℓ_1-*norm* algorithm in terms of computation time. This efficiency is obtained at price of accuracy. Comparing ℓ_1-*norm* and the state-of-the-art algorithm, it is observed that, when deadline coefficient is small, averaged running time for the state-of-the-art algorithm is comparatively longer. The underlying reason is that only if there exists the path whose expected travel time is not longer than deadline, the state-of-the-art algorithm can work efficiently. Otherwise, it will employ an enumeration method to determine the optimal path, which is inefficient. Since at each time, the travel time is randomly sampled from the real data, if deadline coefficient is small, the chance that the smallest expected travel time is larger than deadline increases. Consequently, higher running time for small coefficient is observed. As coefficient becomes larger, that chance will decrease. Consequently, the corresponding running time will also become short. However, it is not always becoming shorter for the state-of-the-art algorithm, because once there exists the path whose expected travel time is not longer than deadline, the computation time will not change, which can be observed from Table 3.10.

Combining both accuracy and computation time together, it is found that ℓ_1-*norm* algorithm and the state-of-the-art algorithm have their own advantages. When data size is comparatively small, i.e., 100 and 500, the accuracy of ℓ_1-*norm* algorithm is always higher (at least not lower) than that of the state-of-the-art algorithm. When data size is large, i.e., 1,000, and deadline coefficient is small, i.e., 0.2 and 0.4, accuracy for the state-of-the-art algorithm is higher. However, for all different data sizes, as long as deadline coefficient is large, i.e., not less than 1.2,

accuracy for the ℓ_1-*norm* algorithm is always 100%, which is much higher than that of the state-of-the-art algorithm. It is also highlighted that, the deadline coefficient is determined by driver, and it can be any positive values. It only considers the range of 0.2-1.2 for the sake of conveniently analyzing the factors that may affect accuracy. More importantly, as stated in Sections 3.3 and 3.4, accuracy obtained for ℓ_1-*norm* algorithm in Fig. 3.8 is only the lower bound. By contrast, computation time of the state-of-the-art algorithm is obviously affected by deadline coefficient. When deadline coefficient is small, its computation time is much higher than that of ℓ_1-*norm* algorithm. Furthermore, as deadline coefficient increases, its computation time approximately converges, which is slightly lower than that of ℓ_1-*norm* algorithm. Taking all these into account, it can be concluded that ℓ_1-*norm* algorithm is better than the state-of-the-art algorithm. Additionally, comparing ℓ_1-*norm* algorithm with *iterated* ℓ_1-*norm* and *reweighted* ℓ_1-*norm* algorithms, it is concluded that ℓ_1-*norm* algorithm is better considering both accuracy and computation time.

Note that the proposed approach is data-driven. It largely relies on travel time samples of each road link, which can be obtained by processing the GPS trajectories of vehicles. In a metropolitan city, daily traffic may involve millions of vehicles, almost all of which are equipped with GPS devices. Therefore, it is easy and feasible to deploy the proposed approach into realistic vehicle routing by exploring the big traffic data. Generally, as analyzed previously, if the proposed approach adopts huge amount of the traffic data, accuracy is expected to be higher, but running time is also accordingly longer. On the other hand, if the size of traffic data is small in the proposed approach, accuracy may decrease although running time is shorter. Therefore, it is needed to seek an optimal balance between them.

3.6 Summary

This chapter aims at solving the arriving on time problem for a single vehicle, and the direct objective is to determine an optimal path that maximizes the probability of reaching destination before deadline. Generally, this problem is difficult to solve unless it applies some strong assumptions on travel time distribution, correlation or deadline. To eliminate these assumptions,

it has transformed the problem into a cardinality minimization problem, and further used an ℓ_1-norm algorithm and its variants to solve the problem. The simulation results on an artificial road network have shown that the algorithm works well under a variety of probability distributions. The performance is not affected even when travel time dependencies are considered. Moreover, it can solve the problem with various deadlines. When tested on the real world road network with real traffic data, the results show that the ℓ_1-*norm* algorithm outperforms the state-of-the-art algorithm. Comparing the ℓ_1-*norm* algorithm with its two variants, it has been concluded that the ℓ_1-*norm* algorithm is a better choice by considering both accuracy and computation time.

Note that, with the ℓ_1-*norm* relaxation, searching an optimal path in this arriving on time problem finally becomes solving an MILP problem, where the decision variable size is the sum of arc amount in the graph and traffic data size for each arc. Moreover, the traffic data size is also the number of inequality constraints, and node amount is the number of equality constraints. Generally, the worst computation complexity of this MILP problem will exponentially increase as the graph or traffic data size scale up. Although a central routing engine server might be powerful, the computation may still become prohibitively time-consuming, especially, real road network scale and traffic data size are generally huge. Accordingly, the computation efficiency of this MILP-based arriving on time problem will be improved by exploring some favorable features: different from the general MILP problem, the equality constraint in the arriving on time problem satisfies the total unimodularity property [116, 58], and a more efficient solution will be seek by exploring its advantage in the next chapter.

Chapter 4

Improving the Computation Efficiency

To improve the computation efficiency of the MILP-based arriving on time problem in Chapter 3, a partial Lagrange multiplier method is proposed in this chapter[1]*. The proposed approach explores the total unimodularity of an incidence matrix in the equality constraint [116]. In particular, if it is the only constraint in this MILP problem apart from the lower bound and upper bound constraints, then the optimal solution to the corresponding linear programming (LP) problem will naturally yield integer values (i.e., the part corresponding to the path). At the same time, solving an LP problem only incurs polynomial computation [117], which is more efficient than an MILP solver, i.e., the B&B method. However, the existence of the inequality constraints in the MILP problem is likely to impede us from utilizing the advantage of this total unimodularity property. Thus, the partial Lagrange multiplier method is adopted to relax those inequality constraints [118]. More specifically, it reformulates the inequality constraints by shifting them into the objective function as additional terms. Then the subgradient method is employed to solve this partial Lagrange multiplier problem iteratively [119], and in each iteration the total unimodularity property is efficiently exploited.*

[1]This chapter is developed based on my publication: Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. Improving the Efficiency of Stochastic Vehicle Routing: A Partial Lagrange Multiplier Method. IEEE Transactions on Vehicular Technology (TVT), vol. 65, no. 6, pp. 3993-4005, 2016.

The proposed approach in this chapter has two advantages. Firstly, the partial Lagrange multiplier method can significantly improve the computation efficiency, in comparison with the traditional B&B method. Secondly, the proposed approach has attractive generalization, because it can be easily applied to many other problems that can be expressed as some specified convex integer programming forms (linear or nonlinear). To demonstrate this desirable generalization, it is also applied to solve the SSPD model (Stochastic Shortest Path Problem with Delay Excess Penalty, as described in Section 2.1.3 of Chapter 2) based SSP problem.

The remainder of this chapter is structured as follows. Section 4.1 briefly introduces the two representative models in the SSP problem, i.e., the PT model and the SSPD model, and the total unimodularity of the incidence matrix. Section 4.2 proposes a partial Lagrange multiplier method to efficiently solve the SSP problems based on the two models. Moreover, it elaborates the potential improvement and generalizes the proposed method to other applications. Section 4.3 provides the convergence analysis and scalability analysis of the proposed method. Section 4.4 reports the experimental results. Section 4.5 discusses several key issues regarding the proposed method, and concludes this chapter.

4.1 The Two Models and Total Unimodularity

In order to demonstrate the desirable generalization of the proposed method, the SSPD model as well as the PT model are considered. For better comparison of the two models from the mathematical perspective, the PT model is presented again in this section, which is the same with Eq. (3.16).

4.1.1 The Two Models

To solve the PT model based SSP problem, the cardinality minimization and ℓ_1-norm relaxation are explored to reformulate it as an MILP problem in Chapter 3, which is expressed as:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \xi_i \;\Bigg|\; \begin{array}{l} \vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T \leq \xi_i;\ \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0;\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}. \end{array} \tag{4.1}$$

In the SSPD model based SSP problem, each arc has a deterministic travel fee and a random travel time. It aims to obtain an optimal path that minimizes the sum of these two types of cost, i.e., the total travel fee and the expected penalty for arriving at the destination later than the predefined deadline [81]. This problem can be formulated as follows:

$$\min_{\vec{\mathbf{x}}} \vec{\mathbf{c}}^{\top}\vec{\mathbf{x}} + \frac{d}{S}\sum_{i=1}^{S}\xi_i \;\Bigg|\; \begin{array}{l} \vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T \leq \xi_i;\ \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0;\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}, \end{array} \tag{4.2}$$

where $\vec{\mathbf{c}}$ denotes the vector of the deterministic travel fees on arcs and d is the penalty for each unit delay with respect to the deadline T. Note that, with $\vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T \leq \xi_i$ and $\xi_i \geq 0$, the penalty is only effective when the delay happens. The other variables share the same meaning with those of the PT model.

4.1.2 Total Unimodularity of the Incidence Matrix

The two optimization models in Eq. (4.1) and Eq. (4.2) can be further formulated as a standard MILP problem, and then solved by existing solvers. The solvers typically employ the B&B method and suffer from exponential computation complexity. Because $\mathbf{M}$ is the incidence matrix of a directed graph, it satisfies the total unimodularity property [120, 121, 58]. Additionally, considering that there are only integer elements in $\vec{\mathbf{b}}$, the optimal solution to the corresponding LP problem in Eq. (4.3), where the SSPD model is taken as an example, would be integer if $\mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}$ is the only constraint with or without the lower bound and upper bound constraints [122]. Therefore, the solution to the MILP problem can be obtained by solving only the corresponding LP problem without using the B&B method, which will greatly reduce computation complexity.

$$\min_{\vec{\mathbf{x}}} \vec{\mathbf{c}}^{\top}\vec{\mathbf{x}} + \frac{d}{S}\sum_{i=1}^{S}\xi_i \;\Bigg|\; \begin{array}{l} \vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T \leq \xi_i; \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0;\ 0 \leq x_i \leq 1. \end{array} \tag{4.3}$$

Unfortunately, in the arriving on time problem, the inequality constraints always exist, e.g., $\vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T \leq \xi_i$ in the two models. The inequality constraint will undermine the advantages brought by the total unimodularity in the equality constraint and consequently the B&B method is still required.

4.2 Partial Lagrange Multiplier Method

The total unimodularity property in the two models is attractive as it ensures that an optimal integer solution can be found by solving the LP problem. Therefore, the partial Lagrange multiplier is introduced to the LP problem (e.g., Eq. (4.3)) of the two aforementioned optimization models. Here, the SSPD model is taken to explain the method. The similar procedure can be applied to the PT model.

4.2.1 Partial Lagrange Multiplier

The (full) *Lagrange multiplier* [123] is usually used to solve the optimization problem:

$$\min_{\vec{\mathbf{x}}} f_0(\vec{\mathbf{x}}) \;\Bigg|\; \begin{array}{l} f_i(\vec{\mathbf{x}}) \leq 0, \;\; i = 1, \ldots, m; \\ h_i(\vec{\mathbf{x}}) = 0, \;\; i = 1, \ldots, p, \end{array} \tag{4.4}$$

where $\vec{\mathbf{x}} \in \mathbb{R}^n$. The basic idea in the Lagrange multiplier is to relax the constraints in Eq. (4.4) by augmenting the objective function with a weighted sum of the constraint functions. Then the *Lagrange function* becomes:

$$L(\vec{\mathbf{x}}, \vec{\lambda}, \vec{\nu}) = f_0(\vec{\mathbf{x}}) + \sum_{i=1}^{m} \lambda_i f_i(\vec{\mathbf{x}}) + \sum_{i=1}^{p} \nu_i h_i(\vec{\mathbf{x}}), \tag{4.5}$$

where $\vec{\lambda}$ and $\vec{\nu}$ are the Lagrange multipliers. Its *dual function* $g(\vec{\lambda}, \vec{\nu})$ can be expressed as follows:

$$g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{\mathbf{x}} \in \mathbb{R}^n} L(\vec{\mathbf{x}}, \vec{\lambda}, \vec{\nu}). \tag{4.6}$$

Finally, with $\vec{\lambda} \geq 0$, the *dual problem* for Eq. (4.4) can be expressed as follows:

$$\max_{\vec{\lambda}, \vec{\nu}} \; g(\vec{\lambda}, \vec{\nu}) \;\Big|\; \vec{\lambda} \geq 0. \tag{4.7}$$

If the problem in Eq. (4.4) is convex, then the problem in Eq. (4.7) will be concave, and the two problems usually have the same optimal solution with respect to $\vec{\mathbf{x}}$. Moreover, this dual

problem can be efficiently solved by the subgradient method [119] which iteratively updates $\vec{\mathbf{x}}$, $\vec{\lambda}$ and $\vec{\nu}$ until they converge.

To take advantage of the total unimodularity in Eq. (4.3), which is also called the *primal problem*, it utilizes the *partial Lagrange multiplier*. This relaxes the inequality constraint by transforming and placing the constraint in the objective function with the corresponding Lagrange multiplier [118, 124]. Thus, the dual function $g(\vec{\lambda})$ of the SSPD model in Eq. (4.3) becomes:

$$\min_{\vec{\mathbf{x}}} \vec{\mathbf{c}}^{\top}\vec{\mathbf{x}} + \sum_{i=1}^{S} \frac{d}{S}\xi_i + \lambda_i(\vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T - \xi_i) \left| \begin{array}{l} \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0; 0 \leq x_i \leq 1, \end{array} \right. \tag{4.8}$$

where $\lambda_i \geq 0$.

4.2.2 Solution to the Partial Lagrange Multiplier

Eq. (4.8) is only the dual function. To find the real solution to the primal problem, its dual problem should be solved, which has a similar form to that in Eq. (4.7). It first transforms the dual function $g(\vec{\lambda})$ in Eq. (4.8) as follows:

$$\min_{\vec{\mathbf{x}}} (\vec{\mathbf{c}}^{\top} + \mathbf{W}^{\top}\vec{\lambda})\vec{\mathbf{x}} - \vec{\lambda}^{\top}\vec{\mathbf{T}} + (\frac{d}{S}\vec{\mathbf{1}} - \vec{\lambda})^{\top}\vec{\xi} \left| \begin{array}{l} \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0; 0 \leq x_i \leq 1, \end{array} \right. \tag{4.9}$$

where $\lambda_i \geq 0$. $\mathbf{W}$ is the traffic data matrix consisting of $\vec{\mathbf{w}}_i$. $\vec{\mathbf{T}}$ is the deadline vector, each element of which is T. $\vec{\mathbf{1}}$ is a vector of ones. The solution of the corresponding dual problem of Eq. (4.9) can be obtained using the subgradient method which iteratively solves two linear sub-problems, referred to as Sub-problem I and Sub-problem II, to obtain $\vec{\mathbf{x}}$ and $\vec{\xi}$, respectively. Then it updates the multiplier $\vec{\lambda}$. Although the subgradient method usually involves more than one iteration, as long as the number of iterations is bounded by a linear or polynomial function of the problem scale, i.e., graph size and traffic data size on each arc, the complexity for the method is still polynomial. This is because the two linear sub-problems need at most polynomial computation.

4.2.2.1 Sub-problem I

Sub-problem I is an LP problem with $\vec{\mathbf{x}}$, which should be solved in each iteration with updated $\vec{\lambda}$, i.e.,

$$\min_{\vec{\mathbf{x}}}(\vec{\mathbf{c}}^{\top} + \mathbf{W}^{\top}\vec{\lambda})\vec{\mathbf{x}} - \vec{\lambda}^{\top}\vec{\mathbf{T}} \left| \begin{array}{l} \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ 0 \leq x_i \leq 1, \end{array} \right. \tag{4.10}$$

where $\lambda_i \geq 0$. Since Eq. (4.10) is an LP problem, the incidence matrix $\mathbf{M}$ satisfies the total unimodularity property and the O-D vector $\vec{\mathbf{b}}$ is an integer, its optimal solution is always integer. Moreover, this LP problem can be efficiently solved by the interior point method (IPM), which guarantees polynomial computation complexity [125].

4.2.2.2 Sub-problem II

Sub-problem II is an LP problem with $\vec{\xi}$, which should also be solved in each iteration with updated $\vec{\lambda}$, i.e.,

$$\min_{\vec{\xi}}(\frac{d}{S}\vec{\mathbf{I}} - \vec{\lambda})^{\top}\vec{\xi} \;\Big|\; \xi_i \geq 0, \tag{4.11}$$

where $\lambda_i \geq 0$. Since this objective function is linear and the constraint is to guarantee the lower bound, it can be solved with linear computation complexity. Specifically, in real implementation, if $(\frac{d}{S} - \lambda_i) > 0$, $\xi_i = 0$. If $(\frac{d}{S} - \lambda_i) < 0$, $\xi_i = +\infty$, and a big value of $\mathbf{\Lambda}$ (i.e., 10^7) is used to replace $+\infty$. If $(\frac{d}{S} - \lambda_i) = 0$ and according to the KKT condition [123], it has $\lambda_i(\vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}} - T - \xi_i) = 0$. Since $\lambda_i \neq 0$ due to $(\frac{d}{S} - \lambda_i) = 0$, then it has $\xi_i = \vec{\mathbf{w}}_i^{\top}\vec{\mathbf{x}}^* - T$, where $\vec{\mathbf{x}}^*$ is the optimal solution to Eq. (4.10) at the same iteration.

4.2.2.3 Updating $\vec{\lambda}$

The Sub-problems I and II are connected by the mutual partial Lagrange multiplier $\vec{\lambda}$. In each iteration, $\vec{\lambda}$ will be updated as follows:

$$\vec{\lambda}^{(k+1)} = \vec{\lambda}^{(k)} + \alpha_k \cdot \nabla g(\vec{\lambda}^{(k)}), \tag{4.12}$$

where α_k and $\nabla g(\vec{\lambda}^{(k)})$ are step size and subgradient of $\vec{\lambda}$ at the kth iteration, respectively.

More specifically, for α_k, the diminishing step rule is applied into an optimal version, and it has $\alpha_k = (R/G)/\sqrt{k}$, where R and G are relevant to the bound of $\vec{\lambda}$ and $\nabla g(\vec{\lambda})$ respectively (more details will be presented in Section 4.3.1 and Section 4.3.2). Regarding the subgradient, it has $\nabla g(\vec{\lambda}^{(k)}) = (\mathbf{W}\vec{\mathbf{x}}^{(k)} - \vec{\mathbf{T}} - \vec{\xi}^{(k)})$, which can be easily derived from Eq. (4.8), where $\vec{\mathbf{x}}^{(k)}$ and $\vec{\xi}^{(k)}$ are the optimal solutions to Eq. (4.10) and Eq. (4.11), respectively. Further, to keep that each element of $\vec{\lambda}^{(k+1)}$ is non-negative, it has

$$\lambda_i^{k+1} = \max\{\lambda_i^{k+1}, 0\}. \tag{4.13}$$

4.2.2.4 Stopping Criterion

The subgradient method is different from the descent method [126], and there is usually no dominating stopping criterion. In real applications, the proposed method always keeps track of the best solution, which is expressed as $g_{best}^{(k)} = \max\limits_{i=1,\ldots,k} g(\vec{\lambda}^{(i)})$, where $g(\vec{\lambda})$ is the dual function in Eq. (4.9). Here, the iteration is terminated if $g_{best}^{(k)}$ has no significant improvement within Φ iterations [127, 128] (i.e., $g_{best}^{(k+1)} - g_{best}^{(k)} \geq \epsilon_1$, where ϵ_1 is a small positive value). The overall procedure of the partial Lagrange multiplier method is summarized in Algorithm 1.

4.2.3 Improvement

Sub-problem I in Eq. (4.10) is an LP problem with respect to $\vec{\mathbf{x}}$, and it can be efficiently solved with polynomial computation complexity by the IPM. Due to the total unimodularity, $\vec{\mathbf{x}}$ will be a 0-1 vector by default, which represents a path connecting the origin and the destination. Therefore, considering the objective function in Eq. (4.10), Sub-problem I is also a shortest path problem in nature, the objective of which is to minimize certain cost, i.e., $\vec{\mathbf{c}}^{\top} + \mathbf{W}^{\top}\vec{\lambda}$. Since in Sub-problem I, $\vec{\lambda}$ is a constant and the output which will be utilized in the subsequent steps is only $\vec{\mathbf{x}}$, the traditional shortest path algorithm can also employed, i.e., the Dijkstra algorithm, to solve it. To find this optimal path $\vec{\mathbf{x}}$ with respect to Eq. (4.10), the Dijkstra algorithm only needs logarithmic computation complexity [129]. Hence, as the problem scales up, the Dijkstra algorithm is much more efficient than the IPM. Note that, generally, an LP

Algorithm 1: Partial Lagrange Multiplier Method

Input : $\mathbf{W}$, $\mathbf{M}$, $\vec{\mathbf{c}}$, $\vec{\mathbf{b}}$, d and Φ;
Output: $\vec{\mathbf{x}}^*$;
Initialize $g_{best}^{(0)} = -\infty$;
Set the iteration index $k = 1$, $k_x = 0$;
Randomly initialize λ_i and ensures that $0 \leq \lambda_i^{(k)} \leq \frac{d}{S}$;
do
  Get optimal solution $\vec{\mathbf{x}}^{(k)}$ [Sub-problem I in Eq.(4.10)];
  Get optimal solution $\vec{\xi}^{(k)}$ [Sub-problem II in Eq.(4.11)];
  if $g_{best}^{(k)} - g_{best}^{(k-1)} \geq \epsilon_1$ **then**
    $k_x = k$;
  $k = k + 1$;
  Compute the step size α_k;
  Compute the gradient $\nabla g(\vec{\lambda}^{(k)})$;
  Update multiplier $\vec{\lambda}^{(k)}$ using Eq. (4.12) and Eq. (4.13);
while $k - k_x > \Phi$;
Output optimal path $\vec{\mathbf{x}}^* = \vec{\mathbf{x}}^{(k_x)}$.

problem cannot be solved by the shortest path algorithm. But in Eq. (4.10), $\mathbf{M}$ is the node-arc incidence matrix of a graph, and $\vec{\mathbf{b}}$ is a O-D vector, so the Dijkstra algorithm is able to solve it.

However, it is natural to write Sub-problem I as an LP problem because the SSPD model is an MILP problem, and this LP is part of it, which is helpful to analyze the partial Lagrange multiplier method. But in real applications, it may employ shortest path algorithm, i.e., Dijkstra algorithm, to solve Eq. (4.10) to further improve efficiency.

4.2.4 Generalization

The proposed partial Lagrange multiplier method has attractive generalization, because it can be easily extended to other problems besides the SSP problem. More specifically, based on the

descriptions in Section 4.2.2, these problems should have the following form:

$$\min_{\vec{\mathbf{x}}} \rho(\vec{\mathbf{x}}, \vec{\xi}) \left| \begin{array}{l} \mathbf{W}\vec{\mathbf{x}} - \vec{\mathbf{T}} \leq \psi(\vec{\xi});\ \mathbf{M}\vec{\mathbf{x}} = \vec{\mathbf{b}}; \\ \xi_i \geq 0;\ \vec{\mathbf{x}} \in \{0,1\}^{|A_r|}, \end{array} \right. \tag{4.14}$$

where $\rho(\vec{\mathbf{x}}, \vec{\xi})$ should be a linear function with respect to $\vec{\mathbf{x}}$, and convex function (linear or nonlinear) with respect to $\vec{\xi}$. $\psi(\vec{\xi})$ should be a concave function (linear or nonlinear) with respect to $\vec{\xi}$. $\mathbf{M}$ should satisfy the total unimodularity and $\vec{\mathbf{b}}$ should be an integer vector. Generally, most of the SSP problems satisfy these conditions [120] and therefore the partial Lagrange multiplier method is applicable. As for other problems of this form, the B&B method is subject to the exponential computation complexity. For the proposed method, regarding Eq. (4.14), Sub-problem I is an LP problem with respect to $\vec{\mathbf{x}}$ which always generates an integer solution. Sub-problem II is a convex optimization problem (linear or nonlinear) with respect to $\vec{\xi}$. Both of them can be solved with at most polynomial computation complexity. Here, Sub-problem I can be solved with logarithmic computation complexity as stated in Section 4.2.3. Therefore, as long as the number of iterations is bounded by a linear or polynomial function of the problem, its computation complexity would be determined as polynomial at most.

4.3 Theoretical Performance Analysis

In this section, the convergence of the partial Lagrange multiplier method for the PT model and the SSPD model is analyzed. Then the scalability analysis is performed.

4.3.1 Convergence Analysis

To demonstrate theoretically that the proposed method can efficiently solve the two optimization models, it first proves that the dual function $g(\vec{\lambda})$ is concave. Then, it proves that $g(\vec{\lambda})$ can converge within a limited number of iterations. Finally, it proves the strong duality that when $g(\vec{\lambda})$ converges, the corresponding $\vec{\mathbf{x}}^*$ is the optimal solution to the MILP problem of the two optmization models. Again, for the sake of better illustration, the PT model is taken as an example. The same procedure can be applied for the SSPD model.

4.3.1.1 Concavity

The dual function of the PT model is formulated as follows:

$$g(\vec{\lambda}) = \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \xi_i + \lambda_i(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i), \tag{4.15}$$

where the domain of $\vec{\mathbf{x}}$ is defined by the constraint in Eq. (4.8). To show the concavity, the parameter θ ($0 \leq \theta \leq 1$) is introduced, and it has

$$\theta \cdot g(\vec{\lambda}^1) = \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \theta\xi_i + \theta\lambda_i^1(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i), \tag{4.16}$$

$$(1-\theta) \cdot g(\vec{\lambda}^2) = \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} (1-\theta)\xi_i + (1-\theta)\lambda_i^2(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i), \tag{4.17}$$

and

$$\begin{aligned}
&g(\theta\vec{\lambda}^1 + (1-\theta)\vec{\lambda}^2) \\
&= \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \xi_i + (\theta\lambda_i^1 + (1-\theta)\lambda_i^2)(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i) \\
&= \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} (\theta+(1-\theta))\xi_i + (\theta\lambda_i^1 + (1-\theta)\lambda_i^2)(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i) \\
&= \min_{\vec{\mathbf{x}}} \sum_{i=1}^{S} \theta(\xi_i + \lambda_i^1(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i)) + \\
&\quad \sum_{i=1}^{S} (1-\theta)(\xi_i + \lambda_i^2(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}} - T - \xi_i)).
\end{aligned} \tag{4.18}$$

Because $\min_{p_i \in \vec{\mathbf{p}}}\{p_i\} + \min_{q_i \in \vec{\mathbf{q}}}\{q_i\}$ is always no larger than $\min_{e_i \in \vec{\mathbf{e}} = \vec{\mathbf{p}} + \vec{\mathbf{q}}}\{e_i\}$ [123], considering Eq. (4.16), Eq. (4.17) and Eq. (4.18) together, it can derive that $\theta \cdot g(\vec{\lambda}^1) + (1-\theta) \cdot g(\vec{\lambda}^2) \leq g(\theta\vec{\lambda}^1 + (1-\theta)\vec{\lambda}^2)$. Therefore, $g(\vec{\lambda})$ is concave.

4.3.1.2 Convergence

It has been proved that $g(\vec{\lambda})$ is concave. Then, that it can converge to the optimum by updating $\vec{\lambda}$ iteratively will be shown. Because $\nabla g(\vec{\lambda}^{(k)}) = (\mathbf{W}\vec{\mathbf{x}}^{(k)} - \vec{\mathbf{T}} - \vec{\xi}^{(k)})$, where $\mathbf{W}$ is the real world travel time data, $\vec{\mathbf{T}}$ is the user defined deadline and $\vec{\xi}$ is a non-negative decision variable

of the minimization problem in Eq. (4.11), $\|\nabla g(\vec{\lambda})\|_2 \leq G$ can be assumed where its details can be found in Section 4.3.2. From Eq. (4.11), it derives that $\lambda_i \leq \frac{d}{S}$. Additionally, considering that $\lambda_i \geq 0$ is an intrinsic constraint, therefore $\|\vec{\lambda}^{(1)} - \vec{\lambda}^*\|_2$ is bounded, and it can assume $\|\vec{\lambda}^{(1)} - \vec{\lambda}^*\|_2 \leq R$ (details can be found in Section 4.3.2). Then it has:

$$\begin{aligned}
&\|\vec{\lambda}^{(k+1)} - \vec{\lambda}^*\|_2^2 = \|\vec{\lambda}^{(k)} + \alpha_k \nabla g(\vec{\lambda}^{(k)}) - \vec{\lambda}^*\|_2^2 \\
&= \|\vec{\lambda}^{(k)} - \vec{\lambda}^*\|_2^2 + 2\alpha_k \nabla g(\vec{\lambda}^{(k)})(\vec{\lambda}^{(k)} - \vec{\lambda}^*) + \\
&\alpha_k^2 \|\nabla g(\vec{\lambda}^{(k)})\|_2^2 \\
&\leq \|\vec{\lambda}^{(k)} - \vec{\lambda}^*\|_2^2 + 2\alpha_k (g(\vec{\lambda}^{(k)}) - g(\vec{\lambda}^*)) + \\
&\alpha_k^2 \|\nabla g(\vec{\lambda}^{(k)})\|_2^2.
\end{aligned} \tag{4.19}$$

The inequality in Eq. (4.19) is derived based on the fact that for a concave function it has $g(\vec{\lambda}^*)) \leq g(\vec{\lambda}^{(k)})) + \nabla g(\vec{\lambda}^{(k)})(\vec{\lambda}^* - \vec{\lambda}^{(k)})$ [130]. Then, applying it recursively, the following result can be obtained:

$$\begin{aligned}
&\|\vec{\lambda}^{(k+1)} - \vec{\lambda}^*\|_2^2 \leq \|\vec{\lambda}^{(1)} - \vec{\lambda}^*\|_2^2 + \\
&2\sum_{i=1}^{k} \alpha_i (g(\vec{\lambda}^{(i)}) - g(\vec{\lambda}^*)) + \sum_{i=1}^{k} \alpha_i^2 \|\nabla g(\vec{\lambda}^{(i)})\|_2^2 \\
&\leq R^2 + 2\sum_{i=1}^{k} \alpha_i (g(\vec{\lambda}^{(i)}) - g(\vec{\lambda}^*)) + G\sum_{i=1}^{k} \alpha_i^2.
\end{aligned} \tag{4.20}$$

Because $g(\vec{\lambda})$ is concave and it always keeps the current best function value, then the following relationship can be derived:

$$\sum_{i=1}^{k} \alpha_i (g(\vec{\lambda}^{(i)}) - g(\vec{\lambda}^*)) \leq (g_{best}^{(k)} - g(\vec{\lambda}^*))(\sum_{i=1}^{k} \alpha_i). \tag{4.21}$$

Further, the following result can be obtained:

$$g_{best}^{(k)} - g(\vec{\lambda}^*) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} \alpha_i^2}{2\sum_{i=1}^{k} \alpha_i}. \tag{4.22}$$

For the optimal version of the diminishing rule $\alpha_k = (R/G)/\sqrt{k}$, as k increases, $g_{best}^{(k)}$ will converge to $g(\vec{\lambda}^*)$. Furthermore, the number of iterations required is $K \leq (RG/\epsilon_2)^2$, where ϵ_2 is the error tolerance.

4.3.1.3 Strong Duality

It will be shown that when $g(\vec{\lambda})$ is maximized, the corresponding $\vec{\mathbf{x}}^*$ is the optimal solution to the respective primal problem. According to the Slater's constraint qualification, if the original optimization problem is convex and it is strictly feasible that there exists at least one feasible solution guaranteeing that " $<$ " holds for the inequality constraint, then the optimal solution to the dual problem is also optimal to the primal problem [131]. The LP version of the SSPD model is convex, and it is easy to find at least one strictly feasible solution as long as the destination is reachable from the origin and that each λ_i is large enough. Consequently, it is concluded that if $\vec{\lambda}^*$ is the optimal solution to the dual problem, then the corresponding $\vec{\mathbf{x}}^*$ is also the optimal integer solution to its primal problem and hence the SSPD model.

As for the PT model, its convergence analysis can be easily deducted in a similar way with that of the SSPD model.

4.3.2 Scalability Analysis

Section 4.3.1 has proved that the proposed method will converge within $K \leq (RG/\epsilon_2)^2$ iterations. Regarding R, because it has $0 \leq \lambda_i \leq 1$ for the PT model and $0 \leq \lambda_i \leq \frac{d}{S}$ for the SSPD model, considering $\|\vec{\lambda}^{(1)} - \vec{\lambda}^*\|_2 \leq R$, it is valid to assume $R = \sqrt{S}$ for the PT model, where S is the length of $\vec{\lambda}$ and also the number of the traffic data samples on each arc. For the SSPD model, it is safe to assume $R = \sqrt{d}$. Regarding G, it has $\|\nabla g(\vec{\lambda})\|_2 \leq G$ and $\nabla g(\vec{\lambda}^{(k)}) = (\mathbf{W}\vec{\mathbf{x}}^{(k)} - \vec{\mathbf{T}} - \vec{\xi}^{(k)})$ for both models. On one hand, if $(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}}^{(k)} - T - \xi_i^{(k)}) \geq 0$, considering $\|\nabla g(\vec{\lambda})\|_2 \leq \|\mathbf{W}\vec{\mathbf{x}}\|_2 \leq \|\mathbf{W}\vec{\mathbf{I}}_{(|Ar|\times 1)}\|_2$, it is reasonable to assume $G = Q_1\sqrt{|Ar|}$, where Q_1 is a constant associated with the traffic data and $|Ar|$ is the length of $\vec{\mathbf{x}}$ and also the total number of arcs in the graph. On the other hand, if $(\vec{\mathbf{w}}_i^\top \vec{\mathbf{x}}^{(k)} - T - \xi_i^{(k)}) < 0$, considering $\xi_i = 0$ or $\xi_i = \vec{\mathbf{w}}_i^\top \vec{\mathbf{x}}^{(k)}$ in Sub-problem II, it has $\|\nabla g(\vec{\lambda})\|_2 \leq \|\vec{\mathbf{T}}\|_2$. Since T is the user-defined deadline, which usually increases with the number of arcs connecting the origin and destination, it is reasonable to assume that $G = Q_2\sqrt{|Ar|}$, where Q_2 is also a constant associated with the traffic data. Combing the two cases, it is safe to assume $G = Q\sqrt{|Ar|}$, where $Q = \max\{Q_1, Q_2\}$.

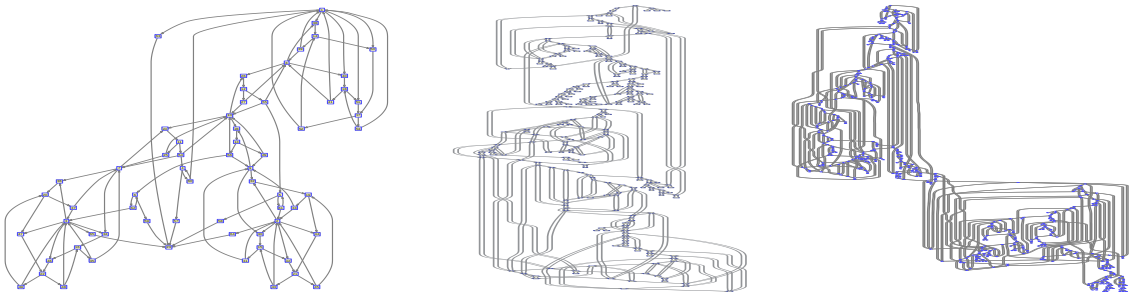
Incorporating them into $K \leq (RG/\epsilon_2)^2$, it can be concluded that for the PT model the required number of iterations is $K \leq Q^2 \times S \times |Ar|/(\epsilon_2)^2$. It is basically bounded by a linear function of the traffic data size on each arc and the graph size. For the SSPD model, the required number of iterations is $K \leq Q^2 \times d \times |Ar|/(\epsilon_2)^2$, where d is the penalty for the unit delay, hence a constant. Thus, the required number of iterations is bounded by the linear function of the graph size. Within each iteration, the major computation complexity comes from the LP problem in Sub-problem I, which can guarantee polynomial computation complexity using the IPM. Consequently, considering both optimization models, the overall computation complexity for the proposed partial Lagrange multiplier method can be measured as polynomial.

Note that the above conclusion is drawn based on the assumption that Sub-problem I is solved by the IPM. In real applications, the Dijkstra algorithm may be employed instead, which can guarantee logarithmic computation complexity. Therefore, the above conclusion of polynomial computation complexity is only an upper bound.

4.4 Experimental Analysis

It has been theoretically concluded in Section 4.3.1 and Section 4.3.2 that the proposed partial Lagrange multiplier method can solve the two optimization models of SSP problems efficiently. Here, it provides comparison with traditional B&B method (i.e., adopted in Chapter 3) to further justify the advantages of the proposed method. Moreover, to comprehensively evaluate the improvement brought by the Dijkstra algorithm in the proposed partial Lagrange multiplier method, the Sub-problem I are solved using both the IPM and the Dijkstra algorithm. In particular, for each experimental setting, it will utilize three methods to solve the SSPD model and the PT model, i.e., B&B method, PLM_IPM method (partial Lagrange multiplier method with IPM) and PLM_Dijkstra method (partial Lagrange multiplier method with the Dijkstra algorithm).

For each method, it solves the optimization models on three different scales of graphs: small scale, medium scale and large scale. For each scale, different traffic data sizes are applied. Additionally, for the large scale graph, the proposed method is implemented into a simulator, which is used in a commercial navigation system.



4.1.a 65-node,123-arc 4.1.b 256-node,668-arc 4.1.c 538-node,1450-arc

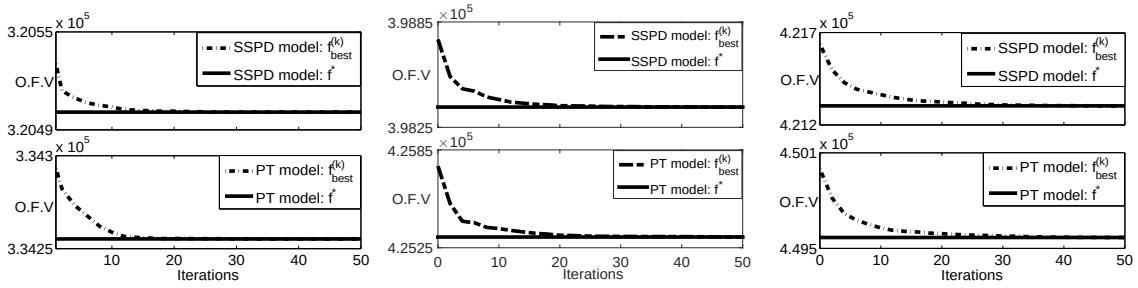
Figure 4.1: Three Graphs of Small Scale

4.4.1 Small Scale: Three Different Graphs

The experimental setting for both optimization models is stated as follows:

- It implements the methods on MATLAB (version: 2014a), which is installed in a PC with Intel Core i7-3540M processor and 8.00 GB RAM.
- It generates three graphs with different size: 1) 65-node and 123-arc, which is an artificial graph; 2) 256-node and 668-arc, which is extracted from the road network of Pennsylvania; and 3) 538-node and 1450-arc, which is extracted from the road network of California. The three graphs are shown in Fig. 4.1. Note that their topologies are described by the node-arc incidence matrix, which is M respectively in Eq. (4.1) and Eq. (4.2).
- It uses different traffic data size: 500, 1,000, 1,500 and 2,000, and all the traffic data (travel time) for each arc is randomly generated according to the normal distribution[2]. The unit of the travel time is second. Note that the data size is S, and the travel time data is $\vec{\mathbf{w}}_i$ in Eq. (4.1) and Eq. (4.2). For the SSPD model, is randomly generates the travel fee on each arc, which is $\vec{\mathbf{c}}$ in Eq. (4.2).
- It randomly chooses the origin-destination (O-D) pair, which is described by $\vec{\mathbf{b}}$ in Eq. (4.1) and Eq. (4.2), as long as D is reachable from O.

[2]In real-world scenarios, travel time may follow arbitrary distributions [132, 133]. Here, it uses normal distribution for simplification.



4.2.a 65-node,123-arc 4.2.b 256-node,668-arc 4.2.c 538-node,1450-arc

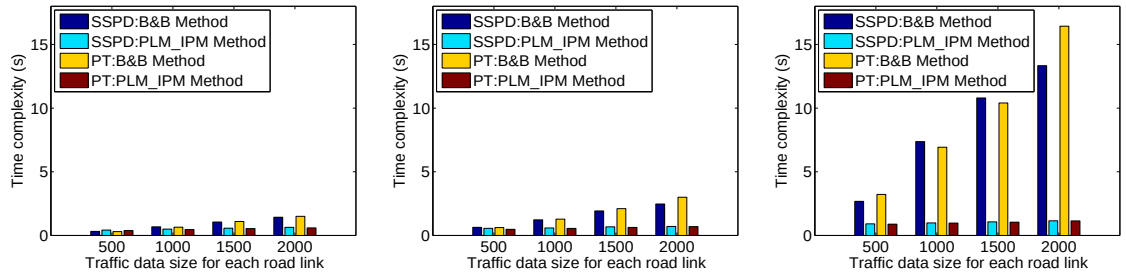
Figure 4.2: Average $f_{best}^{(k)}$ Over the Iterations

- It randomly chooses the deadline T. In most of the cases, it imposes the condition that, for the optimal path, the probability of arriving at the destination before the deadline is larger than 0% and smaller than 100%.
- For each setting, it runs experiments for 1000 times.

More specifically, the B&B method uses the *intlinprog* function of the MATLAB optimization toolbox to directly solve the MILP problem. The PLM_IPM method uses the *linprog* function of the same toolbox to solve the LP problem in Sub-problem I. The PLM_Dijkstra method uses the *graphshortestpath* function in the MATLAB bioinformatics toolbox. The convergence of the partial Lagrange multiplier method is shown in Fig. 4.2, and the average computation time of the three methods is shown in Fig. 4.3 and Fig. 4.4.

In Figs. 4.2(a), (b), and (c), 'O.F.V' on vertical axis refers to the objective function value. Regarding the proposed partial Lagrange multiplier method, the values of the best objective function $f_{best}^{(k)}$ are plotted for each graph as the number of iterations increases, and these values are averaged over different data sizes. Additionally, it plots the average of the actual optimal values f^*. From these curves, it is observed that all the objective functions on different graphs can converge to the real optimal solutions. Although the number of iterations required slightly increases as graph size scales up, the number of iterations for convergence is still no larger than 25 on the largest graph.

Comparing Figs. 4.3(a), (b), and (c), with respect to the PT model and the SSPD model, it can be easily observed that computation time of the PLM_IPM method is always less than that of

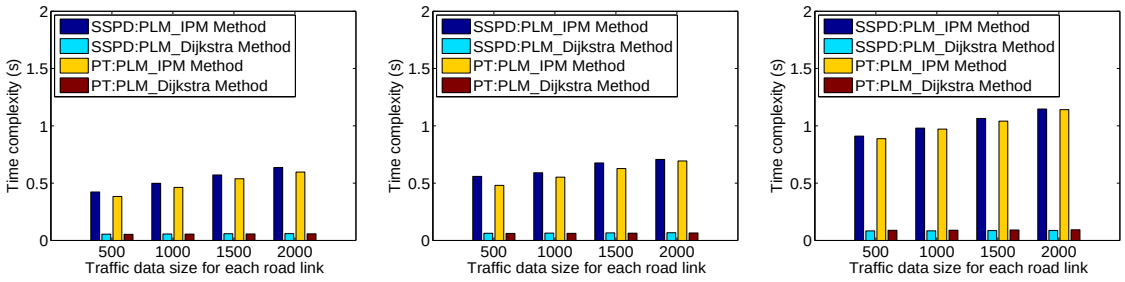


4.3.a 65-node,123-arc 4.3.b 256-node,668-arc 4.3.c 538-node,1450-arc

Figure 4.3: Computation Time with Respect to the B&B Method and the PLM_IPM Method

B&B method for most cases except for traffic data size of 500 in Fig. 4.3(a). As graph becomes larger, computation time of B&B method increases significantly, especially in Figs. 4.3(b) and (c). The computation time only slightly increases for the PLM_IPM method. Again, almost no increase in computation time can be noticeably found for the PLM_IPM method.

Before analyzing the causes for the observed effects of the computation time, note that the variable $\vec{x}$ represents the arcs of the graph. The length of $\vec{\xi}$ is the traffic data size of each arc, which is also the number of the inequality constraints in the PT model and the SSPD model. To solve the standard MILP problem of the optimization models, the B&B method will solve the corresponding LP problem of the same size in each branch, where $\vec{x}$ and $\vec{\xi}$ together form the decision variables. Consequently, this LP problem becomes bigger as the graph size or the traffic data size becomes larger. Moreover, the number of the inequality constraints also becomes larger if the traffic data size is increased. Although this LP problem can be solved with polynomial computation using the IPM, its scale is relatively large and the number of branches might be exponential. In the PLM_IPM method, in each iteration, the major computation is from the LP problem of Sub-problem I, whose decision variable is only $\vec{x}$. Additionally, considering that there is no inequality constraint, this LP problem is only subject to the scale of the graph. Thus, it is simpler than the B&B method. Although the decision variable $\vec{\xi}$ in Sub-problem II is associated with the traffic data size, it can be efficiently solved with linear computation because there is only a lower bound constraint. Its computation time can be nearly neglected. More importantly, the number of iterations is bounded by a linear function



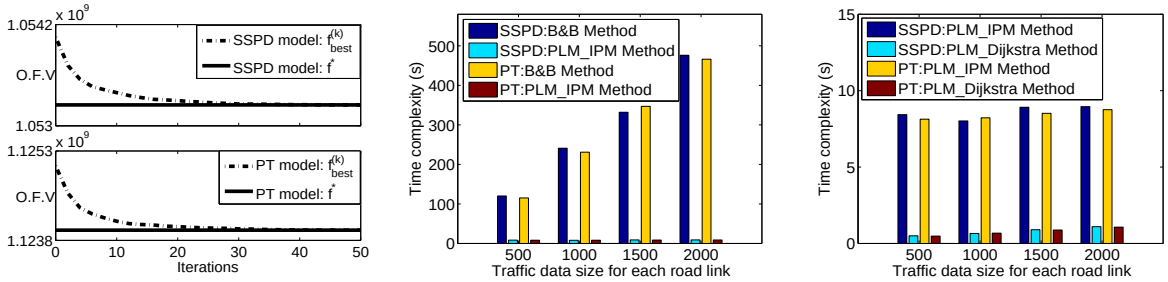
4.4.a 65-node,123-arc 4.4.b 256-node,668-arc 4.4.c 538-node,1450-arc

Figure 4.4: Computation Time with Respect to the PLM_IPM Method and the PLM_Dijkstra Method

of the graph scale or the traffic data size. By contrast, because of the total unimodularity, Sub-problem I always results in integer solutions, which also increases the convergence speed. Thus this number of iterations is more desirable than the number of branches in the B&B method. Taking all the analysis into consideration, it can postulate that the computation time slightly increases as the graph scales up while almost does not change as the traffic data size becomes larger within each graph. However, this advantage is not evident if graph size and traffic data size are both small, e.g., traffic data size of 500 on the 65-node and 123-arc graph. This is reasonable because the B&B method can also solve the MILP problem efficiently in this case.

Comparing Figs. 4.4(a), (b), and (c), with respect to the PT model and the SSPD model, it can be easily observed that computation time of PLM_Dijkstra method is always less than that of PLM_IPM method for each case, and computation time of the former is always around 0.1s. As the graph scales up, the computation time of the two methods only increases slightly.

Before analyzing the underlying causes of this result, note that the computation time of the three methods were not shown in the same figure because it is obvious that, the PLM_Dijkstra method is far more efficient than the B&B method, and the bar almost cannot be seen if it is plotted against that of the B&B method in the same figure. In the proposed partial Lagrange multiplier method, there are three major steps: solving Sub-problems I, solving Sub-problems II and updating Lagrange multiplier. As stated previously, the first step is more



4.5.a Convergence

4.5.b B&B method and PLM_IPM method

4.5.c PLM_IPM method and PLM_Dijkstra method

Figure 4.5: Comprehensive Results on the Graph of Medium Scale

time-consuming although the IPM method is efficient. On the other hand, from the functionality aspect, PLM_IPM method and PLM_Dijkstra method are the same. Nonetheless, the Dijkstra algorithm is more efficient to solve the Sub-problems I because it can guarantee logarithmic computation complexity. Therefore, it can be observed that computation time of the PLM_Dijkstra method is much less than that of the PLM_IPM method. Furthermore, it is far more efficient than the B&B method.

4.4.2 Medium Scale: Graph of Singapore Main Roads

The proposed method is further tested on a relatively large graph: Singapore main road network. To create this graph, it first extracts all nodes and arcs of the whole Singapore road network from OpenStreetMap [134]. Then it removes some small roads, and keep only the major ones. Thus, it has the Singapore main road graph with 6,476 nodes and 10,253 arcs. The experiment setting is exactly the same as the one in Section 4.4.1. The convergence curve of the partial Lagrange multiplier method is shown in Fig. 4.5(a), and the average computation time of the three methods are shown in Figs. 4.5(b) and (c).

From Fig. 4.5(a), it can be observed that although the graph scale is large, the proposed method can still guarantee convergence for the SSPD and PT models. Furthermore, the maximum number of iterations required for convergence is less than 30.

From Fig. 4.5(b), it can be observed that the computation time of the B&B method increases significantly as the data size increases. It is longer than 450s for the data size of 2,000, and longer than 100s even for the data size of 500, which is prohibitively time-consuming. While the computation time of the PLM_IPM method almost does not increase, which is always around 9s. Therefore, the PLM_IPM method is more efficient than the B&B method. From Fig. 4.5(c), it can be seen that, the computation time of the PLM_Dijkstra method only slightly increases as the data size scales up, which is always less than 1s, and also less than that of the PLM_IPM method. Therefore, it is far more efficient than the B&B method. Moreover, the outcomes of the analysis in Section 4.4.1 still holds for these observations.

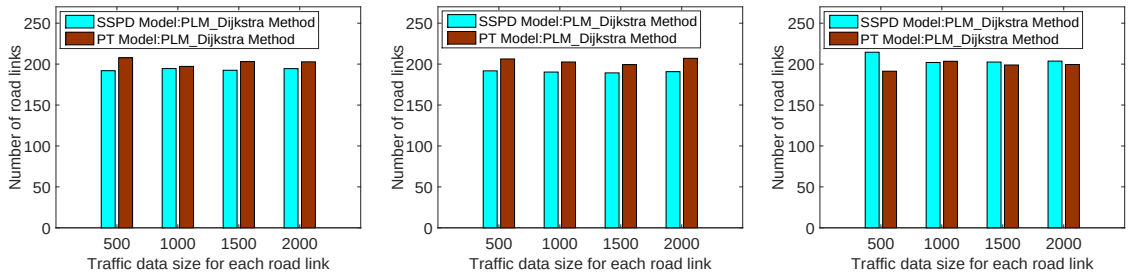
4.4.3 Large Scale: Graph of All Roads in Beijing

The experiments in Sections 4.4.1 and 4.4.2 showed promising results regarding the proposed partial Lagrange multiplier method, and it is continued to be tested on large scale road network, which is extracted from the map of Beijing city, as shown in Fig. 4.6. In [132, 135, 136], a popular set of real trajectory data of 10,357 taxis in Beijing is provided, which mainly records the taxi coordinates in terms of longitude and latitude at one moment[3]. However, it is not easy to incorporate them into the routing algorithms where travel time on each road link is directly needed. To make it more applicable, another set of GPS trajectory data generated by 32,670 taxicabs in Beijing is provided in [71], which includes the travel time on each road link structured in time slots and the underlying graph. To be more specific, there are 48 slots for one day and each slot spans 30 minutes. The graph contains 148,068 nodes and 329,148 arcs. Note that it treats bidirection road link as two different arcs in the graph.

The PLM_Dijkstra method is the most efficient in Sections 4.4.1 and 4.4.2, which is also the focus of this Chapter. It will further justify its feasibility and efficiency by solving real world routing on the Beijing road network with the real traffic data. Moreover, experiments will be performed on three different time dependent scenarios, i.e, 1am-4am, 7am-10am and 5pm-8pm. To achieve this, it only needs to query the traffic data which are generated within the

[3]The dataset is available at: https://www.microsoft.com/en-us/research/people/yuzheng/

Figure 4.6: Beijing Road Network

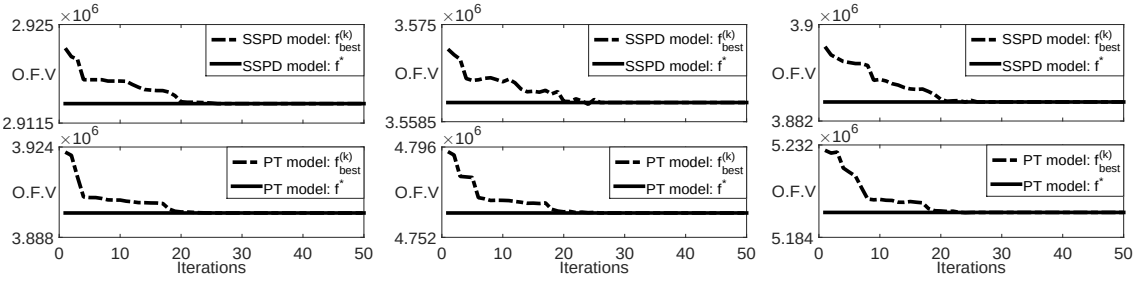


4.7.a 1am-4am 4.7.b 7am-10am 4.7.c 5pm-8pm

Figure 4.7: The Average Minimum Number of Road Links between O-D for Each Scenario

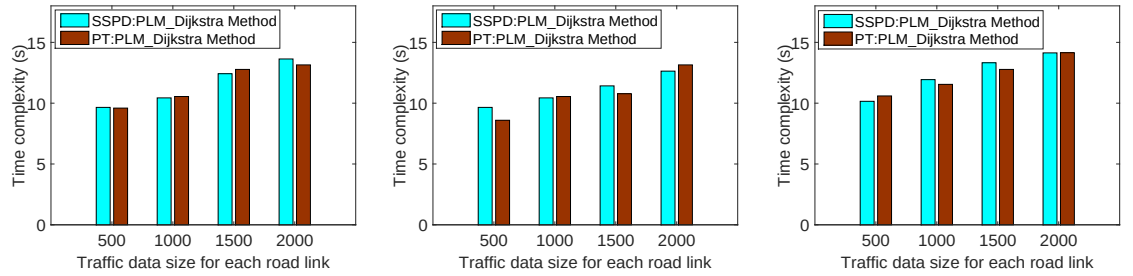
specified slots, and then apply them into the algorithm. The travel time data can be easily extracted according to the 'time slots' attribute in [71]. For each time dependent scenario, it again samples 500, 1000, 1500 and 2000 data on each road link, and then incorporates them into the matrix $\mathbf{W}$ in Eq. (4.1) and Eq. (4.2). Generally, finding a longer path will be more difficult in terms of computation. To better justify the proposed method, it randomly chooses reachable O-D pairs and only keep those which contain at least 150 road links on its 'shortest' path. Here, 'shortest' refers to minimum number of road links, and it can be easily achieved by setting each arc weight as 1. Other experiment settings are the same with those of Sections 4.4.1 and 4.4.2. Then, the average number is recorded in Fig. 4.7.

From Fig. 4.7 it can be observed that the average number of road links is around 200 for each O-D under the three time dependent scenarios, which means that on average, the driver should traverse at least 200 roads links to reach destination. With those O-D pairs, it then performs the PLM_Dijkstra method to solve the SSPD model and the PT model. The average



4.8.a 1am-4am 4.8.b 7am-10am 4.8.c 5pm-8pm

Figure 4.8: Average $f_{best}^{(k)}$ Over the Iterations for Different Time Slots



4.9.a 1am-4am 4.9.b 7am-10am 4.9.c 5pm-8pm

Figure 4.9: Computation Time with Respect to Different Time Slots

objective function for each iteration is shown in Fig. 4.8, and the average computation time of the PLM_Dijkstra method with respect to three different scenarios is shown in Fig. 4.9.

From Fig. 4.8 it can be observed that the average value of objective function $f_{best}^{(k)}$ always converges to optimal value f^* for the two models, within 30 iterations. The convergence is not as smooth as those in Fig. 4.2 and Fig. 4.5(a). The major reason is that in Sections 4.4.1 and 4.4.2, all traffic data is generated following normal distributions. Conversely, travel time data used in Section 4.4.3 is from real traffic, which is more chaotic and may not follow certain distribution. Another noticeable phenomenon is that the traffic is time dependent. The values of objective function, which is correlated with the potential delay, of 7pm-10pm and 5pm-8pm are much larger than that of 1am-4am. This is because 7am-10am and 5pm-8pm are peak hours, and more delay will be experienced.

From Fig. 4.9 it can be observed that, the computation time slightly increases as the traffic data size becomes larger for the two models, roughly from 9 to 14 seconds. It is much longer than that of Fig. 4.5(c), which is around one second. This is because the size of the Beijing road network (148,068 nodes and 329,148 arcs) is significantly larger. From this evaluation scenario which is considered to be realistic, the performance of the PLM_Dijkstra method on the Beijing road network is satisfactory. It is also highlighted that the B&B method in this case cannot be applied because it needs thousands of seconds to compute one path, which is impractical in practice.

Comparing Figs. 4.9(a), (b), and (c), it is also observed that different from the objective function, the computation time is independent of the time slots. This is because although the proposed method is data driven in nature, the computation complexity mainly depends on the size of the problem, which is up to polynomial, as analyzed in Section 4.3.2. In particular, the travel time will not significantly affect the computation.

Note that the largest graph that considered in Section 4.4 is Beijing city. The proposed method was not tested on continental level graphs. This is because the the arriving on time problem targets on the users in a city, where road traffic uncertainties often exist due to various factors. Nonetheless, if the driver plans to travel, e.g., from New York to San Francisco, the recommended approach is to first find a path by conventional deterministic path finding (i.e., the shortest distance). Then upon reaching San Francisco city, the driver can switch to the arriving on time solution to further find an optimal path.

4.4.4 Implementation on Real Navigation System

The experiments in Sections 4.4.1, 4.4.2 and 4.4.3 have justified convergence and efficiency of the proposed partial Lagrange multiplier method. Since the PLM_Dijkstra method is the most efficient, which is also the focus of this Chapter, its feasibility to solve real world SSP problem will be further assessed. Here, it implements the PLM_Dijkstra method into a vehicle routing simulator, i.e., Qtrip, which is a front-end user interface of the BMW's traffic routing system. To the best knowledge of the author, searching a path in Qtrip is the same as finding a path in a real commercial navigation system.

For the underlying graph, the whole Singapore road map (Fig. 4.10) is loaded into the simulator. In the graph, there are 18,246 nodes and 37,225 arcs. Each time it reads the travel time data on each road link from a database instead of directly generating them. The data size is still 500, 1,000, 1,500 and 2,000. Other experiment settings are the same as those of Sections 4.4.1 and 4.4.2.

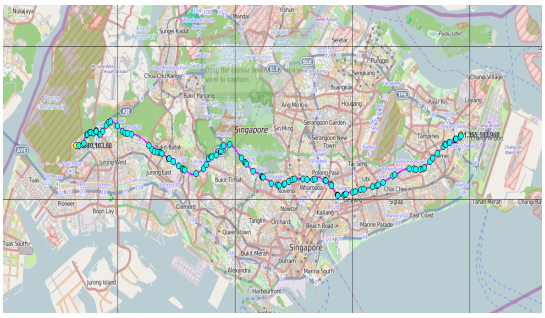


Figure 4.10: Singapore Road Network

The PLM_Dijkstra method is used to solve SSPD model and PT model for 1,000 times. For each model, the average time of the overall application processing or response time, and algorithm computation time are recorded in Table 4.1. The overall response time refers to the time duration from the moment when the user clicks "Search" button in the software to the moment when an optimal path is returned. This overall response time includes the time taken for extracting the graph information, reading traffic data from the database for each road link, and displaying the optimal path on the map. The algorithm computation time refers to the time duration to execute all the steps in Algorithm 1.

Table 4.1: Computation Time Complexity for the Real World Road Network

	500	1000	1500	2000
SSPD (Algorithm Computation)	2.31	2.39	2.37	2.42
SSPD (Overall Response)	6.32	7.39	8.53	9.73
PT (Algorithm Computation)	2.46	2.51	2.49	2.57
PT (Overall Response)	6.58	7.62	8.77	9.89

From Table 4.1, it is observeed that, on this large scale graph, the algorithm computation time of the PLM_Dijkstra is always smaller than 3s, and almost does not increase as the data size

increases. It is significantly efficient considering that the average computation time of the B&B method is always longer than 100s even for the medium scale graph as presented in Section 4.4.2. The overall response time is much longer than the algorithm computation time. This is because it includes additional system overhead, e.g., extracting the graph information from the map and displaying the optimal path on the map. It is also observed an increase for the overall response time as the data size scales up, and it is expected since the system needs more time to read the data from the database and needs more memory space to store them. However, the longest average overall response time for data size of 2,000 is still shorter than 10s, and it is acceptable considering that the routing is done for the whole city. Moreover, the optimal path of the two optimization models takes the stochastic nature of the traffic into consideration, and is more promising than the traditional shortest distance path or the least expected travel time path. Besides, in the real application, it may need less data for each road link, which will further reduce the overall response time. Therefore, it is concluded that the proposed partial Lagrange multiplier method, especially the PLM_Dijkstra method, can solve the real world vehicle routing problem with high efficiency. Note that it is easy to make the routing on Qtrip time dependent because the proposed method is data driven in nature. As long as traffic data matrix $\mathbf{W}$ in the two models is extracted according to the specific time slots, as described in Section 4.4.3, it is able to solve the time-dependent routing problem.

4.5 Summary

The PT model and SSPD model based SSP problems can be approximately solved by expressing them as MILP problems. The traditional way to solve these MILP problems is the B&B method, which is subject to exponential computation complexity. The incidence matrix in the equality constraint of these MILP problems satisfies the total unimodularity property, which helps to obtain an optimal solution by only solving a linear programming (LP) problem. However, the existence of inequality constraints always undermines this advantage. To fully utilize the benefit of the total unimodularity property, the partial Lagrange multiplier method have been proposed to relax the inequality constraints, which can be further efficiently solved using

the subgradient method. The proposed method can guarantee polynomial computation complexity. It has theoretically proved the convergence and the efficiency, which are also assessed by the experiments on the small, medium, and large scale graphs. Additionally, the experimental results on the Beijing road network with real traffic data have shown that the proposed method can efficiently solve the time-dependent arriving on time problem. The implementation on a navigation system based on the Singapore road network has further confirmed that the proposed method can be applied to efficiently solve the real world arriving on time problem. However, the polynomial computation complexity is not claimed for all general MILP problems because they are NP-hard [137]. The achievable efficiency only holds for problems with specific conditions as stated in Section 4.2.4, and the PT model and SSPD model based SSP problems happen to comply with them.

The proposed method significantly improves the computation efficiency in comparison with the B&B method in Chapter 3. The solution regarding the PT model based SSP problem in this chapter is accurate to the ℓ_1-norm relaxation in Chapter 3, but still an approximation to the cardinality minimization, thus also an approximation to the arriving on time problem. Therefore, there is still room to improve the accuracy of finding the real optimal path for the arriving on time problem.

Chapter 5

Improving the Accuracy

In Chapter 3, the arriving on time problem is expressed as a cardinality minimization problem, and it is relaxed using the ℓ_1-norm, which is further reformulated as an MILP problem. Consequently, the solution to the MILP problem is an approximation to the arriving on time problem. Although the computation efficiency is significantly improved in Chapter 4, the solution is directly targeting at the MILP problem, thus, still an approximation to the arriving on time problem. Therefore, there is still much room for improving the accuracy. To improve the accuracy of finding the real optimal path for the arriving on time problem, in this chapter[1]*, a practical Q-learning method is proposed. In particular, rather than treating the arriving on time problem as a monolithic non-convex optimization problem, it can also be treated as a sequential decision making problem. The proposed Q-learning method is based on the key insight that the arriving on time problem can be reformulated in such a way that dynamic programming approaches do apply. To overcome the disadvantage that the reformulation may blow up the state space, a neural network is proposed to approximate the value function, which generalizes over the continuous space of deadlines.*

[1]This chapter is developed based on my publication: Zhiguang Cao, Hongliang Guo, Jie Zhang, Frans Oliehoek and Ulrich Fastenrath. Maximizing the Probability of Arriving on Time: A Practical Q-Learning Method. 31th AAAI Conference on Artificial Intelligence (AAAI), 2017.

The proposed Q-learning method in this chapter is designed to maximize the probability of arriving on time, which is expressed as the ratio between the number of times in which the vehicle reaches destination before deadline, i.e., success, and the total number of travels. Specifically, each success event is considered as a *reward*, each pair of intersection and deadline are considered as a *state*, and the driving direction at each intersection is considered as an *action*. Thus, the travel time data of road links could be directly utilized to calculate the reward, and the complete optimal path can be found by deciding a best action at each intersection. In so doing, the main benefit of the proposed method is its practicality, in that it can be readily applied in the real world. Specifically, 1) it is interpretable by normal drivers: the converged Q-values have the practical meaning as the actual probabilities of reaching destination before deadline, thus the accuracy of finding the real optimal path gets improved; 2) it is computationally feasible even for large road networks; 3) it enables the capability of providing time-dependent path recommendations; 4) it directly utilizes available travel time data, so does not require any strong assumptions. Extensive experiments on both artificial and real road networks are conducted, and the results justify the significant advantages of the proposed method over others. Thus, the solution has high potential to be deployed in real vehicles for field test.

The remainder of this chapter is organized as follows. Section 5.1 introduces the background of the Q-learning method. Section 5.2 elaborates the basic Q-learning method which is used to handle the case of discrete deadlines. Section 5.3 develops a practical Q-learning method which handles the continuous deadlines and large scale network, where the dynamic neural network is adopted to learn a value function. Section 5.4 verifies the Q-learning method by conducting experiments both on artificial and real road networks. Section 5.5 summaries this chapter.

5.1 Background on Q-learning

To better illustrate that the Q-learning method can solve the arriving on time problem, the PT model is first re-interpreted in a simplified manner. Particularly, the problem can be mathematically described as follows: in a directed graph $G = (\mathcal{V}, \mathcal{L})$, $\mathcal{V}$ is a set of nodes representing

road intersections, $\mathcal{L}$ is a set of arcs representing road links, and o,$d \in \mathcal{V}$ represent the origin and destination, respectively. The objective is to maximize $Prob(\vec{\mathbf{w}}^\top \vec{\mathbf{x}} \leq \tau)$, where $\vec{\mathbf{w}}$ is a random vector containing the travel time for each road link, τ is the deadline (i.e., τ is the same with T in Eq. 3.2, which actually represents the time to deadline), and $\vec{\mathbf{x}}$ is a set of road links in which an element is "1" if the road link is on the corresponding path [17]. This problem is difficult to solve efficiently as it does not follow any typical optimization forms, e.g., convex optimization.

Rather than treating the problem as a monolithic non-convex optimization problem, it can also be treated as a sequential decision making problem. In particular, in the next section it will describe how the problem can be modeled as a *Markov decision process (MDP)* [138]. An MDP is defined by a five-tuple (S, A, $P^{s'}_{s,a}$, $R(s)$, $\gamma \in [0, 1]$), where S represents the state space ($s, s' \in S$), A represents the set of actions, $P^{s'}_{s,a}$ represents the transition distribution from s to s' by action a ($a \in A$), $R(s)$ represents the reward at s, and γ is the discount factor. The MDP aims to find an optimal policy π^* which maps states to actions in such a way that the expected cumulative discounted reward is maximized. In cases where the MDP is not known in advance, *reinforcement learning* (RL) methods can be used to learn the optimal policy [61]. For instance, Q-learning [139] repeatedly applies the following equation to sampled transitions (s, a, s', r):

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R(s') + \gamma \max_a Q(s', a) - Q(s, a)], \tag{5.1}$$

with learning rate $\alpha \in (0, 1]$. This can be shown to converge to an optimal Q-function: the converged Q-values represent the expected summation of discounted future reward, i.e., $Q^*(s, a) = E\left(\sum_{k=0}^{\infty} \gamma^k R_k\right)$. From Q^*, the optimal policy can be extracted by taking the greedy action in each state. (Please refer to [61] for more details.)

5.2 Basic Q-Learning for Discrete Deadlines

Before dealing with the more complex and realistic case of continuous time-to-deadline, the case where each time-to-deadline takes a value from a discrete finite set is treated. This allows for a relatively straightforward, but effective formulation of the problem as a discrete MDP, and thus the application of canonical RL algorithms, e.g., Q-learning.

5.2.1 MDP Formulation for PT Model

To solve the PT model based SSP problem by the Q-learning method, naturally, $s = \langle v, \tau \rangle$, $s' = \langle v', \tau' \rangle$, $v, v' \in \mathcal{V}$, and v' be the succeeding intersection of v. Γ ($\tau, \tau' \in \Gamma$) is the set of time-to-deadlines, and it has $\tau - t_{v,v'} = \tau'$, where $t_{v,v'}$ is the random travel time on road link $l_{v,v'}$ ($l_{v,v'} \in \mathcal{L}$), and τ' is the remaining deadline at v'. Note that, τ at origin o is determined by the user, and the remaining deadline τ' at intermediate intersection v' between origin o and destination d is determined by τ and travel time cost $t_{v,v'}$ on the associated road links. A represents driving directions. $P_{s,a}^{s'} = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the distribution of $t_{v,v'}$ by action a such that the vehicle will move from intersection v with time-to-deadline τ at step t, to intersection v' with time-to-deadline τ' at step $t + 1$. $R(s')$ is the immediate reward received after transiting to intersection v' with time-to-deadline τ' from intersection v with time-to-deadline τ. The reward depends on whether the vehicle arrives on time.

5.2.2 Q-Value Representation and Path Planning

The just defined MDP has a particularly nice property: the reward and discount factor can be defined in such a way that the converged Q-value represents the probability of arriving on time. Specifically, it denotes the immediate reward as $R(s') \in \{0, 1\}$ with $R(s') = 1$ if and only if $v' = d$ and $\tau - t_{v,v'} \geq 0$. Thus, the immediate reward is 1 only at the intersection node preceding d if the vehicle can arrive at the destination before the deadline; otherwise, $R(s') = 0$. The discount factor γ is set to be 1. (General Q-learning with $\gamma = 1$ does not necessarily ensure convergence, but for SSP problem, convergence can be guaranteed with $\gamma = 1$ [140]).

In this case, after an o-d pair is determined, Q-value is updated using Eq. (5.1) for each intersection, starting with v (i.e., $v = o$) in each iteration. To transit to next intersection v', it employs the action selection policy, i.e., Softmax strategy [61]. This strategy balances between exploration and exploitation for the candidate succeeding intersections. Then, it uses $P_{s,a}^{s'}$ to sample $t_{v,v'}$ to decide τ' at v'. These steps are repeated until all Q-values converge.

The converged Q-values in the problem can be shown to be the probabilities of arriving on time in the following. From Eq. (5.1), Q-value converges when $Q_{k+1}(s, a) = Q_k(s, a)$, $\forall s$

and $\forall a$. After the convergence, it has the Q-value defined as $Q^*(s,a)$ given by $Q^*(s,a) = Q^*(s,a)+\alpha_k[R(s')+\gamma \max_a Q^*(s',a)-Q^*(s,a)]$. After a simple combination and elimination, it has

$$Q^*(s,a) = R(s') + \gamma \max_a Q^*(s',a). \tag{5.2}$$

As this equation will be executed for a large number of times given different samples of rewards, in essence, it is the averaged/expected reward. Thus, Eq (5.2) can be written as $Q^*(s,a) = E\left(R(s')\right) + \gamma \max_a Q^*(s',a)$, which is exactly the probability of arriving on time.

When Q-learning has converged, it can obtain an optimal path by first stopping the Softmax strategy and then determining the best action a^* at a specified intersection with a certain deadline as $a^*(\langle v,\tau\rangle) = \arg\max_a Q(\langle v,\tau\rangle, a)$. It first computes $a^*(\langle o,\tau\rangle)$ to determine the optimal driving direction at origin o, i.e., the current state, with a user-defined deadline. The direction will determine the next intersection and the remaining deadline, i.e., next state. Then, the second intersection with the remaining deadline is inputted, to find the next intersection. The same process is repeated until the destination is reached. Thus, the complete optimal path can be found.

5.3 Q-Learning for Continuous Deadlines

The basic version of the proposed Q-learning method introduced in the previous section will become prohibitively time-consuming in the case of continuous deadlines (i.e., the time-to-deadline is also continuous) and large-scale road networks due to the huge size of state space. To address those challenges, a practical value function approximation method is developed based on the dynamic neural network to directly learn the optimal Q-values, which approximates the probability of arriving at the destination on time. In the following, the value function update scheme, function approximation method and the deployment steps will be introduced.

5.3.1 Value Function Update Scheme

In order to deal with continuous deadlines and large scale networks, the proposed approach uses approximator $f(\cdot)$ to represent the value function $V(s)$, where $V(s) = \max_a Q(s,a)$. The special structure of the problem is proposed to be exploited: even for large networks with thousands of nodes, the locations can still be easily enumerated. Therefore, the value function fitting operator is applied by enumerating all the locations, but sampling from the travel time distributions. This means that all locations get the same amount of coverage, leading to a good approximation in entire network[2]. The value function with any time-to-deadline for any intersection is represented as:

$$V_{k+1}(s) = \max_a \left\{ \sum P_{s,a}^{s'} [V_k(s') + R(s')] \right\}, \tag{5.3}$$

where $0 \leq V(s') \leq 1$, $V(s') = 0$ if $v' = d$, $s = \langle v, \tau \rangle$, and $s' = \langle v', \tau' \rangle$. Eq. (5.3) uses the expected function value at s' (for the succeeding location) in the k^{th} iteration to compute the function value at s in the $(k{+}1)^{th}$ iteration. In particular, $V_k(s')$ on the right hand side of Eq. (5.3) is computed through the approximator $f_k(\cdot)$ in that iteration.

5.3.2 Function Approximation through Neural Network

To approximate $V_k(\cdot)$, the proposed approach adopts a two-layer dynamic neural network[3] to learn function $f_k(\cdot)$ in each iteration as the probability of arriving on time for a given intersection with a given deadline. Specifically, $f_k(\cdot)$ is expressed as follows:

$$f_k(\vec{\mathbf{z}}) = g_2(g_1(\vec{\mathbf{z}} \cdot \vec{\mathbf{w}}_1 + u_1) \cdot \vec{\mathbf{w}}_2 + u_2), \tag{5.4}$$

where $\vec{\mathbf{z}}$ is the feature; $\vec{\mathbf{w}}_1$, u_1, $\vec{\mathbf{w}}_2$, and u_2 are parameters of the neural network; $g_1(\cdot)$ and $g_2(\cdot)$ are activation functions.

[2]To save computation time, utilizing a subset of locations is also feasible, which is shown in the experiments, i.e., Fig. 5.4(a).

[3]Other supervised machine learning methods, such as support vector regression, can also be employed for function approximation. It uses the dynamic neural network here because it can naturally output values between zero and one (due to its sigmoid function) for representing the probabilities of arriving on time.

The feature $\vec{\mathbf{z}}$ used to represent s is defined as follows: $\vec{\mathbf{z}}(s) = \langle \tau, \vec{\mu}(v), \vec{\sigma}(v) \rangle$, where τ and v are the time-to-deadline and location in s; $\vec{\mu}$ and $\vec{\sigma}$ are the means and standard deviations of travel time for the K-shortest paths from current location to destination. The rationale for $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ is that the probability of arriving on time is highly associated with the mean and standard deviation of travel time. $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ can be easily computed based on the travel time data on each road link. Consequently, the feature of a training sample for s can be expressed as $\vec{\mathbf{z}} = [\tau, \vec{\mu}(v), \vec{\sigma}(v)]^\top$, where the length of $\vec{\mu}(v)$ and $\vec{\sigma}(v)$ is K. Since $f(\vec{\mathbf{z}})$ will be applied for continuous deadlines, τ in training samples should cover extensive values. Therefore, $\mathcal{N}$ deadlines are randomly selected, each of which is denoted by τ_i, and it has $\tau_i = \beta \cdot T_e$, where β is the deadline parameter, and $\beta \in [0, 2]$. T_e is the least expected travel time from v to d. Note that larger β implies loose deadlines, and vice versa. Thus, the entire feature vector of an individual intersection v is expressed as:

$$\vec{\mathbf{Z}}_s = [\tau_1, \vec{\mu}(v), \vec{\sigma}(v); \cdots ; \tau_{\mathcal{N}}, \vec{\mu}(v), \vec{\sigma}(v)]^\top. \tag{5.5}$$

To dynamically train the neural network at the $(k{+}1)^{th}$ iteration, the proposed method sets $V_k(s)$ on the right-hand side of Eq. (5.3) by the learned $f_k(\vec{\mathbf{z}})$, and $V_{k+1}(s)$ on the left-hand side can then be updated accordingly. This value will be adopted as the new label to train $f_{k+1}(\vec{\mathbf{z}})$ in the next iteration through back propagation. This process is repeated until convergence.

5.3.3 Core Deployment

Considering both the value function update scheme and function approximation method, it deploys the core part (i.e., training value function) of the proposed method as shown in Alg. 2. Specifically, Lines 1-2 initialize parameters and prepare feature values. Lines 3-20 dynamically learn value functions for each intersection using the neural network. In particular, for each intersection, the trained neural network from the preceding iteration computes the value functions of the succeeding intersections, which are then used to update the value function of the current intersection (Lines 5-14). In Lines 15-16, convergence errors of all intersections are accumulated. In Lines 17-18, the average convergence error is updated to determine the loop

Algorithm 2: Value Function Training

input : $G = (\mathcal{V}, \mathcal{E})$, road network;
$\Omega_{v,v'}$, set of travel time data on $l_{v,v'}$;
N, size of travel time data on each road link;
$\mathbb{N}$, the configured neural network; o-d pair;
ϵ_1, the convergence error; k, the iteration number.

1 Initialize $V_0(s)$ via warm start; and set $\epsilon_1 = 1$; $\epsilon_2 = 0$; $k = 0$; $N = 1000$; $\zeta = 0.025$;
2 Import feature data $\vec{\mathbf{Z}}_s$ for each s by Eq. (5.5);
3 **while** $\epsilon_1 > 0.01$ **do**
4 **foreach** $v \in \mathcal{V}$ **do**
5 **foreach** v' *that succeeds* v **do**
6 Sample N travel time $t_{v,v'}$ out of $\Omega_{v,v'}$;
7 **if** $v' \neq d$ **then**
8 **if** $k \geq 1$ **then**
9 Calculate $V_k(s')$ in Eq. (5.3) through the trained $f_k(\vec{\mathbf{z}})$ in Eq. (5.4);
10 **else**
11 $V_k(s') =$ initial values $V_0(s')$;
12 **else**
13 Compute $R_k(s')$ in Eq. (5.3);
14 Update $V_{k+1}(s)$ in Eq. (5.3);
15 **if** $k \geq 1$ **then**
16 Update $\epsilon_2 = \epsilon_2 + |V_{k+1}(s) - V_k(s)|$;
17 **if** $k \geq 1$ **then**
18 Update $\epsilon_1 = \epsilon_2/|\mathcal{V}|$; Reset $\epsilon_2 = 0$;
19 Learn $f_{k+1}(\vec{\mathbf{z}})$ by incorporating feature $\vec{\mathbf{Z}}_s$ and new label $V_{k+1}(\langle v, \tau \rangle)$ for each v into $\mathbb{N}$;
20 Update iteration number: $k = k + 1$;

output: Converged $V^*(s)$ for each $\langle v, \tau \rangle$.

termination. In Lines 19-20, the neural network is trained using feature values and updated labels.

To find the optimal path before departure (which is preferred in a real application), the best

action at a certain intersection with a given deadline is determined by:

$$a^*(s)=\arg\max_a \left\{ \sum P_{s,a}^{s'}[V^*(s')+R(s')] \right\}, \tag{5.6}$$

where $P_{s,a}^{s'}$ is represented by travel time data on corresponding road link. The next state s' (which consists of v' and τ') can be decided as follows: v' is the succeeding intersection, and $\tau' = \tau - E(t_{v,v'})$, where $E(t_{v,v'})$ is the expected travel time over the previously chosen road link. Then it iteratively uses the same equation to compute the next action until the complete path is achieved.

5.3.4 Other Practical Considerations

Here, some other practical features considered in the proposed method would be discussed, including time-dependent path recommendation, simultaneous training for multiple destinations and accelerating the training process.

If the data $P_{s,a}^{s'}$ is collected according to a specified time period, such as peak hour of 7am-9am, the proposed Q-learning method is able to provide time-dependent path recommendation. Also note that the Q-learning method adopts Alg. 2 to dynamically learn the value function for a specified destination from all nodes. If the destination is changed, Alg. 2 needs to run again to learn a new value function. To better generalize the proposed Q-learning method, multiple destinations are randomly selected on the road network, and concatenate their feature values to the same matrix (i.e., $\vec{\mathbf{Z}}_s$ in Eq. (5.5)). Then it only needs to run Alg. 2 once, and the learned value function will be applied to any other destinations on the same road network. When the destination is changed, it just changes the input features (i.e., $\vec{\mu}$ and $\vec{\sigma}$ of the K-shortest paths to the new destination) to the trained neural network. In this way, the proposed Q-learning method is able to calculate the probability of arriving on time from any origin(s) to any destination(s).

Traditionally, value function $V_0(\langle v, \tau \rangle)$ is initialized as 0, which may slow down the convergence speed [141]. Therefore, a *warm start* strategy[4] that approximates the probability of

[4]This warm start strategy is also applicable to the basic version of the Q-learning method for discrete deadlines as long as the Q_0 is calculated in the same way.

arriving on time for vehicles at intersection v with time-to-deadline τ is proposed as follows: $V_0(\langle v, \tau \rangle) = 1/(1 + e^{-\zeta(\tau - T_e)})$, where ζ is the coefficient. The rationale is that it needs to increase the probability from 0 to 1 (that is why sigmoid function is used) as τ increases.

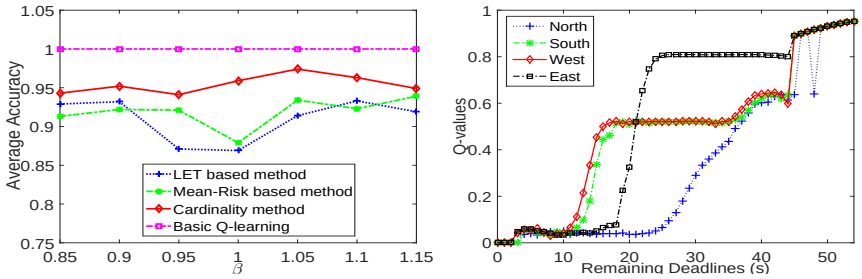
5.4 Experimentation

Extensive experiments are conducted to evaluate the proposed Q-learning method and compare with other methods on various road networks. All experiments are performed on a typical PC with Intel Core i7-3540M processor and 16GB RAM.

5.4.1 Artificial Network with Discrete Deadlines

The basic version of the proposed Q-learning method for discrete deadlines is first tested on an artificial road network – a grid with 20×20 intersections. It uses $m_1 = 15$ as mean and $\sigma_1 = 3$ as standard deviation to randomly generate the mean travel time m_2 for each individual road link. It then adopts m_2 and $\sigma_2 = 0.3m_2$ to generate 200 instances of travel time data for corresponding links to represent $P_{s,a}^{s'}$. All travel time is rounded up as positive integers in seconds. At each intersection, there are four travel directions (north, south, west and east). Three methods are compared: (1) LET based method computes a path of the least expected travel time based on the travel time data; (2) Mean-Risk based method computes a path with minimal value of $\mu_p + \lambda\nu_p$, where μ_p is the expected travel time, ν_p is the variance, and λ is the coefficient [75]; and (3) Cardinality method approximates and computes a path of the PT model by formulating it as the MILP problem, which is solved using the partial Lagrange multiplier method proposed in Chapter 4. It randomly selects 100 o-d pairs and use deadline parameters $\beta = 0.85, 0.90, \ldots, 1.10, 1.15$. For each o-d pair and specified deadline, it defines the ground-truth path (achieved through enumerating all the paths) as the one having the maximal number of times of not being late, given the 200 instances of data on each link.

The accuracy of finding the ground-truth path is plotted in Fig. 5.1(a). For all deadlines, the Q-learning method always achieves average accuracy of 100%, confirming that the converged Q-values are able to represent probabilities of arriving on time. The cardinality method obtains the



5.1.a different deadlines 5.1.b converged Q-values

Figure 5.1: Artificial Network with Discrete Deadlines

average accuracy of about 95% as it adopts ℓ_1-norm to approximately minimize the frequency of not being late. The LET and Mean-Risk based methods obtain the average accuracy of around 88%. The high accuracy of the proposed method also shows that the recommended path guarantees arriving on time more often than others. This advantage comes from the fact that the proposed method incorporates the event of arriving on time in the reward and objective.

The accuracy of the proposed method can also be demonstrated by the plot of converged Q-values for an intersection which is two grids to the north of the destination as shown in Fig. 5.1(b). The converged Q-values in [0, 1] denote the probabilities of arriving on time from the current location by taking a corresponding action. Meanwhile, the optimal action usually changes with deadlines, e.g., traveling west is optimal when the time-to-deadline is between 10 and 20 and traveling east is optimal when the time-to-deadline is larger than 20. The Q-value for traveling south is not higher than that of traveling west and east although the current location is north to the destination. This is because the cost of traveling south is always large according to the generated traffic data. So, traveling south yields a lower chance of arriving on time. It is also observed that the Q-values for all actions are 0 when time-to-deadline is less than 4, which is too tight. Thus, it is concluded that an optimal path depends strongly on deadline even though an origin is fixed. This factor is not incorporated in LET or Mean-Risk based methods, which is why their average accuracy fluctuates when deadline is varied.

5.4.2 Real Networks with Continuous Deadlines

To verify the proposed Q-learning method for the PT model with continuous deadlines, experiments on three large road networks extracted from the city maps of Munich, Singapore, and Beijing are performed, as summarized in Table 5.1. It randomly selects 200 o-d pairs on each network, and the average minimal number of road links between each origin and destination is given in the third row. 1,000 instances of travel time data for each road link are prepared: (1) On Munich network, it uses actual length of each road link to divide the collected real travel speed of vehicles from July 2014 to March 2015[5]; (2) On Singapore network, it uses actual length of each road link as the mean to randomly generate travel time data, and the standard deviation is 0.3 times of the length; (3) On Beijing network, it directly uses the travel time data collected from real travel trajectories of taxi from September to October 2013 [71][6]. For the proposed method, 100 destinations on a network are randomly selected and their feature values are concatenated to a same matrix (i.e., $\vec{\mathbf{Z}}_s$ in Eq. (5.5)). Note that the author did not explicitly investigate the dependence or correlation of travel time on different road links, because it will be naturally included in the real-world data set if there is any, and the travel time data is the direct input to the proposed method.

Table 5.1: Settings of the Three Real Road Networks

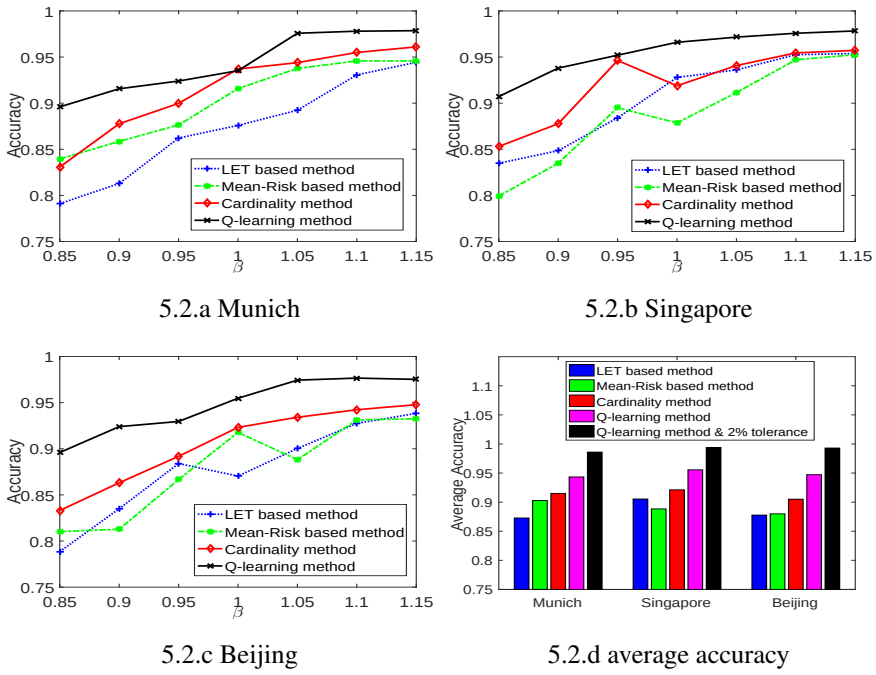
	Munich	**Singapore**	**Beijing**
#Nodes	51,517	6,476	129,607
#Links	115,651	10,225	294,868
#Links between o-d	221	86	358

5.4.2.1 Accuracy

The accuracy of each method on the three networks are plotted in Figs. 5.2(a-c). The proposed Q-learning method achieves around 95% accuracy, higher than all other methods for

[5]The dataset is provided by the BMW Group.

[6]The dataset is available at: https://www.microsoft.com/en-us/research/people/yuzheng/



5.2.a Munich 5.2.b Singapore

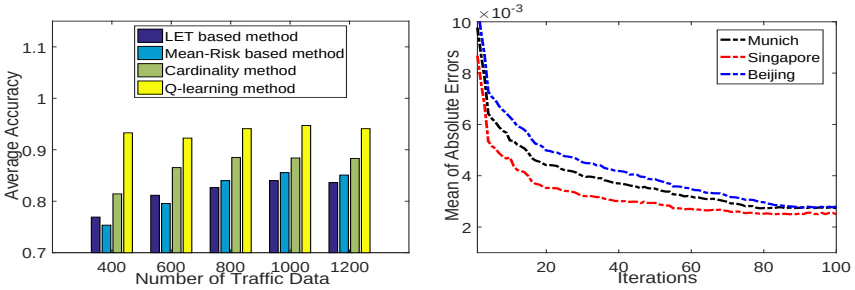
5.2.c Beijing 5.2.d average accuracy

Figure 5.2: Accuracy Results on the Three Real Road Networks with Continuous Deadlines

each deadline. Compared with the results in Fig. 5.2(a), the proposed method achieves slightly lower accuracy, because, to handle continuous deadlines, a neural network is used to learn value functions in which learning error may exist. Besides, the three real networks are considerably larger than the grid. The accuracy of the cardinality method is higher than the LET and Mean-Risk based methods. Moreover, the accuracy of all methods becomes lower as deadline becomes tighter. The proposed method is less affected because it dynamically updates Q-values for each specified deadline.

The overall accuracy of all methods is also plotted in Fig. 5.2(d), as well as the 2%-tolerance accuracy[7] of the proposed Q-learning method. The overall accuracy on Munich and Beijing is slightly lower than that on Singapore for most of the methods. This is because the sizes of Munich and Beijing networks are much larger, so as the number of possible paths between an o-d pair, thus making it more difficult to achieve the ground-truth path. In particular, the proposed

[7]If the difference between the probability of arriving on time for the path returned by the proposed method and the probability for the ground-truth path is less than 2%, then the returned path is considered the same as the ground-truth path as the difference is very marginal.



5.3.a different data size 5.3.b mean absolute error

Figure 5.3: Accuracy Changes with Data Size and Iterations

method will determine a road link to traverse according to Eq. (5.6) at each intersection. Even if only one road link is not on the ground-truth path, the returned path is not counted as accurate. In the case with at least 358 intersections on average (Table 5.1) for each path finding on Beijing network, the overall accuracy of around 95% is sufficiently high. Moreover, the 2%-tolerance accuracy of the proposed method is nearly 100%, indicating that the paths found by the proposed method is of high quality even if they might not be the ground-truth path.

The accuracy against the quantity of travel time data on each road link is also tested by taking Beijing network as an example since its size is largest. As shown in Fig. 5.3(a), the proposed method is almost not affected by the quantity of travel time data. By contrast, the accuracy of other methods is reduced when data size is small. Moreover, in each iteration of the value function learning, it records the mean absolute error (i.e., the difference between the neural network output and the target value) on the three road networks, shown in Fig. 5.3(b). The average absolute error becomes smaller as the iteration increases. This clearly justifies the efficacy of the dynamic neural network to learn accurate value functions.

The proposed Q-learning method is data-driven and easy to execute in a time-dependent manner, which further improves the performance of routing service, i.e., avoiding frequent value function learning while maintaining an acceptable level of accuracy. To justify, it explores the travel time data of eight continuous weeks (July-August 2014) on Munich network and further extract eight segments according to different time units: (1) **week**, it divides the data into eight weeks, and each segment represents a week, (2) **day**, it extracts the data of eight Mondays separately, and each segment represents a Monday, (3) **hour**, it extracts the data of 7am-9am

(i.e., peak hours) on each Monday separately, and each segment represents a span of peak hour. For all of the three units, the proposed method is applied only on the first segment of the traffic data and then use the learned value functions to perform routing service based on all of the eight segments accordingly. The accuracy results are recorded in Table 5.2.

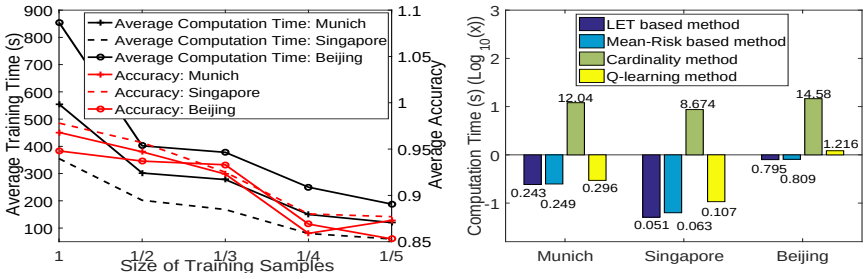
Table 5.2: Accuracy of Time-Dependent Q-learning (%)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
week	94.8	93.7	95.1	93.3	94.5	92.9	90.8	91.6
day	96.1	95.0	94.2	95.5	95.8	96.3	94.1	93.2
hour	97.4	97.7	97.0	96.2	97.3	98.2	96.5	96.1

It can be observed that, segment 1 usually achieves higher accuracy than that of the other seven segments. This is because the training data and testing data are the same for segment 1. Generally, as the segment index increases, the accuracy will decrease, but the deterioration is very slight, especially for the **hour**, which achieves the accuracy higher than 96% even for week 8. Another remarkable observation is that the higher the time resolution, the better the accuracy. This happens because the high time resolution usually captures the traffic characteristics better, i.e., the peak hour pattern captured by the **hour** unit. Thus, it can conclude that the time-dependent Q-learning method helps to avoid frequent value function learning while achieving good accuracy, especially for the time unit **hour**.

5.4.2.2 Computation Time

Normally, the computation time in each iteration of the dynamic neural network is associated with the size of training samples. Previously, training samples of all nodes are used to learn the value function. To improve computation efficiency, it may only use samples from some nodes, while maintaining an acceptable level of accuracy. Thus, the average computation time of the proposed method in each iteration and its accuracy with respect to different sizes of training samples are evaluated, as plotted in Fig. 5.4(a). As expected, the computation time and the accuracy decrease as the training sample size reduces. On Beijing network, the deterioration



5.4.a different training sample size 5.4.b average computation time

Figure 5.4: Computation Time

of accuracy is only around 1% while the reduction of computation time is about 360 seconds (i.e., almost half) as it uses no less than 1/3 of the training samples.

The average computation time of each routing for all methods is also recorded in Fig. 5.4(b). It only counts the time of path finding for the Q-learning method since learning value function can be done offline. The LET and Mean-Risk based methods have the shortest average computation time, which slightly increases as network size increases. Although the computation efficiency of the cardinality method is improved using partial Lagrange multiplier method in Chapter 4, it still needs the longest time (around 15 seconds) to compute a path on Beijing network in comparison with other three methods. By contrast, the proposed Q-learning method takes slightly longer than the LET and Mean-Risk based methods. It takes only 1.216 seconds to obtain an optimal path on Beijing network, which is efficient.

5.5 Summary

In this chapter, a practical Q-learning method is designed to more accurately solve the arriving on time problem. Compared with the partial Lagrange multiplier method in Chapter 4, the accuracy and computation efficiency of finding the real optimal path are further improved. Additionally, in comparison with other baselines in this chapter, the proposed method offers several important practical features: 1) It considers travel time, risk and deadlines simultaneously when computing an optimal path; 2) It can easily provide the probability of arriving

on time from any origin(s) to any destination(s) once the value function is learned; 3) It can flexibly provide the time-dependent path recommendation and avoid frequent training even if the travel time data changes. The extensive experimental results for both artificial and real road networks justify the significant advantages of the proposed method over others. Thus, the proposed Q-learning solution has the high potential to be deployed in real vehicles for the field test.

Chapter 6

Arriving on Time for Multiple Cooperative Vehicles

In this chapter[1]*, a decentralized multiagent-based approach is proposed to solve the arriving on time problem for multiple cooperative vehicles, which aims to increase the chances of arriving on time for all vehicles. The motivation is stated as follows: (1) On one hand, the multiagent-based approach can handle the traffic dynamics well by exploring the intentions of vehicles. However, most of multiagent-based route guidance purely provide LET paths to vehicles. Although it helps to reduce the total travel time, the driver utilities are often governed by deadlines, which cannot be fully met by LET paths. (2) On the other hand, the arriving on time problem for a single vehicle considers deadlines. However, most of existing approaches incorporating the PT model independently pre-calculate a path for each individual vehicle before it departs, without considering the intentions of others. Since traffic is always dynamic, optimality of a pre-computed path may not hold any more when all vehicles are en-route due to the influence from each other. Combining the above two aspects, it is desirable to leverage both the advantages of a decentralized multiagent-based approach and the PT model.*

[1]This chapter is developed based on my publication: Zhiguang Cao, Hongliang Guo, Jie Zhang and Ulrich Fastenrath. Multiagent-based Route Guidance for Increasing the Chance of Arrival on Time. 30th AAAI Conference on Artificial Intelligence (AAAI), pp. 3814-3820, 2016.

The decentralized multiagent-based approach in this chapter is characterized by two types of agents, i.e., vehicle agents and infrastructure agents. Infrastructure agents locally collect intentions of concerned vehicle agents and transform the route guidance problem into a route assignment problem, to guarantee their arrival on time. In addition, the predicted travel time on each assigned road link is refined by iteratively linearizing a non-linear function to achieve more accurate approximation; the computational efficiency is enhanced by introducing additional variables and reformulating the route assignment from an MIQP (mixed integer quadratic programming) problem [142] to an MILP problem; and the overall performance is improved by allowing communication between neighboring infrastructure agents. Accordingly, in this chapter, it makes two major contributions: 1) To the best knowledge of the author, it is the first one to consider arriving on time for all vehicles in the multiagent-based route guidance. 2) The proposed approach is decentralized and provides local immediate route guidance at each intersection by solving a route assignment problem, which can save computation cost in comparison with the centralized route guidance.

The remainder of this chapter is organized as follows. Section 6.1 introduces the framework of the proposed multiagent-based route guidance. Section 6.2 elaborates the proposed approach that considers arriving on time for all vehicles, which is expressed as a route assignment problem. Section 6.3 proposes three improvements to the route guidance approach. Section 6.4 presents experimental settings, comparative results and corresponding analysis. Section 6.5 gives the summary for this chapter.

6.1 Multiagent-based Route Guidance

In this section, the scheme of the proposed decentralized multiagent-based approach for vehicle route guidance is briefly introduced, which involves two types of agents, i.e., vehicle agents and infrastructure agents. Vehicle agents representing drivers, travel on a road network by following route guidance. Infrastructure agents, located at road intersections, collect intentions (i.e., deadlines and destinations) from vehicle agents.

In traffic, vehicles may influence each other due to limited road capacity and rush hour effect. To consider such influence, intention collection is necessary [47, 48, 49]. In the proposed approach, each vehicle agent determines a destination and a preferred deadline before departure, and travels along an initial route given by the probability tail model in Eq. (4.1). Each infrastructure agent is associated with all traffic lights at a road intersection. It collects intentions from vehicle agents, which are (1) located on road links directly connected to that infrastructure agent; and (2) facing red lights. The motivation for the latter is to avoid unnecessary and frequent route change. Facing green lights may imply that the current route is sufficiently satisfactory.

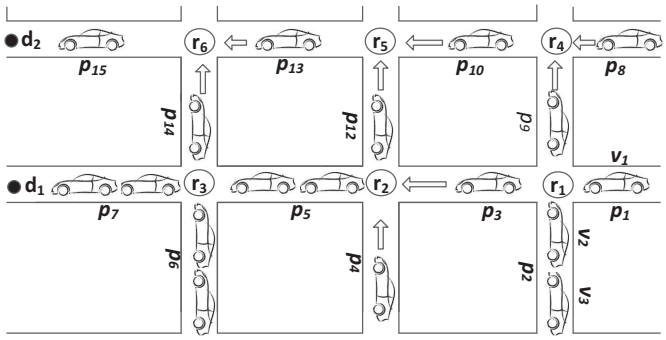


Figure 6.1: Two Types of Agents and Intention Collection

Take a one-way road network in Fig. 6.1 as an example, where r_i is an infrastructure agent, v_j is a vehicle agent, and p_k is a road link. Assume that at this moment, the traffic light associated with r_1 shows red color to p_1 and p_2. Then r_1 collects intentions of v_1, v_2 and v_3. Since this red color will last for a while, r_1 will also collect intentions of other vehicle agents if they later enter p_1 or p_2 during the same red color period. Other infrastructure agents also work in the same manner.

Once the intentions are collected, the infrastructure agent at each road intersection will compute and provide route guidance, to increase the chance of arrival on time for all concerned vehicle agents. Then the vehicle agents update routes accordingly. Vehicles generally have different types of deadlines. Simply seeking LET paths for all vehicles may cause some vehicles with tight deadlines to miss their deadlines due to the influences from other vehicles with loose

deadlines. Motivated by this concern, a desirable approach is to distribute vehicles with loose deadlines to detour crowded paths, to make ways to those with tight deadlines. On the other hand, vehicles always face choices at an intersection: go straight, turn left, turn right or turn back to enter the next road link. Thus, in the proposed approach, the infrastructure agent at each road intersection provides route guidance to vehicle agents by formulating it as a route assignment problem [143], which incorporates the collected intentions.

6.2 Route Assignment

In this Section, the proposed multiagent-based approach to solve the problem of arriving on time for all vehicle agents is elaborated, which formulates the route guidance at each intersection as a route assignment problem.

6.2.1 Route Assignment Formulation

Take infrastructure agent r_1 and vehicle agents v_1, v_2 and v_3 in Fig. 6.1 as an example, and focus on the route assignment for v_1. Assume that: (1) destination of v_1 is d_2, and its preferred deadline is T_1; (2) v_1, v_2 and v_3 are currently facing red light, and will next enter p_3 or p_9. Assignment for an vehicle agent always influences others. Suppose that the predicted travel time of v_1 on p_3 and p_9 are T^p_{13} and T^p_{19}, which linearly depend on the number of assigned vehicle agents to the road links and their lengths. Besides, it needs to consider traffic conditions from r_2 and r_4 to d_1 or d_2, where historical expected travel time is used because r_2 and r_4 are comparatively far away from r_1. Assume that the expected travel time from r_2 and r_4 to d_2 are T^e_{22} and T^e_{42}. Hence, if v_1 is assigned to p_3 or p_9, there would be relative deadlines on p_3 and p_9 for v_1, denoted as T^r_{13} and T^r_{19}. Since deadlines should always be non-negative, it has $T^r_{13} = max\{0, T_1 - T^e_{22}\}$, and $T^r_{19} = max\{0, T_1 - T^e_{42}\}$. Thus there would be a potential delay ξ_1 for v_1, satisfying $T^p_{13} - T^r_{13} \leq \xi_1$ and $T^p_{19} - T^r_{19} \leq \xi_1$ (i.e., ξ_1 is non-negative, and there is no delay only if $\xi_i = 0$). To guarantee arrival on time for the three vehicle agents, r_1 should minimize the cardinality[2] of $\vec{\xi}$ whose components are ξ_i ($i = 1, 2, 3$).

[2]Cardinality is the number of non-zero components in a vector.

In view of the above example, the route assignment performed by each infrastructure agent can be generalized. First introduce several symbols: (1) l_j, choices of road links to enter next, such as p_3 and p_9 in Fig. 6.1; (2) $x_{ij} \in \{0, 1\}$, $x_{ij} = 1$ means that v_i is assigned to l_j, otherwise $x_{ij} = 0$; (3) $I = \{1, ..., Q\}$ and $J = \{1, ..., L\}$, indices of vehicle agents and road link choices associated with the infrastructure agent. Then the problem of maximizing the chance of arrival on time for all concerned vehicle agents can be expressed as minimizing the cardinality of the potential delay $\vec{\xi}$, as follows:

$$\min_{\vec{\mathbf{x}}} Card(\vec{\xi}) \left| \begin{array}{l} \sum_{j \in J} (f_j(\vec{\mathbf{x}}) - T_{ij}^r) \cdot x_{ij} \leq \xi_i, \forall i \in I; \\ \sum_{j \in J} x_{ij} = 1, \forall i \in I; \xi_i \geq 0; x_{ij} \in \{0, 1\}, \end{array} \right. \tag{6.1}$$

where $\vec{\xi} = \{\xi_1, ...\xi_Q\}$, and delay occurs for v_i if $\xi_i > 0$; $\vec{\mathbf{x}}_j = (x_{1j}, ..., x_{Qj})$, indicates assignments to l_j; $f_j(\vec{\mathbf{x}})$, a linear function, denotes predicted travel time on l_j; T_{ij}^r is relative deadline for v_i on l_j; $\sum_{j \in J} x_{ij} = 1$ ensures that v_i can only enter one road link, thus only one potential delay takes effect in $\sum_{j \in J} (f_j(\vec{\mathbf{x}}) - T_{ij}^r) \cdot x_{ij}$. Particularly, the linear function $f_j(\vec{\mathbf{x}})$ for l_j is expressed as:

$$f_j(\vec{\mathbf{x}}) = c_j \sum_i x_{ij} + \gamma_j \tag{6.2}$$

where $\sum_i x_{ij}$ is amount of vehicle agents assigned to l_j, c_j and γ_j are coefficients. The cardinality minimization in Eq. (6.1) is difficult to be directly solved. It thus uses ℓ_1-norm to approximately solve it [104], which can be further expressed as:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{Q} \xi_i \left| \begin{array}{l} \sum_{j \in J} (f_j(\vec{\mathbf{x}}) - T_{ij}^r) \cdot x_{ij} \leq \xi_i, \forall i \in I; \\ \sum_{j \in J} x_{ij} = 1, \forall i \in I; \xi_i \geq 0; x_{ij} \in \{0, 1\}. \end{array} \right. \tag{6.3}$$

Eq. (6.3) is a mixed integer quadratic programming (MIQP) problem in nature, which can be solved by existing solvers.

Note that: (1) Eq. (6.3) only outputs a road link for a vehicle agent to enter next, but the remaining path from assigned road link to destination is also available due to the computation of relative deadline T_{ij}^r. The vehicle agent can then follow this complete route if it does not receive

any further guidance after an route assignment, which may happen if afterwards it always faces green light; (2) As time elapses, deadline T_j will decrease, and it always uses the latest T_j when route guidance is performed; (3) It previously takes a simple one-way road network as an example, and double-way road links can also be easily applied, as long as infrastructure agents dynamically recognize on which road links vehicle agents are facing red light, and which road links are available to be assigned to those vehicle agents.

6.2.2 Bound Analysis

The ℓ_1-norm minimization in Eq. (6.3) is an approximation to the cardinality minimization in Eq. (6.1), and its bounds is analyzed in this subsection. It is assumed that regarding Eq. (6.3), $\vec{\mathbf{x}}_1^*$ and $\vec{\xi}_1^*$ are optimal solution to the primal problem, and p_1^* is the optimal value of objective function. It is also assumed that the optimal value of objective function in Eq. (6.1) is p_2^*. Since Eq. (6.1) and Eq. (6.3) share the same constraints, $\vec{\mathbf{x}}_1^*$ and $\vec{\xi}_1^*$ are feasible solution to the cardinality minimization problem in Eq. (6.1). Therefore p_1^* is an upper bound to p_2^*. The dual function of Eq. (6.1) can be expressed as $g(\vec{\lambda}, \vec{\nu}) = \min_{\vec{\mathbf{x}}, \vec{\xi}} \text{Card}(\vec{\xi}) + \sum_{i \in I}(\nu_i(\sum_{j \in J} x_{ij} - 1)) + \sum_{i \in I}(\lambda_i(\sum_{j \in J}(f_j(\vec{\mathbf{x}}) - T_{ij}^r) \cdot x_{ij} - \xi_i))$. Since any feasible solution to the dual problem is a lower bound to the primal problem [144], it is assumed that $\vec{\nu} = \vec{\mathbf{0}}$, thus the lower bound can be easily calculated by solving $g(\vec{\lambda}, \vec{\mathbf{0}}) = \min_{\vec{\mathbf{x}}, \vec{\xi}} \text{Card}(\vec{\xi}) + \sum_{i \in I}(\lambda_i(\sum_{j \in J}(f_j(\vec{\mathbf{x}}) - T_{ij}^r) \cdot x_{ij} - \xi_i))$ (the second term can be expressed as linear functions, see Section 6.3.2), where λ_i is known and $\lambda_i \geq 0$. In addition, the accuracy for solving the general cardinality minimization problem can be improved by some variants of ℓ_1-norm minimization, e.g., iterated weighted ℓ_1-norm heuristic [145]. The details are not elaborated here since they are beyond the interests of this thesis.

6.2.3 Pseudo-Code Summary

The proposed multiagent-based route guidance is summarized in Algorithm 3. Lines 1-2 initialize infrastructure agents and vehicle agents. In Lines 3-23, each infrastructure agent recursively assigns paths to vehicle agents who need route guidance at intersections, until they all

Algorithm 3: Multiagent-based Route Guidance

Input : $V = \{v_1, ..., v_Q\}$, a set of vehicle agents; $R = \{r_1, ..., r_G\}$, a set of infrastructure agents; $V^i = \varnothing$, vehicle agents that need guidance at r_i; TL_i, traffic lights associated with r_i; T_g, T_r, green light and red light duration; $flg = 0$, indicator of route guidance computation;

Each $r_i \in R$ turns on its associated traffic lights TL_i;
Each $v_j \in V$ determines destination d_j and deadline T_j, and computes an initial path P_j using Eq. (4.1);
while $|V| > 0$ **do**
    foreach $v_j \in V$ **do**
        if v_j reaches d_j **then**
            Deletes itself from V: $V = V - v_j$;
        else
            Travels along P_j;
            Updates deadline T_j;
    foreach $r_i \in R$ **do**
        foreach $tl_k \in TL_i$ **do**
            Runs according to T_g and T_r;
            if tl_k is in red color phase **then**
                r_i finds v_k facing tl_k: $V^i = V^i + v_k$;
                r_i collects d_k and T_k from v_k;
            if tl_k is at end of red color phase **then**
                $flg = 1$;
        if $flg == 1$ **then**
            foreach $v_j \in V^i$ **do**
                Computes relative deadline for v_j based on the latest T_j;
                Computes its new path $P_j^{'}$ via Eq. (6.3);
                Updates its route: $P_j = P_j^{'}$;
            Resets parameters: $V^i = \varnothing$, $flg = 0$;

reach destinations. Particularly, in Lines 4-9, each vehicle agent travels along a current route and updates its deadline if it has not reached destination. In Lines 11-17, during the red-color phase, each infrastructure agent recursively finds the set of vehicle agents who need route guidance and collects their intentions including deadlines and destinations. In Lines 18-22, upon the completion of the red-color phase, each infrastructure agent computes the optimal road links for all concerned vehicle agents to enter next based on route assignment in Eq. (6.3), and accordingly updates theirs routes.

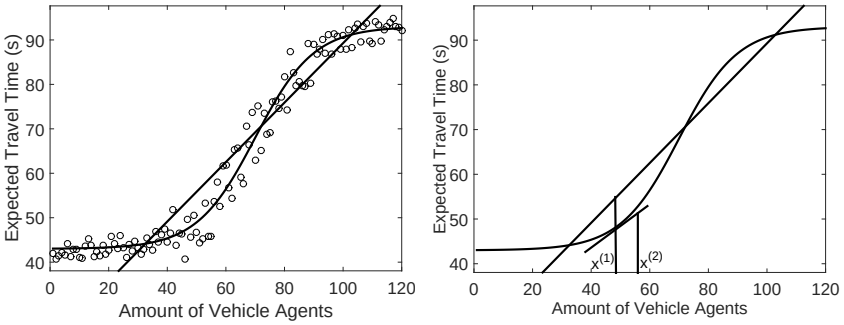
6.3 Further Improvements

In this Section, the proposed route guidance approach is further improved, in the following aspects: (1) predicted travel time on assigned road links, through iterative linearization for more accurate prediction function; (2) computational efficiency of the proposed approach, by introducing new variables and additional linear constraints; (3) real-time traffic information acquirement, by allowing communications between infrastructure agents.

6.3.1 Refinement of Predicted Travel Time

In view of analysis in Section 6.2, it is beneficial to assume that $f_j(\vec{\mathbf{x}})$ in Eq. (6.2) is linear, because route assignment in Eq. (6.3) can be formulated as a canonical optimization problem, i.e., MIQP, which enables tractable computation. However, by visualizing the statistical relationship between amount of vehicle agents and expected travel time on some road links (i.e., Fig 6.2.(a)), it is obvious that non-linear function fits the relationship better than that of linear function. Therefore, to guarantee both tractable computation and accurate route assignment, it adopts the non-linear function to predict the travel time on assigned road link, but solve the optimization problem in Eq. (6.3) by iterative linearization.

To be more specific, it takes the two functions in Fig 6.2 (b) as an example, and denote the linear and non-linear functions as $\mathcal{F}_j(\vec{\mathbf{x}})$ and $\mathbb{F}_j(\vec{\mathbf{x}})$ respectively. In the first iteration, it uses



6.2.a Two Fitted Functions 6.2.b Iterative Linearization

Figure 6.2: Illustration of Refinement of Predicted Travel Time Function.

$\mathcal{F}_j(\vec{\mathbf{x}})$ to replace $f_j(\vec{\mathbf{x}})$, and it gets a solution, i.e., $\vec{\mathbf{x}}^{(1)}$ to Eq. (6.3). In the next iteration, it uses the first order Taylor expansion of $\mathbb{F}_j(\vec{\mathbf{x}})$ at $\vec{\mathbf{x}}^{(1)}$ to replace $f_j(\vec{\mathbf{x}})$, which is expressed as:

$$f_j^{(k)}(\vec{\mathbf{x}}) = \mathbb{F}_j(\vec{\mathbf{x}}^{(k-1)}) + \nabla\mathbb{F}_j(\vec{\mathbf{x}}^{(k-1)})(\vec{\mathbf{x}} - \vec{\mathbf{x}}^{(k-1)}), \tag{6.4}$$

where $\nabla\mathbb{F}_j(\vec{\mathbf{x}}^{(k-1)})$ refers to the first order derivative of $\mathbb{F}_j(\vec{\mathbf{x}})$ at $\vec{\mathbf{x}}^{(k-1)}$, and k is the iteration number (k is equal to 2 in this context). Consequently, an optimal solution to Eq. (6.3) can be obtained, i.e., $\vec{\mathbf{x}}^{(2)}$. A better solution around $\vec{\mathbf{x}}^{(2)}$ on $\mathbb{F}_j(\vec{\mathbf{x}})$ is continued to be searched, until the objective function value in Eq. (6.3) does not significantly decrease. To guarantee that the algorithm locally searches for the optimum on the linearized straight line in each iteration, it limits the searching area by adding a constraint. Therefore, the optimization problem in the k-th iteration is formulated as:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{Q} \xi_i \left| \begin{array}{l} \sum_{j\in J}(f_j^{(k)}(\vec{\mathbf{x}}) - T_{ij}^r)\cdot x_{ij} \le \xi_i, \forall i \in I; \\ \sum_{j\in J} x_{ij} = 1, \forall i \in I; \xi_i \ge 0; x_{ij} \in \{0,1\}; \\ -\epsilon_2 \le \sum_{i\in I} x_{ij} - \sum_{i\in I} x_{ij}^{(k-1)} \le \epsilon_2, \forall j \in J, \end{array} \right. \tag{6.5}$$

where ϵ_2 is a positive integer, denoting local search range.

6.3.2 Improvement on Computational Efficiency

Route assignment in Eq. (6.3) is a MIQP problem mainly due to $(f_j(\vec{\mathbf{x}}) - T_{ij}^r)\cdot x_{ij}$ in the first constraint. After unfolding, quadratic part comes from the term $x_{kj}\cdot x_{ij}$ ($k \in I$, and x_{kj} is

a component of $\vec{\mathbf{x}}_j$). Since both $x_{kj}, x_{ij} \in \{0,1\}$, $x_{kj} \cdot x_{ij}$ can be replaced by x_{ij} if $k = i$. Therefore, the term $x_{kj} \cdot x_{ij}$ is quadratic only if $k \neq i$. However, $x_{kj} \cdot x_{ij}$ ($k \neq i$) can also be replaced by a binary variable with two additional linear constraints.

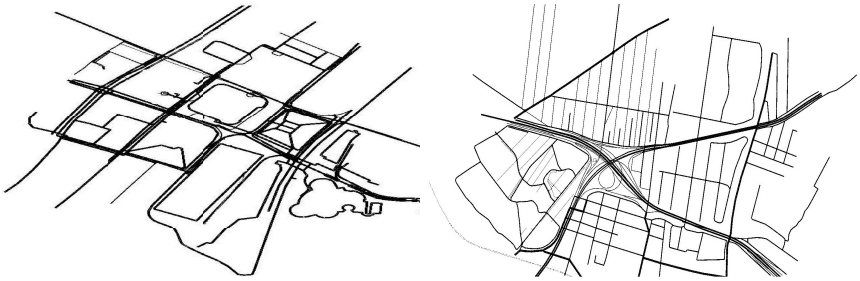
There are four correct permutations for vector (x_{kj}, x_{ij}, $x_{kj} \cdot x_{ij}$), i.e., (0,0,0), (0,1,0), (1,0,0) and (1,1,1). And a new variable $y_{kij} \in \{0,1\}$ is introduced to replace $x_{kj} \cdot x_{ij}$ ($k \neq i$), where eight permutations for vector (x_{kj}, x_{ij}, y_{kij}) exist. Therefore two linear cuts are added, i.e., $x_{kj} + x_{ij} + y_{kij} \leq 1$ and $-x_{kj} - x_{ij} + 2y_{kij} \leq 0$, to filter out the four faulty permutations [146]. Then it reformulates the MIQP problem in Eq. (6.3) as follows:

$$\min_{\vec{\mathbf{x}}} \sum_{i=1}^{Q} \xi_i \left| \begin{array}{l} \sum_{j \in J} (g_j(\vec{\mathbf{z}}) + (\gamma_j - T_{ij}^r) x_{ij}) \leq \xi_i, \forall i \in I; \\ -x_{kj} - x_{ij} + 2y_{kij} \leq 0, ..., \\ x_{kj} + x_{ij} + y_{kij} \leq 1, \forall i, k \in I, k < i, \forall j \in J; \\ \sum_{j \in J} x_{ij} = 1, \forall i \in I; \xi_i \geq 0; x_{ij} \in \{0,1\}, \end{array} \right. \tag{6.6}$$

where $g_j(\vec{\mathbf{z}}) = c_j \sum_{i \in I} z_{ij}$; size of $\vec{\mathbf{z}}$ is same with that of $\vec{\mathbf{x}}$; z_{kj} is equal to x_{ij} if $k = i$, and y_{kij} if $k \neq i$; y_{kij} is equal to y_{ikj} in this scenario. Thus, Eq. (6.6) is reduced to a mixed integer linear programming (MILP) problem, which can be solved much more efficiently than MIQP of similar scale [142].

6.3.3 Performance Improvement via Communication

In the proposed approach, the historical expected travel time is used to evaluate the remaining path from the assigned road link to destination. Since the infrastructure agent is always located at an intersection, it can obtain real-time traffic conditions (i.e., travel time) on directly connected road links. It is thus reasonable for an infrastructure agent to communicate with neighboring infrastructure agents to obtain real-time traffic conditions further away. The expectation is that real-time traffic condition can better evaluate a route than the historical traffic condition. It uses E (i.e., $E \in \mathbb{Z}_0^+$) to denote the number of communication hops, and there is no communication if $E = 0$. In Fig. 6.1, if $E = 1$, r_1 only communicates with its neighbors,



6.3.a Singapore 6.3.b New York

Figure 6.3: Two Testing Road Networks

e.g., r_2 and r_4. Thus it can obtain real-time traffic conditions on p_4, p_5, p_{12}, p_{10} and p_8, which can be used to evaluate the paths from r_2 and r_4 to destinations when r_1 performs route assignment for v_1, v_2 and v_3. As E increases to 2, r_1 is able to communicate with infrastructure agents one more hop away, e.g., r_3, thus r_1 can obtain real-time traffic conditions on p_6, p_7 and r_{14} as well. However, as the number of communication hops becomes larger, additional communication and storage costs also incur. The dynamics of traffic may also cause real-time traffic information to be outdated by the time vehicle agents reach the intersection, if the location is far away.

6.4 Experimentation

In this section, experiments in various settings are conducted to extensively compare the proposed route guidance approach with existing methods, showing its advantages of increasing the chances of reaching destination before deadline for all vehicles. In addition, it verifies all the proposed improvements through experimentation, and investigate its extended flexibility of leveraging arrival on time and total travel time.

6.4.1 Road Networks and Parameter Settings

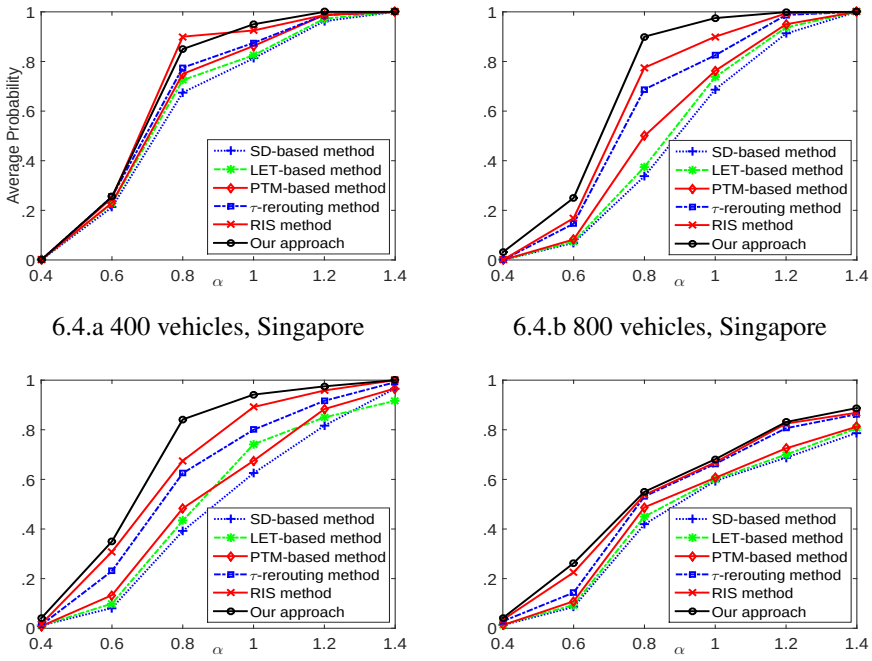
All experiments are conducted on the urban mobility simulator, SUMO [147]. The two testing road networks are parts of two very dense cities, Singapore and New York respectively. Each

road has 2 lanes, and their maps are given in Fig. 6.3, with the following properties summarized by SUMO: (1) network areas are 65,300m^2 and 218,000m^2; (2) numbers of road links are 507 and 1,121; (3) numbers of intersections are 98 and 352.

The configurations of vehicles are as follows: length is 5m; minimal gap is 2.5m; car following model is Krauss [147]; origins and destinations are randomly generated; traffic light duration: $T_g = T_r = 20s$; vehicles will park and not occupy road resources when reaching destinations. In addition, the positive deadline parameter α is introduced to denote different levels of deadlines. Specifically, once O-D are determined, an expected travel time T_e can be derived based on historical traffic data. Thus, $T = \alpha \cdot T_e$ implies a tight deadline if $\alpha < 1$, and a loose deadline if $\alpha > 1$. Once origin and destination of a vehicle are determined, an expected travel time T_e can be derived based on historical traffic data, which can help to describe different levels of deadlines. Moreover, the proposed approach needs historical expected travel time to evaluate some parts of a route. Therefore, before testing the proposed approach, it first randomly runs the simulation for 250 times to get an expected travel time of each road link, where vehicles simply travel along the shortest distance routes. Additionally, it also uses SVR (i.e., support vector regression) [148] to learn the linear and non-linear functions of predicted travel time through those random simulations, which are described in Section 6.3.1. Particularly, all experiments in this Chapter are conducted on an ordinary PC with Intel Core i7-3540M processor and 8.00 GB RAM.

6.4.2 Comparative Performance

The proposed algorithm are compared with five different route guidance methods: (1) SD (i.e., shortest distance) based method, which pre-computes a path of shortest distance; (2) LET-based method, which pre-computes a path of least expected travel time based on historical traffic conditions; (3) PTM-based method, which pre-computes a path of PT model by Eq. (4.1) in Chapter 4; (4) RIS (i.e., route information sharing) method, which constantly computes LET paths for vehicles at each intersection by cooperatively exploring their latest intentions of routes [47]; (5) τ-rerouting method, claimed to be the best-performing vehicle rerouting



6.4.a 400 vehicles, Singapore

6.4.b 800 vehicles, Singapore

6.4.c 1,200 vehicles, Singapore

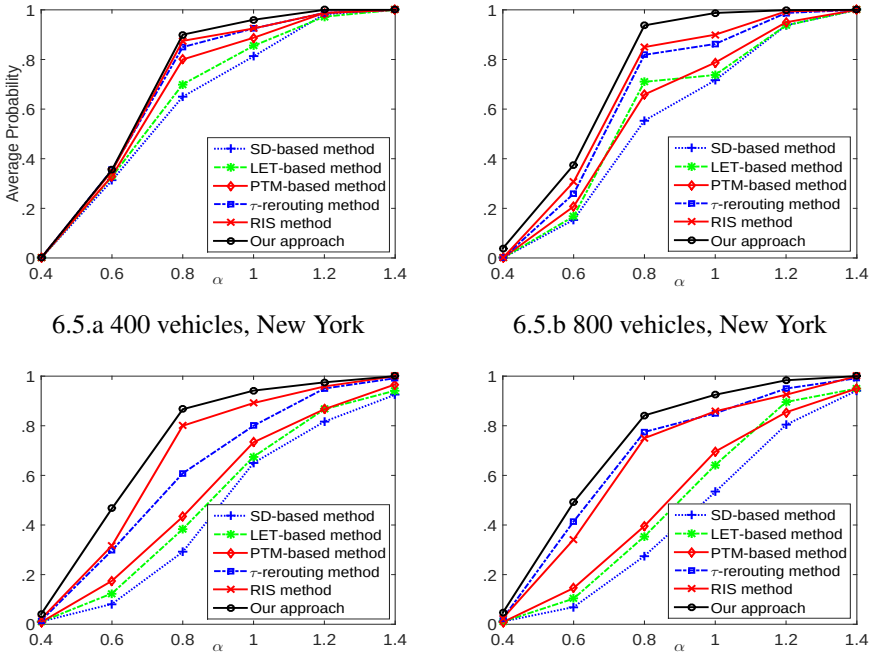
6.4.d 1,600 vehicles, Singapore

Figure 6.4: Different Types of Deadlines on Singapore Map.

strategy [42]. Note that the first three methods pre-compute route guidance before vehicle departure, while the last two and the proposed approach adaptively provide route guidance for vehicles en-route. Although O-D pairs are randomly generated, for each specified O-D pair, it respectively adopts the six methods to provide route guidance.

6.4.2.1 Different Levels of Deadlines

This experiment varies different levels of deadlines α: 0.4, 0.6, 0.8, 1.0, 1.2 and 1.4, with different numbers of vehicles: 400, 800, 1,200 and 1,600 on both networks. The simulation are run for 500 times under each setting, the probability of arrival on time for each vehicle is recorded, and the average is plotted in Fig. 6.4 and Fig. 6.5 respectively. It can be observed that on both networks, the average probabilities always increase with α for all six methods. It is natural because a vehicle with a very loose deadline has higher chance to arrive on time even if it does not follow any smart route guidance. Generally, the three pre-computation methods are



6.5.a 400 vehicles, New York

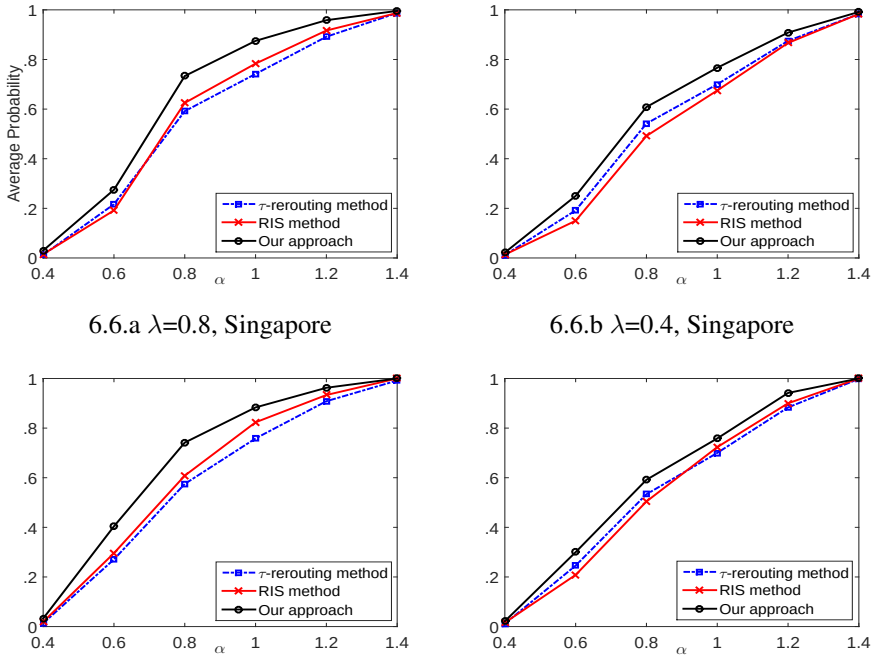
6.5.b 800 vehicles, New York

6.5.c 1,200 vehicles, New York

6.5.d 1,600 vehicles, New York

Figure 6.5: Different Types of Deadlines on New York Map.

inferior to the three adaptive methods, because the optimality of pre-computed paths may not hold, especially in highly dynamic traffic, e.g., New York network with 1,600 vehicles. However, the inferiority is not obvious in extremely sparse or saturated traffic, such as Fig. 6.4 (a), (d) and Fig. 6.5 (a). In sparse traffic, vehicles rarely influence each other, and shortest distance path is sufficiently satisfactory. In over-saturated traffic, vehicles almost cannot proceed even if they receive adaptive guidance. Among the three pre-computation methods, PTM-based method achieves the highest overall performance because it takes deadline into account, although in an independent manner. As for the three adaptive methods, the proposed approach is always better than the other two in terms of overall probabilities of arriving on time, especially in Fig. 6.4 (b), (c), Fig. 6.5 (b), (c) and (d), where the traffic densities are moderate. In most cases, RIS method is better than τ-rerouting method, because it is centralized, where a global server constantly predicts the LET path for each individual vehicle, based on latest intentions of routes. This superiority does not hold for New York network with 1,600 vehicles, because the traffic density is comparatively high, and τ-rerouting method is especially effective where congestion is likely to occur. However, both methods do not care about whether vehicles would



6.6.a λ=0.8, Singapore

6.6.b λ=0.4, Singapore

6.6.c λ=0.8, New York

6.6.d λ=0.4, New York

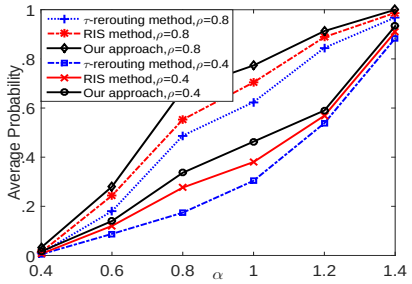
Figure 6.6: Different Penetration Rates.

be late regarding their preferred deadlines. On the other hand, the proposed approach cooperatively explores the deadlines of other vehicles, and recursively provides guidance by solving an optimization problem, which aims to guarantee arrival at destination before deadline for all concerned vehicles, thus achieving the best overall performance.

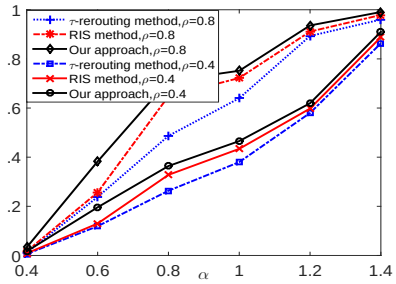
6.4.2.2 Different Penetration Rates

The three adaptive approaches[3] are tested with different penetration rates λ defined as the percentage of vehicles sharing their intentions. It takes both networks with 1,200 vehicles as study cases, and adopts two penetration rates, λ=0.8 and 0.4. From Fig. 6.6, it is noticed that average probabilities for the three methods decrease as λ becomes smaller. It is natural since route guidance in them all rely on intentions. Particularly, RIS method is centralized, and missing intentions of routes may globally influence route guidance of others, so τ-rerouting method

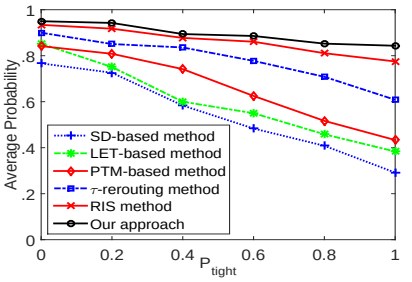
[3]The three pre-computation methods do not involve intention sharing, and it makes no sense to test penetration rates for them.



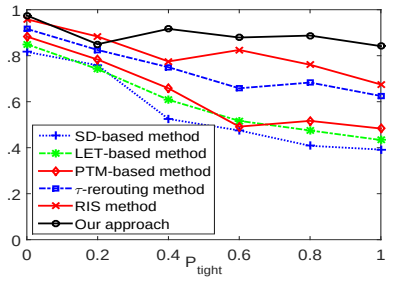
6.7.a 1,200 vehicles, Singapore



6.7.b 1,200 vehicles, New York



6.7.c 1,200 vehicles, Singapore



6.7.d 1,200 vehicles, New York

Figure 6.7: Different Compliance Rates and Different Percentages of Tight Deadlines.

and the proposed approach achieve better performance regarding λ=0.4, especially for tight deadlines on both networks, i.e., from $\alpha = 0.6$ to 0.8. Although τ-rerouting method is decentralized, missing intentions may make infrastructure agents unable to report congestion timely, which causes more vehicles to miss their deadlines. On the other hand, in the proposed approach, predicted travel time on assigned road link relies on both vehicle amount and road link length. Missing intentions only partially influences the proposed approach. Moreover, the proposed approach always takes deadline into account, thus it achieves best overall performance for different penetration rates.

6.4.2.3 Different Compliance Rates

The three adaptive approaches are further tested with different compliance rates ρ defined as the percentage of vehicles following the route guidance by infrastructure agent. It takes both networks with 1,200 vehicles as study cases, and adopts two compliance rates, ρ=0.8 and 0.4,

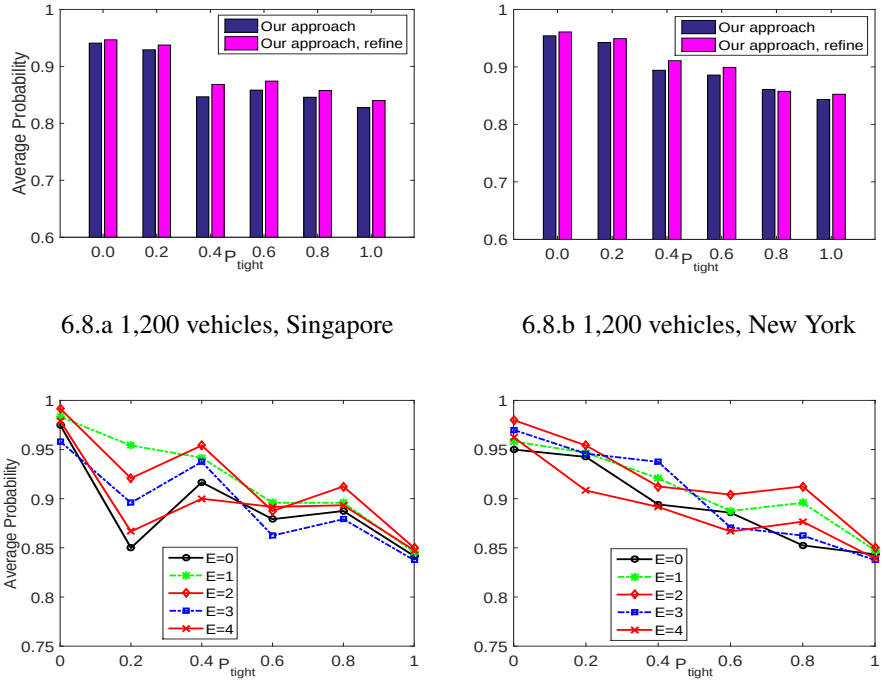
the results of which are plotted in Fig. 6.7 (a) and (b) respectively. Note that the three adaptive approaches in Fig. 6.4 (c) and Fig. 6.5 (c) refer to ρ=1. Combining those figures together, it is noticed that average probabilities for the three methods on both networks decrease as ρ becomes smaller. However, the proposed approach achieves highest overall performance for each setting because it takes arrival on time into account, although some of vehicles may not follow the guidance. It is also observed that, τ-rerouting method in Fig. 6.7 (a) has lower probability, especially for α=0.8 and ρ=0.4. It happens because vehicle density is comparatively higher on Singapore network, where congestion is likely to happen. If most of vehicles do not follow the rerouting, they may have to stay in congestion for a longer time. Generally, the proposed approach will achieve higher probability of arrival on time as ρ increases, which provides convincing incentives for vehicles to follow the route guidance.

6.4.2.4 Different Percentages of Tight Deadlines

It takes both networks with 1,200 vehicles as study cases to further test the proposed approach against different percentages of tight deadlines. In this situation, α is set as 0.8 for tight deadline, and 1.2 for loose deadline. Their percentages are P_{tight} and 1-P_{tight}. In Fig. 6.7 (c) and (d), as P_{tight} increases, the average probabilities for the three pre-computation methods drop more quickly on Singapore network because its traffic density is comparatively high, where vehicles always influence each other. There is only slight decrease for the proposed approach on both networks, which is better than the other two adaptive methods. Although RIS method on Singapore network is competitive to the proposed approach, it is highlighted that RIS method is centralized, becoming prohibitively time-consuming as network size and vehicle number scale up.

6.4.3 Improved Performance

Experiments are also conducted to verify the improvements on travel time prediction, computation efficiency and real-time traffic condition acquirement proposed in Section 6.3.



6.8.a 1,200 vehicles, Singapore

6.8.b 1,200 vehicles, New York

6.8.c 1,200 vehicles, Singapore

6.8.d 1,200 vehicles, New York

Figure 6.8: Refinement of Prediction and Improvement via Communication.

6.4.3.1 Refinement of Travel Time Prediction

Iteratively linearizing a more accurate function is proposed to refine the predicted travel time on assigned road link. It adopts Singapore network and New York network with 1,200 vehicles as study cases, to investigate this improvement to route assignment in Eq. (6.3), and all the results are plotted in Fig. 6.8 (a) and (b) respectively.

From Fig. 6.8 (a) and (b) it is observed that, refinement of travel time prediction improves the average probability of arrival on time for both networks in most cases (except for $P_{tight} = 0.8$ on New York network. Nevertheless, the difference is not significant, which is acceptable). This advantage comes from the fact that the proposed method approximates a non-linear function, which is more accurate than the previous linear prediction. It is also noticed that, compared with other values, the improvement for $P_{tight} = 0$ and $P_{tight} = 0.2$ are not that obvious. It happens because most vehicles have loose deadlines for $P_{tight} = 0$ and $P_{tight} = 0.2$, and they still have higher chances of arrival on time although the route assignment is performed based on

the inaccurate linear prediction of travel time. However, as P_{tight} increases, this improvement becomes more obvious.

6.4.3.2 Improvement of Computation Efficiency

Original route assignment is formulated as an MIQP problem in Eq. (6.3), and it is reformulated as an MILP problem in Eq. (6.6). To show the efficiency improvement, it uses Pyomo (www.pyomo.org) to respectively solve the two problems regarding the same route assignment at each intersection, and records the average computation time for both networks in Table 6.1.

Table 6.1: Average Computation Time (s)

	Singapore				**New York**			
	400	800	1,200	1,600	400	800	1,200	1,600
MIQP	2.31	3.19	7.42	11.61	1.16	1.96	3.29	6.98
MILP	0.38	0.58	0.95	1.19	0.21	0.34	0.62	0.72

From Table 6.1 it is observed that as vehicle number increases, the average computation time becomes longer for both problems. This happens because more vehicles are likely to request for route guidance at an intersection if traffic density is larger, thus the scales of the two optimization problems both increase, and longer computation time is needed. That also explains why the computation time on Singapore network is longer than that of New York network. However, for both, MILP problem can be more efficiently solved than MIQP problem of similar scale, especially for Singapore network with 1,600 vehicles, which is around 10 times faster.

6.4.3.3 Improvement via Communication

To evaluate the benefits brought by communication, the proposed approach is tested against different communication hops (i.e., E). It again studies both networks with 1,200 vehicles for different percentages of tight deadlines. From Fig. 6.8 (c) and (d), it is found that, $E = 1, 2$

on Singapore network, and $E = 1, 2, 3$ on New York network always achieve higher overall probabilities than that of $E = 0$, this is reasonable in that, if $E > 0$, the proposed approach uses real-time traffic conditions to evaluate the first E road link(s) of the path from assigned road link to destination. If $E = 0$, it only uses historical traffic conditions. As E increases to, e.g., 3 and 4 on Singapore network, and 4 on New York network, they do not achieve dominant performances over that of $E = 0$, because the traffic is always dynamic, and knowing real-time traffic conditions far away may not yield desirable route guidance [66]. Moreover, large communication hop also incurs additional cost to dynamically obtain and store traffic information. Therefore, $E = 1$ and $E = 2$ are sufficient to achieve satisfactory route guidance in the proposed approach.

6.5 Summary

In this chapter, a decentralized multiagent-based route guidance approach has been proposed to solve the arriving on time problem for multiple cooperative vehicles. The route guidance is formulated as a route assignment problem at each road intersection by leveraging intentions of the vehicles, which aims to increase the chances of arriving on time. Additionally, the proposed route guidance approach has been improved in the aspects of travel time prediction, computational efficiency and real-time traffic condition acquirement respectively. Experimental results confirm its superior performance over existing methods. As the chances of vehicles' arriving on time is increased, drivers' satisfaction gets improved. Thus, it also reduces the accident rate due to drivers' frustration and impatience, which is an important mission of the intelligent transportation and sustainable urban development.

Chapter 7

Conclusions and Future Work

In this chapter, the proposed approaches to the stochastic shortest path (SSP) problems for a single vehicle and multiple cooperative vehicles in the whole thesis are first concluded. Regarding the former, the unrealistic assumptions have been circumvented by reformulating the problem using a data-driven approach. Then the computation efficiency and accuracy have been improved by the partial Lagrange multiplier method and the Q-learning method, respectively. Regarding the latter, the chances of arriving on time for all relevant vehicle agents have been increased by a multiagent-based approach. Besides, the future research direction for the two problems are pointed out. Regarding the former, better solutions with respect to both computation efficiency and accuracy should be investigated, and more practical factors should be considered as well. Regarding the latter, incentive schemes should be devised, and more personalized route guidance should also be studied.

7.1 Conclusions

This section concludes the proposed studies to solve the arriving on time problem for a single vehicle and multiple cooperative vehicles, respectively. Regarding the former, a data-driven approach is developed to circumvent the unrealistic assumptions in the existing solutions, by reformulating the PT model based SSP problem as a cardinality minimization problem. This problem is solved by further relaxing it into an MILP problem. Then, the computation efficiency and accuracy of finding the real optimal path are improved, through the partial Lagrange multiplier method and Q-learning method, respectively. Regarding the latter, the criterion of arriving on time is incorporated into the multiagent-based route guidance, which increases the chances of reaching destination before deadline for all relevant vehicles. In particular, the multiagent-based approach is decentralized, and each infrastructure agent only provides local route guidance by formulating it as a route assignment problem.

More specifically, **Chapter 3** presents a data-driven approach regarding the arriving on time problem for a single independent vehicle, which addresses the issues of the three unrealistic assumptions. In particular, this approach reformulates the path finding as a cardinality minimization problem by directly employing travel time data of each road link, which aims to minimize the cardinality of being late on the optimal path. Then, this cardinality minimization is approximately solved via the ℓ_1-norm relaxation and its variants, which is further reformulated as an MILP problem. Thus, the arriving on time problem becomes solvable via using an existing MILP solver, i.e., the B&B method. Experimental results on the artificial and real road networks show that, compared with other methods in Chapter 3, the basic ℓ_1-norm algorithm is a satisfactory solution considering both accuracy and computation. However, there is still room for its computation efficiency to be improved.

Chapter 4 proposes a partial Lagrange multiplier method, to improve the computation efficiency of the MILP based arriving on time problem in Chapter 3. The proposed approach explores the property of total unimodularity that exists in the equality constraint of the MILP problem, which guarantees integer solutions by solving a corresponding LP problem if there

are no inequality constraints. Thus, the partial Lagrange multipliers are used to relax the existing inequality constraints by shifting them to the objective function. After that, the subgradient method is employed to address the reformulated problem, and only a LP problem is solved in each iteration, which significantly saves computation time. To show the desirable generalization of the proposed method, it is also applied to solve the SSPD model based SSP problem. The experimental results on the Beijing road network with real traffic data have shown that the proposed method can efficiently solve the time-dependent vehicle routing problem, and significantly save computation time compared with the method in Chapter 3. Besides, the implementation into a navigation system based on the Singapore road network further confirmed that the proposed method can be applied to solve the real world arriving on time problem. However, the ℓ_1-norm is still only an approximation to the cardinality minimization problem in Chapter 3.

Chapter 5 develops a practical Q-learning method, to improve the accuracy of finding the real optimal path. This Q-learning method aims to maximize the expected reward regarding whether the vehicle can arrive on time or not on a specified path. At the same time, the expected reward has the meaning of probability of arriving on time from a current location to destination. To address the issues of continuous deadlines and large-scale road networks, a practical value function approximation method is developed based on the dynamic neural networks, to directly learn the optimal Q-value, which represents the probability of arriving on time. Experimental results on one artificial and three real road networks demonstrate that the Q-learning method is practical for real world application, which improves the accuracy and computation efficiency compared with the methods in Chapter 3 and Chapter 4, and some other baselines. To summarize, the proposed Q-learning method offers several important practical features: 1) It considers travel time, risk and deadlines simultaneously when computing an optimal path; 2) It provides probability of arriving on time from any origin(s) to any destination(s) once the value function is learned; 3) It can flexibly provide time-dependent path recommendation and avoid frequent training even if travel time data changes.

Chapter 6 proposes a decentralized multiagent-based method, which solves the arriving on time problem for multiple cooperative vehicles. In this approach, two types of agents are

involved, i.e., vehicle agents and infrastructure agents. Particularly, each vehicle agent follows the route guidance by the local infrastructure agent, and the infrastructure agent formulates the local route guidance as a route assignment problem, which aims to increase the chances of arriving on time for all relevant vehicle agents. Moreover, the route assignment problem at each infrastructure agent is efficiently solved through existing solvers. Experimental results on part of the Singapore and New York road networks show that, in comparison with other baselines in Chapter 6, the proposed method can significantly improve the average probabilities of arriving on time for all vehicles agents in the system, especially for the case of high traffic densities.

7.2 Future Work

In view of the current studies and conclusions, the future research will be conducted in the following two aspects.

7.2.1 Further Study of Arriving on Time for a Single Vehicle

The proposed solutions to the arriving on time problem for a single vehicle, has successfully circumvented the three unrealistic assumptions, and improved the computation efficiency and accuracy as well. However, besides the on-field test using the real car in future, there are still some other practical issues that can be considered in this arriving on time problem.

- **Develop More Accurate Solution to the Cardinality Minimization.** In Chapter 3, the PT model based SSP problem is reformulated as a cardinality minimization problem to circumvent the strong assumptions. Then this problem is approximately solved using the ℓ_1-norm relaxation based algorithms, which are easy and simple to implement. However, other types of relaxation methods are also feasible, e.g., using a group of sigmoid functions [103] to approximate the cardinality. Although it needs more computation time regarding the same problem, higher accuracy usually can be achieved. Therefore, it is worth to investigate a balance between the accuracy and computation time regarding this

sigmoid function based method. At the same time, the delay might be unacceptably long if the user, who relies on the PT model, arrives at the destination after the deadline. And it is crucial to balance the probability of arriving on time and the worst case of delay.

- **Further Improve the Computation Efficiency.** In Chapter 4, a partial Lagrange multiplier method is proposed to improve the computation efficiency, which explores the advantage of the total unimodularity property of the incidence matrix. In order to get more desirable results, the incidence matrix of the whole road network is utilized. However, in most cases, the optimal path lies in a small area containing the O-D pair, and the incidence matrix of the whole road network is not necessary. At the same time, a smaller incidence matrix implies an optimization problem of smaller size, which accordingly leads to less computation. Therefore, it is beneficial to investigate a strategy to extract a suitable sub-network, the incidence matrix of which would be used to compute the optimal path. In view of this, a simple strategy could be drawing a suitable circle or rectangle that contains the O-D pair. Besides, in some cases, only several meaningful routes (e.g., different expressways) may exist, where the maximum probability of arriving on time may be obtained by simply enumerating all the meaningful routes. Although recognizing those meaningful routes may incur additional computational cost, it is worth to investigate whether this alternative approach can save the total computation in comparison with the original solution.

- **Consider More Practical Factors.** In Chapter 5 the location and deadline are adopted as a state in the Q-learning method. To be more practical, it can also include other factors, such as weather (e.g., sunny, rainy, windy and snowy) and width of the road (e.g., narrow, medium and wide), both of which are likely to influence the traffic on roads. Besides, regarding the arriving on time problem for a single vehicle, several other practical issues can also be incorporated: 1) transition time at intersections, such as waiting time for the desired traffic lights; 2) passing through fixed locations before destination, which is suitable for parents to drop children to the school before going to work; 3) reducing turnings at intersections, which satisfies the preferences of most drivers. All of the three practical issues can be naturally incorporated into the optimization method, which would nourish the solutions to the arriving on time problem.

7.2.2 Further Study of Arriving on Time for Multiple Cooperative Vehicles

In the multiagent-based approach, to guarantee most of vehicle agents arriving on time, it formulates the route guidance as a route assignment problem. In this problem, some vehicle agents with loose deadlines might be assigned to detoured routes to give ways to those with tight deadlines. Consequently, an equity issue is involved, because those "sacrifices" might be unfair to some of them, which is worth to be investigated in future. At the same time, some incentive schemes should be devised, to encourage vehicle agents to follow the provided route guidance. In addition, it is also worth to explicitly consider the potential route assignment at neighboring (or further) intersections when a local infrastructure agent performing the route assignment. On the other hand, in the current method, only arriving on time is considered, and it is desirable to also incorporate the total travel time, which is important to the environment impact. Therefore, in future, more personalized route guidance will be considered by integrating both arriving on time and total travel time into the objective function, the weights of which would be decided by drivers themselves.

Publications

Conference

(i) ***Zhiguang Cao***, Hongliang Guo, Jie Zhang, Frans Oliehoek and Ulrich Fastenrath. "Maximizing the Probability of Arriving on Time: A Practical Q-Learning Method". 31th AAAI Conference on Artificial Intelligence (AAAI), 2017. (In Press)

(ii) ***Zhiguang Cao***, Hongliang Guo, Jie Zhang and Ulrich Fastenrath. "Multiagent-based Route Guidance for Increasing the Chance of Arrival on Time". 30th AAAI Conference on Artificial Intelligence (AAAI), pp. 3814-3820, 2016.

(iii) ***Zhiguang Cao***, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. "A Data-Driven Method for Stochastic Shortest Path Problem". 17th IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 1045-1052, 2014.

(iv) ***Zhiguang Cao***, Qin Li, Hoon Wei Lim and Jie Zhang. "A Multi-hop Reputation Announcement Scheme for VANETs". IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), pp. 238-243, 2014.

(v) Hongliang Guo, ***Zhiguang Cao***, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. "Routing Multiple Cars in Large Scale Networks: Minimizing Road Network Breakdown Probability". 17th IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 2180-2187, 2014.

Journal

(i) Hongliang Guo, ***Zhiguang Cao***, Jie Zhang, Dusit Niyato, Madhavan Seshadri and Ulrich Fastenrath. Routing Multiple Vehicles Cooperatively: Minimizing Road Network Breakdown Probability. IEEE Transactions on Emerging Topics in Computational Intelligence (TETCI), 2017. (In Press)

(ii) ***Zhiguang Cao***, Siwei Jiang, Jie Zhang, and Hongliang Guo. "A Unified Framework for Vehicle Rerouting and Traffic Light Control to Reduce Traffic Congestion". IEEE Transactions on Intelligent Transportation Systems (T-ITS), 2017. (In Press)

(iii) ***Zhiguang Cao***, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. "Finding the Shortest Path in Stochastic Vehicle Routing: A Cardinality Minimization Approach". IEEE Transactions on Intelligent Transportation Systems (T-ITS), vol. 17, no. 6, pp. 1688-1702, 2016.

(iv) ***Zhiguang Cao***, Hongliang Guo, Jie Zhang, Dusit Niyato and Ulrich Fastenrath. "Improving the Efficiency of Stochastic Vehicle Routing: A Partial Lagrange Multiplier Method". IEEE Transactions on Vehicular Technology (TVT), vol. 65, no. 6, pp. 3993-4005, 2016.

References

[1] Dave Versical, *Automotive News China,Automotive News Europe, Automotive News USA*, 2014. http://www.autonews.com/.

[2] LeBeau Phil, *1.7 Billion Cars on the Road by 2035*, 2012. http://www.cnbc.com/id/49796736.

[3] H. Guo, Z. Cao, J. Zhang, D. Niyato, and U. Fastenrath, "Routing multiple cars in large scale networks: Minimizing road network breakdown probability," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2180–2187, 2014.

[4] D. Schrank, B. Eisele, and T. Lomax, "Ttis 2012 urban mobility report," *Texas A&M Transportation Institute. The Texas A&M University System*, 2012.

[5] The Economist, "The Cost of Traffic Jams," November 2014.

[6] Yahoo News, *Where are the busiest roads in Singapore?*, 2012. https://sg.news.yahoo.com/blogs/fit-to-post-autos/where-busiest-roads-singapore-005656369.html.

[7] G. Isaac and T. Lange, "Split routing as a part of the urban navigation," in *Proceedings of 19th Intelligent Transportation Systems World Congress*, 2012.

[8] Y. Fan and Y. Nie, "Optimal routing for maximizing the travel time reliability," *Networks and Spatial Economics*, vol. 6, no. 3-4, pp. 333–344, 2006.

[9] E. D. Miller-Hooks and H. S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks," *Transportation Science*, vol. 34, no. 2, pp. 198–215, 2000.

[10] B. C. Dean, "Algorithms for minimum-cost paths in time-dependent networks with waiting policies," *Networks*, vol. 44, no. 1, pp. 41–46, 2004.

[11] S. T. Waller and A. K. Ziliaskopoulos, "On the online shortest path problem with limited arc cost dependencies," *Networks*, vol. 40, no. 4, pp. 216–227, 2002.

[12] Y. Ohtsubo, "Stochastic shortest path problems with associative accumulative criteria," *Applied Mathematics and Computation*, vol. 198, no. 1, pp. 198–208, 2008.

[13] Y. Pan, L. Sun, and M. Ge, "Finding reliable shortest path in stochastic time-dependent network," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 451–460, 2013.

[14] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "A data-driven method for stochastic shortest path problem," in *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1045–1052, 2014.

[15] Y. Fan, R. Kalaba, and J. Moore II, "Arriving on time," *Journal of Optimization Theory and Applications*, vol. 127, no. 3, pp. 497–513, 2005.

[16] Z. Cao, H. Guo, J. Zhang, and D. Niyato, "Improving the efficiency of stochastic vehicle routing: A partial lagrange multiplier method," *IEEE Transactions on Vehicular Technology(TVT)*, vol. 65, no. 6, pp. 3993–4005, 2016.

[17] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: a cardinality minimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1688–1702, 2016.

[18] S. Kim, M. E. Lewis, and C. C. White, "State space reduction for nonstationary stochastic shortest path problems with real-time traffic information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 273–284, 2005.

[19] D. Sudholt and C. Thyssen, "A simple ant colony optimizer for stochastic shortest path problems," *Algorithmica*, vol. 64, no. 4, pp. 643–672, 2012.

[20] S. Zibaei, A. Hafezalkotob, and S. S. Ghashami, "Cooperative vehicle routing problem: an opportunity for cost saving," *Journal of Industrial Engineering International*, pp. 1–16, 2016.

[21] E. Nikolova and N. E. Stier-Moses, *Stochastic Selfish Routing*, pp. 314–325. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[22] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 713–724, 2013.

[23] E. D. Tate, J. W. Grizzle, and H. Peng, "Shortest path stochastic control for hybrid electric vehicles," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 14, pp. 1409–1429, 2008.

[24] B. Y. Chen, W. H. Lam, A. Sumalee, Q. Li, and M. L. Tam, "Reliable shortest path problems in stochastic time-dependent networks," *Journal of Intelligent Transportation Systems*, vol. 18, no. 2, pp. 177–189, 2014.

[25] R. W. Hall, "The fastest path through a network with random time-dependent travel times," *Transportation science*, vol. 20, no. 3, pp. 182–188, 1986.

[26] E. Nikolova, "Approximation algorithms for reliable stochastic combinatorial optimization," in *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pp. 338–351, Springer, 2010.

[27] I. Murthy and S. Sarkar, "Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function," *European Journal of Operational Research*, vol. 103, no. 1, pp. 209–229, 1997.

[28] C. E. Sigal, A. A. B. Pritsker, and J. J. Solberg, "The stochastic shortest route problem," *Operations Research*, vol. 28, no. 5, pp. 1122–1129, 1980.

[29] S. Samaranayake, S. Blandin, and A. Bayen, "A tractable class of algorithms for reliable routing in stochastic networks," *Transportation Research Part C: Emerging Technologies*, vol. 20, no. 1, pp. 199–217, 2012.

[30] M. Hua and J. Pei, "Probabilistic path queries in road networks: traffic uncertainty aware path selection," in *Proceedings of the 13th International Conference on Extending Database Technology*, pp. 347–358, ACM, 2010.

[31] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher, "Stochastic shortest paths via quasi-convex maximization," in *Proceedings of European Symposium of Algorithms*, pp. 552–563, 2006.

[32] Y. M. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transportation Research Part B: Methodological*, vol. 43, no. 6, pp. 597–613, 2009.

[33] S. Lim and D. Rus, "Stochastic motion planning with path constraints and application to optimal agent, resource, and route planning," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 4814–4821, 2012.

[34] S. Lim, C. Sommer, E. Nikolova, and D. Rus, "Practical route planning under delay uncertainty: Stochastic shortest path queries," in *Robotics: Science and Systems*, vol. 8, pp. 249–256, 2013.

[35] J. France, A. Ghorbani, *et al.*, "A multiagent system for optimizing urban traffic," in *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT)*, pp. 411–414, 2003.

[36] C. Wilt and A. Botea, "Spatially distributed multiagent path planning," in *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[37] S. S. Chen, X. Zhao, and Y. Sheng, "Research on dynamic route guidance for an emergency vehicle considering the intersection delay," in *Applied Mechanics and Materials*, vol. 641, pp. 848–852, 2014.

[38] M. Zolfpour-Arokhlo, A. Selamat, S. Z. M. Hashim, and H. Afkhami, "Modeling of route planning system based on q value-based dynamic programming with multi-agent reinforcement learning algorithms," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 163–177, 2014.

[39] M. Lujak, S. Giordani, and S. Ossowski, "Route guidance: Bridging system and user optimization in traffic assignment," *Neurocomputing*, vol. 151, pp. 449–460, 2015.

[40] S. F. Smith, A. Gallagher, T. L. Zimmerman, L. Barbulescu, and Z. B. Rubinstein, "Multi-agent management of joint schedules.," in *AAAI Spring Symposium: Distributed Plan and Schedule Management*, pp. 128–135, 2006.

[41] J. Oh and S. F. Smith, "A few good agents: multi-agent social learning," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pp. 339–346, 2008.

[42] S. Jiang, J. Zhang, and Y.-S. Ong, "A pheromone-based traffic management model for vehicle re-routing and traffic light control," in *Proceedings of international conference on Autonomous agents and multi-agent systems (AAMAS)*, pp. 1479–1480, 2014.

[43] A. L. Bazzan and F. Klügl, "A review on agent-based technology for traffic and transportation," *The Knowledge Engineering Review*, vol. 29, no. 03, pp. 375–403, 2014.

[44] A. T. Castañeda and E. G. Guerrero, "Mitc: An intention-based model for cooperative resolution of traffic conflicts," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 145–153, 2014.

[45] B. Burmeister, A. Haddadi, and G. Matylis, "Application of multi-agent systems in traffic and transportation," in *Software Engineering. IEE Proceedings-[see also Software, IEE Proceedings]*, vol. 144, pp. 51–60, IET, 1997.

[46] B. Chen and H. H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 11, no. 2, pp. 485–497, 2010.

[47] T. Yamashita, K. Izumi, K. Kurumatani, and H. Nakashima, "Smooth traffic flow with a cooperative car navigation system," in *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 478–485, 2005.

[48] J. Li, Q. Wu, and D. Zhu, "Route guidance mechanism with centralized information control in large-scale crowd's activities," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 7–11, 2009.

[49] R. Claes, T. Holvoet, and D. Weyns, "A decentralized approach for anticipatory vehicle routing using delegate multiagent systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 364–373, 2011.

[50] J. Pan, I. S. Popa, K. Zeitouni, and C. Borcea, "Proactive vehicular traffic rerouting for lower travel time," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 8, pp. 3551–3568, 2013.

[51] Z. Liang and Y. Wakahara, "A route guidance system with personalized rerouting for reducing traveling time of vehicles in urban areas," in *Proceedings of IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1541–1548, 2014.

[52] S. Wang, S. Djahel, and J. McManis, "A multi-agent based vehicles re-routing system for unexpected traffic congestion avoidance," in *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2541–2548, 2014.

[53] Z. Cao, H. Guo, J. Zhang, and U. Fastenrath, "Multiagent-based route guidance for increasing the chance of arrival on time," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1180–1187, 2016.

[54] C. Sommer, "Shortest-path queries in static networks," *ACM Computing Surveys*, vol. 46, no. 4, p. 45, 2014.

[55] A. Namoun, C. A. Marín, B. Saint Germain, N. Mehandjiev, and J. Philips, "A multi-agent system for modelling urban transport infrastructure using intelligent traffic forecasts," in *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pp. 175–186, 2013.

[56] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus, "Stochastic motion planning and applications to traffic," in *Algorithmic Foundation of Robotics VIII*, pp. 483–500, 2009.

[57] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2147–2154, 2012.

[58] P. Wei, Y. Cao, and D. Sun, "Total unimodularity and decomposition method for large-scale air traffic cell transmission model," *Transportation Research Part B: Methodological*, vol. 53, pp. 1–16, 2013.

[59] K. Ito and K. Kunisch, *Lagrange multiplier approach to variational problems and applications*, vol. 15. SIAM, 2008.

[60] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[61] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[62] S. Mahadevan, "Samuel meets amarel: Automating value function approximation using global state space analysis," in *Proceedings of The Twentieth AAAI Conference on Artificial Intelligence (AAAI)*, vol. 5, pp. 1000–1005, 2005.

[63] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural Networks*, vol. 13, no. 2, pp. 143–148, 2000.

[64] F. Yuan, L. Han, S.-M. Chin, and H. Hwang, "Proposed framework for simultaneous optimization of evacuation traffic destination and route assignment," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1964, pp. 50–58, 2006.

[65] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[66] M. de B do Amarante and A. L. Bazzan, "Agent-based simulation of mobility in real-world transportation networks: effects of acquiring information and replanning en-route," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1351–1352, 2012.

[67] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno, "Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies," *European Journal of Operational Research*, vol. 151, no. 1, pp. 1–11, 2003.

[68] J. W. Wedel, B. Schünemann, and I. Radusch, "V2x-based traffic congestion recognition and avoidance," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pp. 637–641, 2009.

[69] K. Y. Chan, T. S. Dillon, and E. Chang, "An intelligent particle swarm optimization for short-term traffic flow forecasting using on-road sensor systems," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 10, pp. 4714–4725, 2013.

[70] T. Hunter, A. Hofleitner, J. Reilly, W. Krichene, J. Thai, A. Kouvelas, P. Abbeel, and A. Bayen, "Arriving on time: estimating travel time distributions on large-scale road networks," *arXiv preprint arXiv:1302.6617*, 2013.

[71] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 25–34, 2014.

[72] E. Nikolova, "High-performance heuristics for optimization in stochastic traffic engineering problems," in *the Seventh International Conference on Large-Scale Scientific Computing (LSSC)*, pp. 1–8, 2009.

[73] G. Yang and E. Nikolova, "Approximation algorithms for route planning with nonlinear objectives," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3209–3215, 2016.

[74] K. R. Hutson and D. R. Shier, "Extended dominance and a stochastic shortest path problem," *Computers & Operations Research*, vol. 36, no. 2, pp. 584–596, 2009.

[75] E. Nikolova and N. E. Stier-Moses, "A mean-risk model for the traffic assignment problem with stochastic travel times," *Operations Research*, vol. 62, no. 2, pp. 366–382, 2014.

[76] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: state of the art," *Computers & Operations Research*, vol. 33, no. 11, pp. 3324–3343, 2006.

[77] I. Murthy and S. Sarkar, "A relaxation-based pruning technique for a class of stochastic shortest path problems," *Transportation science*, vol. 30, no. 3, pp. 220–236, 1996.

[78] I. Murthy and S. Sarkar, "Stochastic shortest path problems with piecewise-linear concave utility functions," *Management Science*, vol. 44, no. 11-part-2, pp. S125–S136, 1998.

[79] J. L. Bander and C. C. White, "A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost," *Transportation Science*, vol. 36, no. 2, pp. 218–230, 2002.

[80] E. Nikolova, M. Brand, and D. R. Karger, "Optimal route planning under uncertainty.," in *Proceedings of International Conference on Automated Planning & Scheduling (ICAPS)*, vol. 6, pp. 131–141, 2006.

[81] J. Cheng, S. Kosuch, and A. Lisser, "Stochastic shortest path problem with uncertain delays.," in *ICORES*, pp. 256–264, 2012.

[82] P. Pattanamekar, D. Park, L. R. Rilett, J. Lee, and C. Lee, "Dynamic and stochastic shortest path in transportation networks with two components of travel time uncertainty," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 5, pp. 331–354, 2003.

[83] U. Fastenrath and M. Becker, "Process for selection of a route for a dynamic navigation on private transport," 2008.

[84] H. Frank, "Shortest paths in probabilistic graphs," *Operations Research*, vol. 17, no. 4, pp. 583–599, 1969.

[85] A. Christman and J. Cassamano, "Maximizing the probability of arriving on time," in *International Conference on Analytical and Stochastic Modeling Techniques and Applications*, pp. 142–157, 2013.

[86] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research.," in *ICE Proceedings: Engineering Divisions*, pp. 325–362, 1952.

[87] A. Polus, "A study of travel time and reliability on arterial routes," *Transportation*, vol. 8, no. 2, pp. 141–151, 1979.

[88] M. Mogridge and S. Fry, "Variability of car journey times on a particular route in central london," *Traffic Engineering & Control*, vol. 25, no. HS-038 005, 1984.

[89] F. Montgromery and A. May, "Factors affecting travel times on urban radial routes," *Traffic engineering & control*, vol. 28, no. 9, 1987.

[90] H. Rakha, I. El-Shawarby, M. Arafeh, and F. Dion, "Estimating path travel-time reliability," in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pp. 236–241, IEEE, 2006.

[91] K. Chen, L. Yu, J. Guo, and H. Wen, "Characteristics analysis of road network reliability in beijing based on the data logs from taxis," in *Transportation Research Board 86th Annual Meeting*, 2007.

[92] Z. Cao, S. Jiang, J. Zhang, and H. Guo, "A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, 2016 (In Press).

[93] M. M. De Weerdt, E. H. Gerding, S. Stein, V. Robu, and N. R. Jennings, "Intention-aware routing to minimise delays at electric vehicle charging stations," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (IJCAI)*, pp. 83–89, 2013.

[94] N. Christofides, "The vehicle routing problem," *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, vol. 10, no. 1, pp. 55–70, 1976.

[95] G. Laporte and F. V. Louveaux, "Solving stochastic routing problems with the integer l-shaped method," in *Fleet management and logistics*, pp. 159–167, Springer, 1998.

[96] M. L. Fisher, "Optimal solution of vehicle routing problems using minimum k-trees," *Operations research*, vol. 42, no. 4, pp. 626–642, 1994.

[97] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

[98] M. Reimann, K. Doerner, and R. F. Hartl, "Analyzing a unified ant system for the vrp and some of its variants," in *Workshops on Applications of Evolutionary Computation*, pp. 300–310, Springer, 2003.

[99] D. Ni and H. Wang, "Trajectory reconstruction for travel time estimation," *Journal of Intelligent Transportation Systems*, vol. 12, no. 3, pp. 113–125, 2008.

[100] X. J. Ban, Y. Li, A. Skabardonis, and J. Margulici, "Performance evaluation of travel-time estimation methods for real-time traffic applications," *Journal of Intelligent Transportation Systems*, vol. 14, no. 2, pp. 54–67, 2010.

[101] M. J. Abdi, *Cardinality optimization problems*. PhD thesis, University of Birmingham, 2013.

[102] M. Fazel, H. Hindi, and S. P. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 6, pp. 4734–4739, IEEE, 2001.

[103] M. Udell and S. Boyd, "Maximizing a sum of sigmoids," 2013.

[104] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 trend filtering," *Siam Review*, vol. 51, no. 2, pp. 339–360, 2009.

[105] P. D. Tao *et al.*, "The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems," *Annals of Operations Research*, vol. 133, no. 1-4, pp. 23–46, 2005.

[106] Y.-B. Zhao and D. Li, "Reweighted ℓ_1 minimization for sparse solutions to underdetermined linear systems," *SIAM Journal on Optimization*, vol. 22, no. 3, pp. 1065–1088, 2012.

[107] G. Pataki, M. Tural, and E. B. Wong, "Basis reduction and the complexity of branch-and-bound," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1254–1261, Society for Industrial and Applied Mathematics, 2010.

[108] MathWorks, *Mixed-Integer Linear Programming Algorithms*, 2014. http://www.mathworks.com/help/optim/ug/mixed-integer-linear-programming-algorithms.html.

[109] Y. Zhang, "Solving large-scale linear programs by interior-point methods under the matlab environment," *Optimization Methods and Software*, vol. 10, no. 1, pp. 1–31, 1998.

[110] E. Danna, E. Rothberg, and C. Le Pape, "Exploring relaxation induced neighborhoods to improve mip solutions," *Mathematical Programming*, vol. 102, no. 1, pp. 71–90, 2005.

[111] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, vol. 18. Wiley New York, 1988.

[112] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices," in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, pp. 2156–2162, IEEE, 2003.

[113] W. Zhang, "Branch-and-bound search algorithms and their computational complexity.," tech. rep., DTIC Document, 1996.

[114] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311, 1984.

[115] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," *Advances in neural information processing systems*, vol. 16, pp. 177–184, 2004.

[116] E. Santos Jr and E. S. Santos, "Polynomial solvability of cost-based abduction," *Artificial Intelligence*, vol. 86, no. 1, pp. 157–170, 1996.

[117] I. J. Lustig, R. E. Marsten, and D. F. Shanno, "Computational experience with a primal-dual interior point method for linear programming," *Linear Algebra and Its Applications*, vol. 152, pp. 191–222, 1991.

[118] A. Faye and F. Roupin, "Partial lagrangian relaxation for general quadratic programming," *4OR*, vol. 5, no. 1, pp. 75–88, 2007.

[119] D. Schuurmans, F. Southey, and R. C. Holte, "The exponentiated subgradient algorithm for heuristic boolean programming," in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 334–341, 2001.

[120] K. R. Rebman, "Total unimodularity and the transportation problem: a generalization," *Linear Algebra and Its Applications*, vol. 8, no. 1, pp. 11–24, 1974.

[121] T. K. Dey, A. N. Hirani, and B. Krishnamoorthy, "Optimal homologous cycles, total unimodularity, and linear programming," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 1026–1044, 2011.

[122] B. Bonet, "An admissible heuristic for sas$^+$ planning obtained from the state equation," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2268–2274, 2013.

[123] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2009.

[124] R. Naz, F. Mahomed, and D. Mason, "Conservation laws via the partial lagrangian and group invariant solutions for radial and two-dimensional free jets," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 6, pp. 3457–3465, 2009.

[125] C. P. Gomes, "Structure, duality, and randomization: Common themes in ai and or," in *Proceedings of the Seventeenth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1152–1158, 2000.

[126] O. Sumer, U. A. Acar, A. T. Ihler, and R. R. Mettu, "Fast parallel and adaptive updates for dual-decomposition solvers," in *Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1076–1082, 2011.

[127] F. Barahona and R. Anbil, "The volume algorithm: producing primal solutions with a subgradient method," *Mathematical Programming*, vol. 87, no. 3, pp. 385–399, 2000.

[128] J.-L. Goffin and K. C. Kiwiel, "Convergence of a simple subgradient level method," *Mathematical Programming*, vol. 85, no. 1, pp. 207–211, 1999.

[129] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, "An analysis of several heuristics for the traveling salesman problem," *SIAM journal on computing*, vol. 6, no. 3, pp. 563–581, 1977.

[130] M. Grant, S. Boyd, and Y. Ye, *Disciplined convex programming*. Springer, 2006.

[131] T. S. Kumar, "Fast (incremental) algorithms for useful classes of simple temporal problems with preferences.," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1954–1959, 2007.

[132] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pp. 99–108, 2010.

[133] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pp. 136–147, IEEE, 2014.

[134] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing, IEEE*, vol. 7, no. 4, pp. 12–18, 2008.

[135] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 316–324, ACM, 2011.

[136] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: enhancing driving directions with taxi drivers' intelligence," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 1, pp. 220–232, 2013.

[137] J. J. Júdice, A. M. Faustino, and I. M. Ribeiro, "On the solution of np-hard linear complementarity problems," *Top*, vol. 10, no. 1, pp. 125–145, 2002.

[138] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[139] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[140] H. Yu and D. P. Bertsekas, "On boundedness of q-learning iterates for stochastic shortest path problems," *Mathematics of Operations Research*, vol. 38, no. 2, pp. 209–227, 2013.

[141] S. Carden, "Convergence of a q-learning variant for continuous states and actions," *Journal of Artificial Intelligence Research*, pp. 705–731, 2014.

[142] A. Fuchs, D. Axehill, and M. Morari, "Efficient evaluation of mp-miqp solutions using lifting," *arXiv preprint arXiv:1311.4752*, 2013.

[143] M. Papageorgiou, "Dynamic modeling, assignment, and route guidance in traffic networks," *Transportation Research Part B: Methodological*, vol. 24, no. 6, pp. 471–495, 1990.

[144] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[145] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[146] R. Yang, A. X. Jiang, M. Tambe, and F. Ordonez, "Scaling-up security games with boundedly rational adversaries: A cutting-plane approach," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence (IJCAI)*, pp. 404–410, 2013.

[147] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: an overview," in *Proceedings of the Third International Conference on Advances in System Simulation*, pp. 55–60, 2011.

[148] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.