Optimizing Passenger On-Vehicle Experience through Simulation and Multi-Agent Multi-Criteria Mobility Planning

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Rongye Shi

B.S., Electronic Information Science and Technology, China Agricultural University
M.E., Electronics and Communication Engineering, Peking University
M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

May 2019

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors, Prof. Manuela M. Veloso and Prof. Peter Steenkiste, for their guidance and generous supports throughout my doctorate study. In particular, I want to thank Manuela. The ways she thinks of problems usually inspire me to achieve a higher research level. I am impressed by her passion and dedication to academic problems. Her remarkable comments and suggestions lead me to high-quality work. I want to thank Peter as well for his constructive feedbacks and comments that extend my knowledge in the field I am working on. His philosophy in doing research truly influences me and helps me present my work in a clear and logical manner. Many thanks to both of them, and without their guidance and support, my research would not have been possible.

I would like to thank Prof. Stephen F. Smith and Prof. Carlee Joe-Wong for being in my thesis committee. They always share insights with me about the technical questions whenever I present to them. I benefit a lot from their comments on my research and thesis.

Many thanks to all my colleagues and friends at CORAL lab: Dr. Vittorio Perera, Anahita Mohseni-Kabir, Devin Schwab, Philip Cooksey, Rui Silva, Max Korein, Kim Baraka, Ishani Chatterjee, Aaron Roth, Travers Rhodes, Kevin Zhang, Ashwin Khadke, Sai Prabhakar, etc., for their friendship and helpful discussions when requested. They have provided a wonderful and friendly academic atmosphere in the lab.

Thank Prof. Susana Sargento and Prof. Ana Aguiar for providing the data necessary for my research.

I would like to acknowledge the funding support from FCT under the CarnegieMellon-Portugal ERI S2MovingCity project (Grant CMUP-ERI/TIC/0010/2014) for making my research possible.

Finally, I truly thank my family, especially my father Youjian Shi, mother Chunhuan Luo and elder sister Minglu Shi, for their love and care during my study at Carnegie Mellon.

ABSTRACT

The rapid growth in urban population poses significant challenges to moving city dwellers in a fast and convenient manner. This thesis contributes to solving the challenges from the viewpoint of public transit passengers by improving their on-vehicle experience. Traditional transportation research focuses on pursuing minimal travel time of vehicles on the road network, paying no attention to people inside the vehicles. In contrast, the research in this thesis is passenger-driven, concerning the role of the on-vehicle experience in mobility planning through the public transit systems. The primary goal of the thesis is to address the following problem: Given an urban public transit network, how can we plan for the optimal experience of passengers in terms of their service preference?

There are several challenges we have to address to meet this goal. First, a model or a simulator that captures not only the road traffic, but also the behaviors of passengers and other relevant factors is a prerequisite for this research but has seldom been developed previously. Second, to plan for passengers' mobility concerning the influence among passengers as well as multiple service preferences is computationally intensive, especially on a city scale.

To achieve the research goal and overcome the challenges, this thesis develops a joint traffic-passenger simulator, which simulates the road traffic, behaviors of passengers and on-vehicle environment dynamics. Specifically, the simulator combines the urban road traffic, the interactions among the passengers and the infrastructures that support certain on-vehicle services, such as on-vehicle Wi-Fi, to provide a passenger-level simulation. A separate passenger behavior model and on-vehicle Wi-Fi service model are designed to run jointly with SUMO, a mature traffic simulator, for simulating the passenger behaviors and on-vehicle travel experience. A joint simulator for the bus transit system in the city of Porto, Portugal has been implemented and tested by comparing the simulation to the real passenger data.

To configure the background passenger flow in the simulation, real passenger data are used. The data were collected by an entry-only system and the destination information was missing. This thesis contributes a machine learning algorithm, called semi-supervised self-training, to infer the missing destinations with a high inference confidence level.

Given the simulation platform, the passenger mobility planning problem can be formalized as a multi-agent path planning (MAPP) problem, where multiple passengers may interfere with each other when contending for service resources. The mobility planning operates on the client

passengers (i.e., a subset of the overall passengers who request the planning service from our planner). State-of-the-art MAPP solvers, such as M*, do not scale well to such a MAPP problem. This thesis proposes the soft-collision-free M* (SC-M*), a generalized version of M*, to efficiently handle the MAPP task under complex urban environments (i.e., with a large client passenger size and multiple types of client passengers requesting multiple types of service resources). We evaluate the performance of the SC-M* through a case study of the bus transit system in Porto, Portugal and the experimental results show the advantages of the SC-M* in terms of path cost, collision-free constraint, and the scalability in run time and success rate.

Contents

1 Introduction **1**

1.1 Thesis Question . . . 2

1.2 Approach . . . 3

1.2.1 Simulating On-Vehicle Experience through Joint Traffic-Passenger Simulation 3

1.2.2 Retrieving the Missing Features of Real Data with Semi-Supervised Learning 4

1.2.3 Planning the Mobility with a Scalable Multi-Passenger Multi-Criteria Planner 4

1.3 Contributions . . . 5

1.4 Reading Guide to the Thesis . . . 5

2 A Joint Traffic-Passenger Simulator for On-Vehicle Experience Simulation (P-SUMO) **8**

2.1 Simulation for Intelligent Public Transportation Systems (IPTS) . . . 8

2.2 Joint Traffic-Passenger Simulator (P-SUMO) . . . 10

2.2.1 Overview of the Methodology for Passenger-Level Simulator . . . 10

2.2.2 Pre-Processing Layer . . . 12

2.2.3 Joint Simulation Layer . . . 19

2.2.4 Evaluation of Synthetic Data with Real Passenger Data . . . 22

2.3 Modeling the On-Vehicle Wi-Fi Service . . . 26

2.4 Usage of Synthetic Data . . . 30

2.4.1 Training and Testing Data Preparation . . . 30

2.4.2 Neural Network Implementation and Performance . . . 31

2.5 Summary . . . 33

3 Infer the Missing Features in Incomplete Data **34**

3.1 Motivations and Challenges of Filling the Missing Features . . . 34

3.1.1 Destination Inference for Passenger Data . . . 35

3.2 Heuristic Method . . . 37

3.3 Methodology . . . 40
3.3.1 Semi-Supervised Learning Problem Setting . . . 40
3.3.2 One-Time Inference Method . . . 40
3.3.3 Self-Training (ST) . . . 42
3.3.4 Self-Training using Personal Historical Information (STP) . . . 46
3.4 Data Pre-Processing for Experiments . . . 51
3.4.1 Passenger Simulation Data of Porto City . . . 51
3.4.2 AFC Passenger Data of Porto City . . . 51
3.5 Experiments . . . 52
3.5.1 Experimental Setup and Evaluation Metrics . . . 52
3.5.2 Results of Parameter Sensitivity . . . 54
3.5.3 Comparison among Methods . . . 56
3.6 Summary . . . 57

4 SC-M*: A Multi-Agent Path Planning with Soft-Collision-Free Constraint 59
4.1 Background of MAPP . . . 60
4.2 Motivations of SC-M* . . . 62
4.3 Technical Briefing of M* . . . 62
4.3.1 MAPP Problem Definition . . . 63
4.3.2 Graphic-Centric Description of M* . . . 64
4.3.3 Algorithm Description of M* . . . 65
4.3.4 M* Challenges and Improvements . . . 66
4.4 Soft-Collision-Free M* (SC-M*) . . . 69
4.4.1 Soft-Collision-Free Constraint . . . 69
4.4.2 Completeness and Cost-Suboptimality . . . 75
4.5 Experiments and Results . . . 77
4.5.1 Planning for the One-Resource-One-Agent-Type . . . 79
4.5.2 Planning for the Two-Resource-Two-Agent-Type . . . 79
4.5.3 Comparison of SC-M* to Baselines . . . 81
4.6 Summary . . . 86

5 SC-M* for Real-World Bus Transit System 87
5.1 Multi-Passenger Multi-Criteria Mobility Planning in Real Public Transit . . . 87
5.2 Briefings of Relevant Building Blocks in Previous Chapters . . . 89

5.2.1 SC-M* . . . 89
5.2.2 Retrieving the Missing Destination in Real Data . . . 89
5.2.3 On-Vehicle Experience Simulator . . . 90
5.3 Additional Stepping Stones Prior to Applying SC-M* . . . 90
5.3.1 Expectation Graph of Bus Transit Network . . . 91
5.3.2 Forward-the-Tail-Agent Approach . . . 93
5.4 Experiments and Evaluations . . . 95
5.4.1 Evaluations with Respect to Path Cost and the Hard-Collision-Free Constraint 95
5.4.2 Evaluations with Respect to Scalability . . . 99
5.5 Summary . . . 103

6 Related Work **104**
6.1 Intelligent Public Transportation Systems . . . 104
6.2 Traffic and Passenger Simulation . . . 105
6.2.1 Simulation and Transfer Learning . . . 105
6.2.2 Passenger Modeling and Road Traffic Simulation . . . 106
6.3 Destination Estimation for Passenger Data . . . 106
6.4 Passenger Mobility Planning . . . 108
6.4.1 Shortest Path Algorithms . . . 108
6.4.2 Multi-Criteria Route Planning . . . 110
6.4.3 Multi-Agent Path Planning . . . 110

7 Conclusion and Future Work **112**
7.1 Contributions . . . 112
7.2 Future Work . . . 113

A SC-M* for Bottleneck-Featured Environment **115**

Bibliography **117**

List of Figures

2.1 Block diagram of the joint traffic-passenger simulator P-SUMO. 11
2.2 Block diagram of pre-processing layer. 12
2.3 Virtual traffic and bus transit network in SUMO. 13
2.4 Expected taxi demands on Wednesday. 14
2.5 Graph of the bus transit network. 16
2.6 Travel plan from stop AAL4 to SDP1. 19
2.7 Block diagram of joint-simulation layer. 20
2.8 Passenger behavior model. 20
2.9 Examples of simulation output log. 22
2.10 Number of transactions vs. date in May for AFC data. 23
2.11 Demand occurrence probability vs. hour in a day. 24
2.12 Spatial KL divergence $D_{KL\_spatial}$ vs. time period T. 26
2.13 Map of Wi-Fi hotspot locations in Porto. 27
2.14 Map of the on-vehicle Wi-Fi capacity in Mbps. 28
2.15 Example of the average Wi-Fi quality profile along Trip 801-1 on the Wednesday of May 5, 2010. 29
2.16 ANN structure and learning results. 31
2.17 Passenger volume prediction at each stop: Trip (a) 207-21 (b) 207-35. 32

3.1 Entry-only automated transaction system. 35
3.2 Traditional methods can apply to passengers who have multiple daily trips. 36
3.3 Heuristic assumption example. 38
3.4 Diagram of semi-supervised self-training. 43
3.5 The design of base predictor. 45
3.6 Selection strategy. 46
3.7 Construction of personal alighting distribution q. 48

3.8 Voting process behind the modify alighting distribution p'. 48
3.9 Modification of p using q under orthogonality condition. 49
3.10 Breakdowns of the data used in experiments. 53
3.11 Impact of the bandwidth on accuracy ($N_k^{-1} = 10$). 55
3.12 Impact of the selection threshold on accuracy ($\Delta = 2$ hours for AFC data, $\Delta = 3$ hours for simulation data). 56

4.1 Illustration of traditional M* for two agents, where we show the evolution of the expanded graph G^{exp} (circle), neighbor graph G^{nbh} (diamond), and policy graph G^{ϕ} (square) for Agent 2 as the M* algorithm proceeds. 67
4.2 Planning for four agents with origins (O_1, O_2, O_3, O_4) and destinations (D_1, D_2, D_3, D_4). 68
4.3 Example designs of CDFs, mapping the resource experience of an agent to a collision probability on certain resource. f_1: sigmoid-based CDF for important (sensitive) resources. f_2: linear CDF for trivial (insensitive) resource. δ: offset parameter adjusting the *tolerance level*. 72
4.4 Counterexample of the inclusion property of SC-M* under the soft-collision context. Agent $r1, r2$, and $r3$ have the planning O-D demands $(a, e), (f, g)$, and (h, i), respectively. Vertexes in the system are labeled as a, b, c, etc. 76
4.5 Grid system with 20x20 stops and 40 bidirectional lines. 77
4.6 Impact of the collision threshold T (given $\delta = 6.0$) and offset parameter δ (given $T = 0.35$) on one-resource-one-type SC-M*. 78
4.7 Impact of the collision threshold T (given $\delta = 9.0$) and offset parameter δ (given $T = 0.35$) on two-resource-two-type SC-M*. 80
4.8 Average cost difference of SC-based MAPP solvers from the individual optimal cost under the one-resource-one-type context. 81
4.9 Illustration of the difference in planning among SC-A*, SC-M*, and SC-CBS. 83
4.10 Success rate, cost, and run time ratio of the three SC-based MAPP solvers under different T. 85
4.11 Success rate, cost, and run-time ratio of the three SC-based MAPP solvers under different δ. 86

5.1 Time-expanded graph for the bus transit network of one weekday. (a) shows the time expanded graph for the blue route and (b) shows how passengers can transfer between two different routes. (c) shows the transfer links between walking-reachable stops. All dashed vertexes/ovals indicate the set of events occurring at certain stop V. 92
5.2 Forward-the-tail-agent approach to handle non-synchronization in real systems. . . . 94
5.3 Comparison of the path costs between individual optimal A* and SC-M* ($T = 0$). . . . 96
5.4 The simulation log of the execution of plans generated by individual optimal A* and SC-M* ($T = 0$). Note the SC-M* ($T = 0$) is equivalent to the traditional hard-collision-free M*. 98
5.5 a) The impact of the collision threshold T on the success rate of SC-M*, given $\delta = 6.0$; b) The impact of the offset parameter δ on the success rate, given $T = 0.35$. 100
5.6 The maximal collision score observed in the planned paths in the planning phase and in the testing phase. To satisfy the constraint, the maximal observed collision score is expected to be less than the collision threshold T. 102

A.1 Transit system with bottlenecks. Blocked areas are shown in black. 115

List of Tables

2.1 Terminology. . . . 10
2.2 Examples of trip demand samples (O, D, t). . . . 16
2.3 Vertex and edge attributes. . . . 17
2.4 Feature and target of 4-D input data. . . . 30
2.5 ANN structure and average prediction error. . . . 32

3.1 Simulation data examples. . . . 51
3.2 Porto AFC data examples. . . . 52
3.3 Destination inference performance of different methods. . . . 57

4.1 Run time of one-resource-one-type SC-M* under different parameters. . . . 78
4.2 Run time of two-resource-two-type SC-M* under different parameters. . . . 80
4.3 Results of the cost optimality/suboptimality experiments. . . . 82
4.4 Average run time of SC-based MAPP solvers under the one-resource-one-type context. 84

5.1 The impact of T and δ on run time (in seconds). . . . 100

A.1 Records of the SC-A* applied to the bottleneck scenario, given $T = 0.35, \delta = 9.0$. . . . 116
A.2 Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.25, \delta = 6.0$. . . . 116
A.3 Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.25, \delta = 2.0$. . . . 116
A.4 Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.05, \delta = 6.0$. . . . 116

Chapter 1

Introduction

The domestic migration into urban areas generates significant challenges for cities, making an efficient and sustainable management of services and resources more than ever necessary. A central service affecting many of the others is mobility. To satisfy the mobility needs given the rapid growth in city population is challenging. The concept of *Smart City* has been proposed to cope with the challenges and to provide access to all the traditional services a city needs to provide citizens in the context of rising population. An important aspect of Smart City is the *Intelligent Public Transportation System (IPTS)*, which utilizes information and synergistic technologies to improve public transportation systems of all kinds.

Traditional research to IPTS focuses on using the information about the state of the transportation networks for scheduling and dispatching mass transit. Although a lot of progress has been made improving the IPTS in terms of infrastructure construction, information sharing, and fleet scheduling, limited attention is paid to the people inside the vehicles, especially to the experience of the travel from the viewpoint of the passengers that use the public transits. For example, in public transit systems, different passengers have different preferences, even necessities, in terms of public resources, such as seat availability (necessary for seniors) or on-vehicle Wi-Fi supply (preferred by video viewers and game players during the trip). These factors seriously affect the satisfactory level of their travel experience on vehicles.

We advocate that, to develop an efficient IPTS, *passenger-centered* research is a necessity, providing customized mobility plans directly to each passenger for a trip with satisfactory on-vehicle experience. This insight comes from the fact that an individual passenger can neither change the existing public transportation infrastructures nor the massive passenger and traffic flow, but rather, can choose an alternative well-designed strategy to accomplish the travel

demand while making his or her experience pleasant.

It is a nontrivial matter to help passengers achieve this goal. On one hand, research to developing a model or a simulator that captures the road traffic, passenger behaviors and other factors relevant to the on-vehicle experience is a prerequisite to the goal but has not been considered previously. Specifically, the on-vehicle experience is affected by the surrounding passengers, traffic conditions, and facilities that support the public on-vehicle services. In addition, the mobility planner should consider the influence among passengers as well as multiple service preferences, which is challenging in the field of planning and scheduling, especially on a city scale.

In this thesis, we attempt to address the challenges from the viewpoint of passengers by considering the important role of passenger experience when planning for pleasant trips through the urban public transportation systems. The thesis explores and contributes a passenger-centered planner for multiple passengers requesting multiple types of public services while limiting the level of interference among passengers. We assume that the total set of passengers is divided into two subsets – one with passengers who request planning service from the planner, called as client passengers, and the other with non-client passengers who do not.

1.1 Thesis Question

The research problem in this thesis is as follows:

> *Given a public transit network of a city and a number of client passengers who request some planning service, how can we plan for the client passengers to optimize their travel experience in terms of their service preferences?*

The research problem has three implications. First, the public transit network is a prerequisite and has to be modeled before planning. The network model should be based on information about the real-world public transit systems. In contrast to static information about the public transit infrastructures and schedules, which is usually accessible, dynamic information about the road traffic, passenger behaviors, and on-vehicle environments is usually not accessible. This requires a simulator that combines the accessible information and a series of traffic-passenger behavior models, to simulate certain aspects of the public transit systems and to generate the dynamic information on request.

Second, the accessible information includes real-world data that are collected by existing transaction systems. Those real-world data are usually incomplete with important features miss-

ing, and thus pre-processing and data mining approaches are needed in order to make use of the real-world data to drive the simulation.

Third, planning tries to improve the travel experience, which requires optimizing over multiple criteria related to each client individual, and limiting the level of interference among multiple client passengers. The travel experience of client passengers are coupled with each other and the interference among them has to be considered. In short, the mobility planner, which is able to address the thesis problem, is expected to be applicable to the real-world public transit systems, customize for client passengers of different types and handle their mobility requests on a city scale.

1.2 Approach

At a high-level, to address the thesis problem, we first need a simulator for simulating the public transit on-vehicle experience to generate data for constructing traffic models, and then based on the model, we conduct mobility planning. The blueprint is that the on-vehicle experience simulator provides information necessary for constructing a graph (a model consisting of vertexes and edges) that describes the public transit system and its dynamics. Based on the graph, we can make mobility plans to efficiently accommodate multiple passengers in terms of multiple service preference and interference constraints. We decompose our approach into three topics.

1.2.1 Simulating On-Vehicle Experience through Joint Traffic-Passenger Simulation

Our approach to the on-vehicle experience simulation is to extend the mature open-source road traffic simulator (SUMO): We design a separate passenger behavior model and a model of the on-vehicle public services to run jointly with SUMO to generate the synthetic data about the travel experience of passengers for post-processing.

To this end, we establish a road network of a city along with a public transit system in SUMO, allowing SUMO to simulate the road traffic and the public transit operations. To capture the passenger behaviors in the simulation, we leverage the control interface of SUMO, TraCI, and develop a monitor-control algorithm to modify the transit conditions, run the passenger behavior model (e.g., waiting at the stop, boarding, and alighting), and record important moments in real time.

To capture an on-vehicle public service, we introduce an on-vehicle Wi-Fi service model. The on-vehicle free Wi-Fi service is assumed to be supported by the city hotspots in the city.

With the hotspot location data, we construct an on-vehicle Wi-Fi service quality map, which maps a geographical location to an on-vehicle quality value. The Wi-Fi service allocated to each passenger is a function of passenger volume and the distance from the hotspot in connection. This model allows us to simulate the on-vehicle Wi-Fi service experience for each passenger. The way we verify the traffic-passenger joint simulator is to compare the synthetic simulation results to the real data.

1.2.2 Retrieving the Missing Features of Real Data with Semi-Supervised Learning

In order to reflect the reality as much as possible, we use some real data to set up the simulation. The passenger transaction dataset is one of the accessible sources, which can be used to directly configure the background passenger flow. However, the passenger transaction data that accessible to us are entry-only and the destination information is missing. This reduces their value for planning purpose.

To make full use of the data, we must explore approaches to retrieve the missing destination information. Traditional heuristic methods can lead to a limited proportion of missing destinations being matched. Following the intuition that the heuristic-based matched data may contain useful information for us to process the remaining unmatched data, we make a step further and develop a semi-supervised learning method, called self-training, to continue the inference. To further improve the inference, we also consider using the personal historical information to improve in the *base predictor*, a core component of the proposed self-training.

1.2.3 Planning the Mobility with a Scalable Multi-Passenger Multi-Criteria Planner

The data-driven on-vehicle experience simulation generates information necessary for building the graphs of the public transit networks, based on which the mobility planning can be conducted. In addition to optimizing over a linear combination of multiple criteria, we formalize the planning problem as a multi-agent path planning (MAPP) problem, where multiple client passengers are coupled and may interfere with each other when using services. State-of-the-art MAPP solvers, such as M*, do not scale well to such a city-scale MAPP task. One bottleneck of M* is that the hard-collision-free constraint is too restricted, incurring an explode in the size of the search graph when many agents collide.

We introduce the concept of soft collisions to the MAPP setting, where a collision among client passengers could be soft, quantified using a *collision score*. The scores reflect the quality of

their experience as they contend for services. We then develop a generalized version of M*, the soft-collision-free M* (SC-M*), for solving the MAPP problem under the soft-collision context. Specifically, the algorithm tracks the collision score of each client passenger and places the client passengers, whose collision scores exceed some threshold, into a soft-collision set for search space expansion. In this way, SC-M* achieves improved scalability to handle a larger number of client passengers while limiting the probability of collisions among the client passengers to a bound.

1.3 Contributions

The key contributions of this thesis are as follows:

- **P-SUMO (Passenger-SUMO)**, a simulator for on-vehicle experience simulation based on the joint traffic-passenger approach, which captures not only the interactions among road traffic, public vehicles, and passengers but also the dynamics of the on-vehicle services, including Wi-Fi quality and space availability;
- **A data mining method** that infers the missing features in the real passenger data: The missing destination information in the real automated fare collection (AFC) data is retrieved by the proposed self-training method with high confidence;
- **SC-M* – A multi-agent path planning algorithm with soft-collision-free**, which has advanced flexibility and scalability for solving the MAPP problems under the soft-collision context and can handle complex environments with multiple types of agents requesting multiple types of resources;
- **A scalable multi-passenger multi-criteria mobility planner for Porto Bus Transit System**: We use different types of real data from Porto and make a case study of the bus transit system in this city. Based on the on-vehicle experience simulation driven by the real data, we successfully apply the SC-M* framework to the real-world system to improve the on-vehicle experience of client passengers.

1.4 Reading Guide to the Thesis

The remainder of this thesis document is arranged as follows:

Chapter 2 – A Joint Traffic-Passenger Simulator for On-Vehicle Experience Simulation (P-SUMO) This chapter introduces the joint traffic-passenger simulator, called P-SUMO, for simulating the passengers' on-vehicle experience. We develop the P-SUMO for the bus transit system in Porto, Portugal. A demonstration of using the simulator to generate synthetic bus passenger data is made. The synthetic bus passenger data have significant similarity to the real passenger data. We further extend the simulation with on-vehicle Wi-Fi service models and use the city hotspot location data to build the on-vehicle Wi-Fi service quality map. The simulation outputs provide detailed passenger experience information and bus state information for post-processing.

Chapter 3 – Infer the Missing Features in Incomplete Data This chapter addresses the missing-feature issue in real passenger transaction data, whose destination information is unavailable. We propose a semi-supervised learning method, called self-training, to make use of the heuristic-based matched data to infer the destinations. Evaluations based on the simulation data and a subset of the real bus AFC data are described. We show that the proposed method achieves a higher accuracy than baselines.

Chapter 4 – SC-M*: A Multi-Agent Path Planning with Soft-Collision-Free Constraint This chapter introduces the soft collision property among the agents and presents a generalized version of M* – soft-collision-free M* (SC-M*), to solve the multi-agent path planning problem under soft-collision context. We demonstrate the SC-M* on a grid network. The results show that SC-M* has advanced flexibility and scalability to handle complex environments, e.g., with multiple types of agents requesting multiple types of resources. Comparisons to other SC-based MAPP solvers are made and the advantages/trade-offs of SC-M* are studied.

Chapter 5 – SC-M* for Real-World Bus Transit System This chapter details a case study of the SC-M* in the bus transit system in Porto: To optimize the mobility of multiple client passengers over multiple public services while limiting the collision risk. We leverage the data-driven on-vehicle experience simulator presented in Chapter 2, which simulates the passenger behaviors and on-vehicle service dynamics, and test the SC-M* on it. Experiments are conducted to evaluate the advantages of SC-M* in terms of path cost, collision-free constraint, and scalability in run time and success rate.

Chapter 6 – Related Work This chapter reviews previous literature related to the approach

presented in this thesis. In particular, we focus on traffic simulation, destination estimation for passenger data, and traditional passenger mobility planning. The mobility planning includes the topics on shortest path algorithms, multi-criteria route planning and multi-agent path finding.

Chapter 7 – Conclusion and Future Work This chapter concludes this thesis with a briefing of its key contributions and expected future work.

Chapter 2

A Joint Traffic-Passenger Simulator for On-Vehicle Experience Simulation (P-SUMO)

A model that reflects the necessary aspects of real-world public transit, passenger behaviors, and on-vehicle service conditions is a prerequisite for passenger-level mobility planning. Construction of the model requires access to large-scale and multisource data, ranging from the static urban road-map to records on the dynamics of passenger flow. However, due to the concerns of privacy infringement and cost, the data about the dynamics of the city are usually inaccessible to researchers. A solution to this issue is simulation, which generates synthetic data that, to a reasonable extent, reflect real-world systems. This chapter realizes the idea through a novel *joint traffic-passenger simulator*, called P-SUMO (**P**assenger-**SUMO**) to simulate the on-vehicle experience and generate synthetic data of interest. Much of the work contained in this chapter was presented in [1].

2.1 Simulation for Intelligent Public Transportation Systems (IPTS)

To build an effective IPTS, a profound understanding of the dynamics of urban road traffic, passenger behaviors, and the environments inside public vehicles is needed. It has long been thought that research into estimating people's behaviors and mobility patterns in cities requires large-scale and multisource human mobility data.

The availability of the human mobility data is two-sided:

- With the advance of sensing technologies and the widespread use of automated data collection (ADC) systems in public transportation operations, it is possible to collect large

quantities of diverse data about cities (e.g., cell phone location data [2], vehicle GPS data [3, 4, 5], automated fare collection [AFC] data [including buses [6], underground trains [7], and so forth]). With heterogeneous and ubiquitous datasets, researchers have significantly expanded their knowledge about human mobility [8, 9, 10], and plenty of machine learning (ML) approaches, such as classification, regression, and clustering, have been successfully applied to the field of transportation study [11, 12, 13].

- When it comes to passenger-related research, the data accessible to researchers are usually insufficient, either because the data are *incomplete*, with important features missing, or because the data are only *indirectly related* to the topic of focus. *Complete data* are usually lacking due to the difficulty in most urban infrastructures to integrate large-scale multsource data in a timely and low-cost fashion. This lack-of-complete-data issue limits passenger-related research. For example, in bus transit systems, the bus passenger data are usually collected by automated passenger count (APC) systems or AFC systems. The data collected by those systems are often incomplete (no alighting feature is recorded), limiting the estimation of the overall trip demand profile in the city. More seriously, origin-destination (O-D) surveys to retrieve the destination information are expensive in terms of human efforts and financial cost. As a result, many state-of-the-art methods for bus passenger estimation and prediction (e.g., [14, 15, 16]) cannot be validated due to the missing features in real data.

To cope with this issue, simulation is a solution. Simulation can combine, and be driven by, some indirectly related data collected from a different source, with some features correlating positively to those in the unknown complete data. In our research, the complete data refer to the data about the on-vehicle experience of passengers and the indirectly related data refer to any information we can access to drive the simulation. Grounded on the indirectly related data, we attempt to develop a simulator for generating synthetic complete data that are most likely to be observed in the real world.

Simulation has proven itself as a prerequisite to building intelligent transportation systems. In traffic control research, instead of directly applying an approach to control the real traffic flow, testing it through simulation and transferring the knowledge to real urban road networks can save a lot of cost and avoid safety problems.

Traffic simulation, such as bus transportation simulation, has experienced rapid and significant developments. Many road traffic simulators (e.g., VISSIM, AIMSUN, Matsim, SUMO, etc.) have been developed with good performance. One commonly used open-source traffic simulator

Term	Definition	Explanation and Example
Trip Demand	A tuple (origin, destination, trip starting time)	A trip demand is (**O, D, t**).
Travel Plan	A set of midway O-D pairs without time information	A travel plan is $\{(\mathbf{O}, D_1), (O_2, D_2), ..., (O_n, \mathbf{D})\}$.
Travel Demand	A tuple (origin, destination, trip starting time, travel plan)	A travel demand is $(\mathbf{O, D, t}, \{(\mathbf{O}, D_1), (O_2, D_2), ..., (O_n, \mathbf{D})\})$.
Trip Demand Generative Model	The description of the distribution from which a passenger trip demand is generated	The distribution model specifies the probability of the occurrence of each trip demand in the demand space.
Experience	The sequence of circumstances and events that the passenger encounters during a trip, and their occurrence in time	During the trip from **O** to **D** starting at time **t**, the passenger may take several transits to get to stop **D**. Then, the experience can be a set of tuples $\{(\mathbf{O}, D_1, \mathbf{t}), (O_2, D_2, t_2), ..., (O_n, \mathbf{D}, t_n)\}$. Here, t_n is the time of arriving at O_n.

Table 2.1: Terminology.

is SUMO[1] (Simulation of Urban MObility), which provides a platform for explicitly simulating vehicles, including cars, buses, and urban trains on a city scale. However, most traffic simulators currently do not provide information about passenger vehicle interactions, which is of great interest in bus passenger behavior and prediction studies.

To fill the gap between passenger on-vehicle experience and traffic simulation, we introduce the simulator P-SUMO for simulating the passengers' on-vehicle behavior and experience in conjunction with the mature traffic simulator (SUMO) on a city scale. To the best of our knowledge, this is the first attempt to generate synthetic passenger data through city-wide traffic-passenger joint simulation. We start with developing the passenger behavior model for the passenger-level simulation and then extend the simulator with the on-vehicle service model for the on-vehicle experience simulation.

2.2 Joint Traffic-Passenger Simulator (P-SUMO)

This section describes the design of the joint simulator, Passenger-SUMO (P-SUMO). Some terminologies used in this section are explained in Table 2.1.

2.2.1 Overview of the Methodology for Passenger-Level Simulator

Neither modeling passenger behavior, nor simulating traffic can generate complete passenger data independently, leading to the idea of combining passenger modeling and traffic simulation. First of all, modeling passenger behavior specifies how people's travels are demanded

[1] http://sumo.sourceforge.net

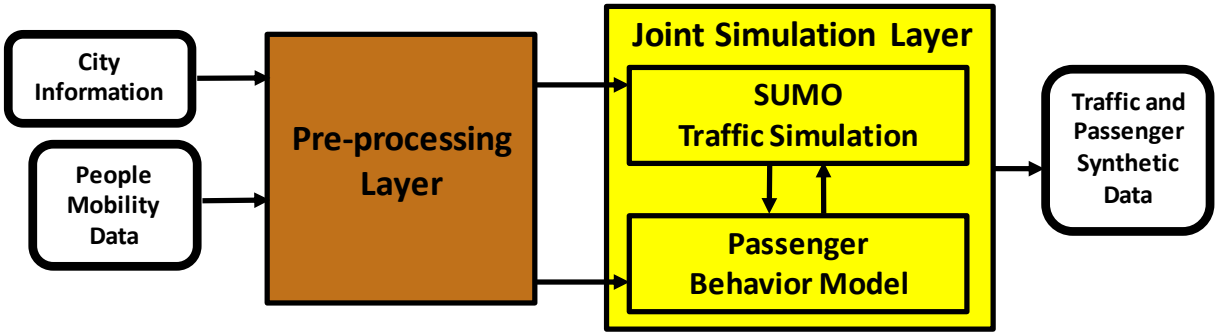

Figure 2.1: Block diagram of the joint traffic-passenger simulator P-SUMO.

and planned (e.g., the O_i and D_i in Table 2.1), but it does not detail the passengers' experience during the travel through an urban traffic environment (e.g., the t_i in Table 2.1). This missing information about experience can be supplemented with traffic simulation. Second, most traffic simulations provide representations of transportation systems and vehicle behaviors, especially how public transits operates in urban road networks. However, passenger-level travel demands/behaviors and the corresponding impact on the public transportation systems (e.g., bus dwell time affected by passengers) are unavailable. This can be supplemented with passenger modeling. Combining passenger modeling and traffic simulation is an effective way to employ the strong points of both approaches and overcome the shortcomings of either.

At a high level, the joint traffic-passenger simulator P-SUMO consists of two layers: a pre-processing layer and a joint simulation layer.

- The **pre-processing layer** processes the city information and people mobility data and conducts preparation work to set up the simulation. Note that depending on the purpose, the specific design of the pre-processing layer may vary.
- The **joint-simulation layer** simulates the city-wide passenger behaviors in conjunction with SUMO and produces synthetic traffic and passenger data. In this layer, a monitor-control algorithm runs jointly with SUMO to monitor the states of the public vehicles in real time and to simulate passenger behaviors accordingly.

The simplified structure of the system is presented in Figure 2.1.

In this chapter, we develop the P-SUMO in the setting of ***bus*** transportation systems for the city of Porto, Portugal. We demonstrate the system by generating the synthetic bus passenger data from some indirect people mobility data and comparing the synthetic data to the real data. Specifically, we learn a bus passenger demand model from indirect people mobility data to

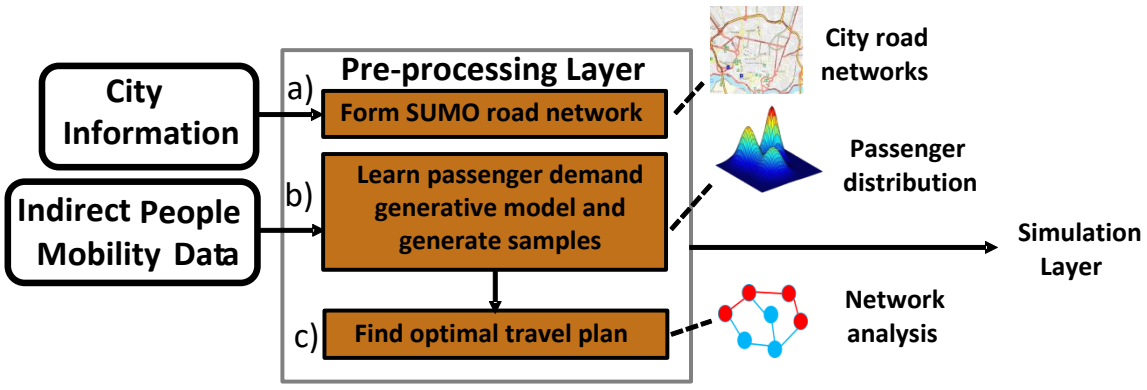

Figure 2.2: Block diagram of pre-processing layer.

generate bus passenger demand samples. The people mobility trend reflected by the indirectly related data can correlate positively, to some extent, to the mobility trend of real bus passengers. We then develop a passenger behavior model to jointly run with SUMO for generating the city-wide bus passenger synthetic data. The simulation outcomes are compared to the real bus AFC data of the same city in terms of distribution difference. We use the similarity between the synthetic data and the real data to evaluate P-SUMO.

2.2.2 Pre-Processing Layer

For the specific design of generating the synthetic passenger data from the indirect people mobility data, the pre-processing layer is a collection of three components, denoted as a, b, and c (see Figure 2.2). This layer conducts city information importing, data learning, and passenger demand generating, respectively, to prepare for the joint traffic-passenger simulation in the second layer. Specifically, component a serves to extract and convert the city road-map and bus transit infrastructures from public information resources into the SUMO format to establish a virtual city traffic network and define the traffic demands of the background vehicle flow. Component b serves to learn a passenger *trip generative model* from the people mobility data for generating the passenger *trip demand* samples. Component c models the way in which a bus passenger plans to travel from the origin to the final destination through the bus transit network (i.e., to generate the *travel plans*). Finally, this component generates the passenger *travel demand* that includes trip starting time $\mathbf{t}$, an O-D pair ($\mathbf{O}$, $\mathbf{D}$), and a travel plan $\{(\mathbf{O}, D_1),(O_2, D_2),..., (O_n, \mathbf{D})\}$.

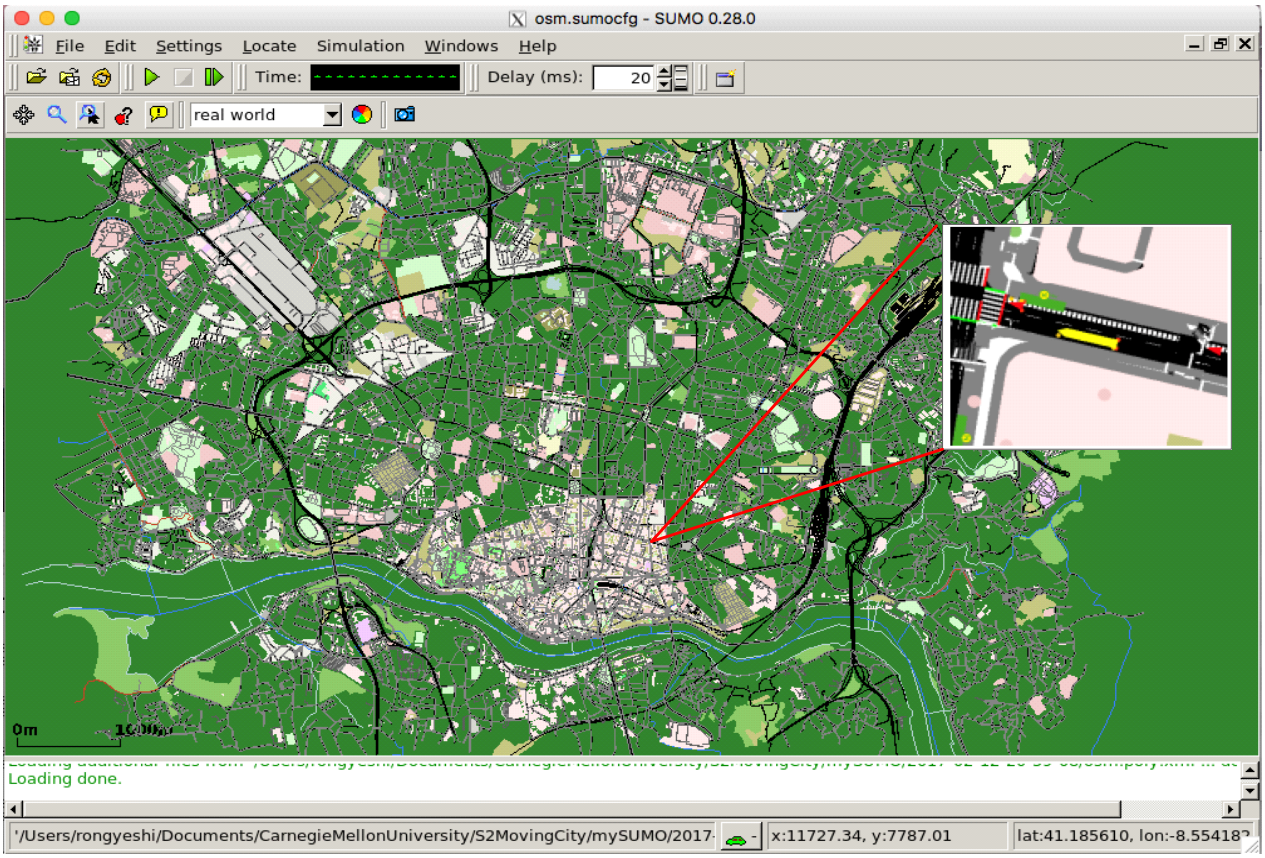

Figure 2.3: Virtual traffic and bus transit network in SUMO.

Bus Transit System Establishment

In the case study of Porto, the component *a* is to build the urban bus transit system for use in SUMO, which reflects the exact system in the real world. The main bus service operator STCP has a company website[2], where detailed information about routes, station geographical locations, and a timetable of bus trips are provided. As shown in Figure 2.3, using the STCP bus service information and other public resources (OpenStreetMap, etc.), we build the bus transit system as well as the urban road network within the selected central city area of Porto (E: -8.559543°, W: -8.661915°, S: 41.136044°, N: 41.185110°). The imported bus network contains 136 routes, 855 bus stops, and 5723 bus trips on a normal weekday. The simulation tests confirmed that the bus performance matches well with the actual Porto bus transit system: Each bus departs at the scheduled time, runs along its designated route, and pulls up at the designated stops correctly.

[2] https://www.stcp.pt

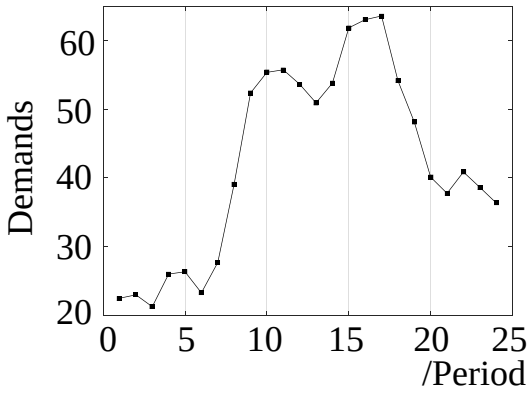

Figure 2.4: Expected taxi demands on Wednesday.

Passenger Trip Demand Generative Model

The second component is to learn the passenger trip demand generative model for generating passenger trip samples. The goal of this model is to generate the trip demand tuple (O, D, t) of a passenger. The approaches used to establish the model highly depend on the source of data available. In this implementation, we use the Porto taxi trajectory data [17] to learn the Porto passenger mobility distribution and then to generate passenger trip samples from the distribution.

The taxi dataset describes a complete year (from 7/1/2013 to 6/30/2014) of the trajectories for all 422 taxis running in Porto. Each data point contains several features in which we are interested: the 1) trip starting time; 2) date type (identifying whether the trip occurred on a holiday); 3) call type (determining whether the trip started from the taxi operation center or on a random street); and 4) poly line (storing the GPS coordinate sequence of the trip trajectory). The data with random street call types are selected, and holiday samples are removed. The dataset contains detailed trip starting times and O-D pairs of random street passengers, making it a nice resource of city dwellers' travel trends. On the other hand, the selected region is the central city area of Porto, which has quite a dense bus network, and this setting mitigates the negative correlation between taxi and bus demand models by excluding areas that are poorly served by buses.

The proposed passenger trip demand model consists of two components, a temporal model and a spatial model. The **temporal model** is an inhomogeneous Poisson process model, which is widely used to model the occurrence of events in time. We consider the Poisson process to be inhomogeneous, and thus, the *rate parameter* λ varies in time. We fit the rate parameter on

an hourly basis for a certain weekday. We divide a day equally into 24 periods and focus on studying all Wednesdays of the year. The average taxi demand in each period on Wednesday is shown in Figure 2.4, according to which we fit the estimated Wednesday rate vector $\hat{\lambda} = (\hat{\lambda}_1, ..., \hat{\lambda}_{24})$. The temporal model is then described as, in period i, the interval τ between two consecutive passenger demands follows the following exponential distribution:

$$\tau \sim f(t; a(\hat{\lambda}_i + \sigma)) = a(\hat{\lambda}_i + \sigma)e^{-a(\hat{\lambda}_i+\sigma)t}. \tag{2.1}$$

Here, a is a coefficient to scale $\hat{\lambda}$, as the number of bus demands is usually greater than taxi demands (a is used to scale daily passenger demands up to around 150 thousand, which is suggested in the STCP 2016 annual service report [18]). In practice, we also introduce uncertainty into the model by adding small noise $\sigma \sim N(0, 1)$ to $\hat{\lambda}$.

The **spatial model** is also learned on an hourly basis. The spatial model is a four-dimensional (4-D) distribution model from which we can generate 4-D samples with the first two components as origin (O^x, O^y) and the last two as destination (D^x, D^y). We apply kernel density estimation to fit the spatial model, using multivariate 4-D normal distribution as a kernel. This method is non-parametric and is effective when prior knowledge about the distribution is unavailable; thus, parametric methods do not apply well. The bandwidth is determined based on the normal distribution approximation [19]. Finally, we have the spatial model:

$$\hat{p}_H(X) = \frac{1}{n}\sum_{k=1}^{n} \frac{1}{(2\pi)^{4/2}|H|^{1/2}} \exp(-\frac{1}{2}(X - D_k)^T H^{-1}(X - D_k)). \tag{2.2}$$

where H defines the bandwidth of each dimension, and $D_k = (O_k^x, O_k^y, D_k^x, D_k^y)$ is the O-D demand of taxi demand data point k. The model generates two geographical points, and we search for the closest bus stop near each point and use it as the origin/destination stop. We set a cut-off distance of 640 meters, with the stop matching outside of this region nulled. This distance is from the public transport accessibility level (PTAL) methodology, which proposes that the longest distance a passenger would normally walk to access a bus service is within the range of an 8-minute walk at the speed of 4.8 km/h [20].

A temporal sample and a spatial sample constitute a passenger trip demand sample (Table 2.2 presents some trip demand examples). This is the core design of the passenger trip demand generative model.

Passenger ID	Origin Stop (O)	Destination Stop (D)	Trip Starting Time (t)
passenger_1244	JM2	PRG1	27000
passenger_8332	PRFL	PT3	46380
passenger_14522	CMS1	BCM5	61620

Table 2.2: Examples of trip demand samples (O, D, t).

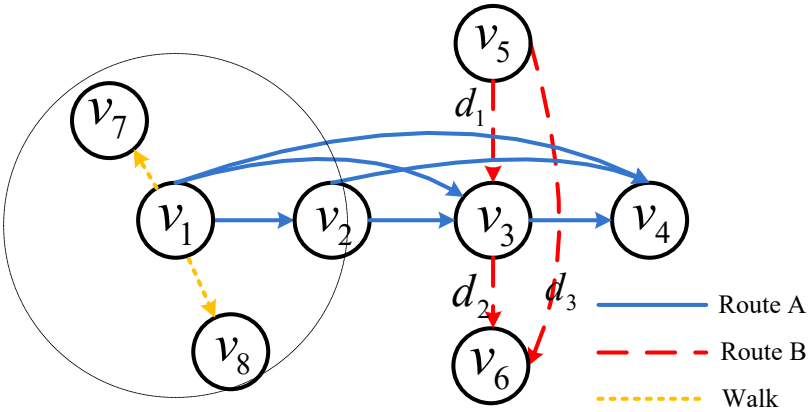

Figure 2.5: Graph of the bus transit network.

Passenger Travel Planning Model

The passenger travel planning model is to provide an answer to the following question: Given an origin bus stop and a destination bus stop (the O-D pair of a trip demand), how does a passenger *think of* a travel plan in the given bus transit network? Specifically, the model provides the set $\{(O, D_1), (O_2, D_2), ..., (O_n, D)\}$ of midway trips that lead the passenger from the origin O to the destination D to accomplish the trip demand. It is assumed that passengers always choose a plan that minimizes travel distance and unnecessary route switching, based on which we design a built-in bus passenger travel planning model.

As illustrated in Figure 2.5, we consider the bus transit network as a *directed graph* G with vertexes $V = \{v_i\}$ denoting bus stops and edges $E = \{e_i\}$ denoting the *reachability* between bus stops. For example, the blue edge from v_1 to v_2 indicates that stop v_2 is reachable from stop v_1 by taking Route A. In the graph, Route A (with blue solid edges) and Route B (with red dashed edges) share the transition stop v_3 and passengers can choose to switch routes there. Along with bus route edges, walking edges are introduced into the graph (see the yellow dotted edges in Figure 2.5). Stops within a certain geographical distance (e.g., 640 meters) are considered to be walking reachable stops, and passengers would be willing to walk a few more meters to

	Attribute	Description
Vertex	ID	Vertex identity
	Geo-location	Longitude and altitude of the corresponding bus stop on map
	Out-edges	List of edge IDs originated from the vertex
	Out-neighbors	List of vertex IDs that terminate out-edges
	Route	List of routes passing through the vertex
Edge	ID	Edge identity
	Source-vertex	The vertex ID the edge originates from
	End-vertex	The vertex ID the edge terminates at
	Weight	Length of the edge (meter)
	Route	The route the edge belongs to

Table 2.3: Vertex and edge attributes.

transfer at those stops. In each bus route, vertexes are designed to be fully connected. After establishing the bus transit network in SUMO, we can measure the exact length of each edge (i.e., the weight of each edge denoted as d). Table 2.3 lists the attributes of vertex and edge. Note that the out-edges of a vertex indicate the outward edges that originate from the vertex, and the out-neighbors indicate the terminal vertexes of corresponding outward edges. The vertex route is a set of routes in which the vertex is involved and the edge's route is the specific route to which the edge belongs. All the vertexes and edges uniquely define the whole bus transit network (graph).

Given the graph structure, a travel plan for an O-D pair consists of a set of edges, and we call each edge a *sub-trip*. We want to find an optimal plan that minimizes a certain cost objective. In addition to the cost objective in traditional shortest-path searching problems, which are based on cumulative distance only, we introduce 1) route-switching penalty Δ to penalize the route switching of a travel plan and 2) penalty ε to penalize the objective when a sub-trip (edge) is added to the travel plan. Introducing ε is beneficial: When considering the travel demand from v_5 to v_6 through Route B in the graph (with $d_3 = d_1 + d_2$), we prefer the optimal travel plan to be represented as $\{edge(v_5, v_6)\}$ rather than $\{edge(v_5, v_3), edge(v_3, v_6)\}$. By introducing ε, the plan $\{edge(v_5, v_6)\}$ with the cost $(d_3 + \varepsilon)$ will win over the plan $\{edge(v_5, v_3), edge(v_3, v_6)\}$ in terms of cost $(d_1 + d_2 + 2\varepsilon)$. Based on the structure of the graph and the definition of the cost objective, we design the optimal travel plan-searching algorithm in Algorithm 1, which is a modified version of *Dijkstra's* algorithm [21], with a more sophisticated cost objective.

Algorithm 1 initializes all distances from source to infinity, except for the source to be zero

Algorithm 1 Travel plan-searching algorithm

```
Input:  s: source vertex; d: destination vertex;
        {V, E}: graph; iniRoute: initial route of s;
        Δ: route-switching penalty; ε :adding sub-trip penalty
Output: cost, trace
1.      cost[s]← 0                                   //zero the cost of source vertex
2.      s.preRoute← iniRoute                         //preRoute used to judge route switching
3.      for all v in V − {s} do
4.        cost[v]← ∞                                 //the cost of non-source vertex is set to infinity
5.        trace.update({v:(s, infEdge)})             //initialize trace-back record
6.      endfor
7.      S← ∅                                         //S: visited vertex set
8.      Q← V                                         //Q: queue set (vertex set to be visited)
9.      while Q ≠ ∅ and d not in S do
10.       u← minCost(Q,cost)                         //select vertex u in Q with minimal cost
11.       S← S + {u}
12.       Q← Q - {u}                                 //move vertex u from Q to visited set S
13.       for v_temp in u.outNeighbors do
14.         E_out← getEdges(u, v_temp)               //examine outward edges
15.         for e in E_out do
16.           update_cost← cost[u]+ e.weight+ ε      // basic cost
17.           if u.preRoute ≠ e.route then           //judge route switched or not
18.             update_cost← update_cost+Δ
19.           endif
20.           if cost[v_temp] >update_cost then      //when basic cost improves
21.             cost[v_temp]← update_cost            //store new cost value
22.             v_temp.preRoute← e.route             //update route information
23.             trace.update({v_temp:(u,e)})         //store the trace-back record
24.           endif
25.         endfor
26.       endfor
27.     endwhile
28.     return cost, trace to d
```

(lines 1 to 6). The visited set S and queue set Q are initialized to empty and the full vertex set V, respectively, to store the visited (expanded) vertexes and non-visited vertexes (lines 7 to 8). In each iteration, the algorithm pops out and expands the lowest-cost vertex u in Q (lines 9 to 12) and investigates each of its neighbor vertexes v_{temp} (lines 13 to 26). For each neighbor v_{temp}, it examines all its outward edges e (lines 13 to 15). Then, it calculates the cost through this edge concerning the edge cost, adding sub-trip penalty ε, and the route-switching cost Δ (lines 16 to 19). The algorithm accepts the new cost when it is cheaper than the old cost, and store the

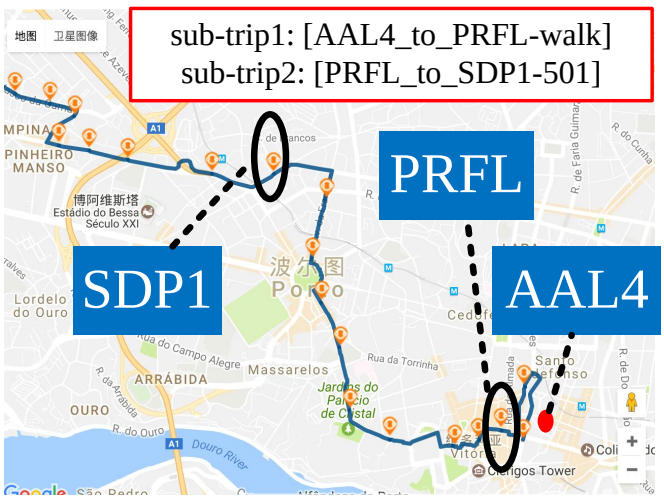

Figure 2.6: Travel plan from stop AAL4 to SDP1.

trace-back information (lines 20 to 24). This process repeats until the destination d is reached or the queue set Q is empty (line 9). The output of the algorithm is the passengers' plan to move from the origin stop to the destination stop through the bus transit network.

We apply the passenger travel planning model to generate a travel plan for each trip demand. The optimal plan-searching algorithm proves itself to be effective in practice. As shown in Fiture 2.6, given the AAL4-SDP1 trip demand, where AAL4/SDP1 is the stop ID (vertex ID) in the network, the algorithm returns a plan with two sub-trips, stating that the passenger should walk a short distance from AAL4 to PRFL and then take Route 501 to get to SDP1. Applying the trip demand generative model and the travel planning model, we can generate passenger travel demand samples on Wednesdays for any passenger size according to need.

2.2.3 Joint Simulation Layer

Given a bus passenger travel demand (containing the trip starting time, an O-D pair, and a travel plan), we need to simulate how the passenger moves through the bus transit network and interacts with buses and traffic to ultimately reach the destination. For example, one of the core functions of the joint simulation layer is to simulate the unknown boarding time t_i for each sub-trip tuple (O_i, D_i). As shown in Figure 2.7, in this layer, a bus passenger behavior model is designed and implemented to run jointly with SUMO via a monitor-control algorithm based on TraCI, a traffic control interface of SUMO. Specifically, the purpose of the algorithm is to 1) modify the conditions and states of buses, 2) simulate passenger behaviors, and 3) record important moments (bus arrival time, etc.) in real time. When conducting simulations, people

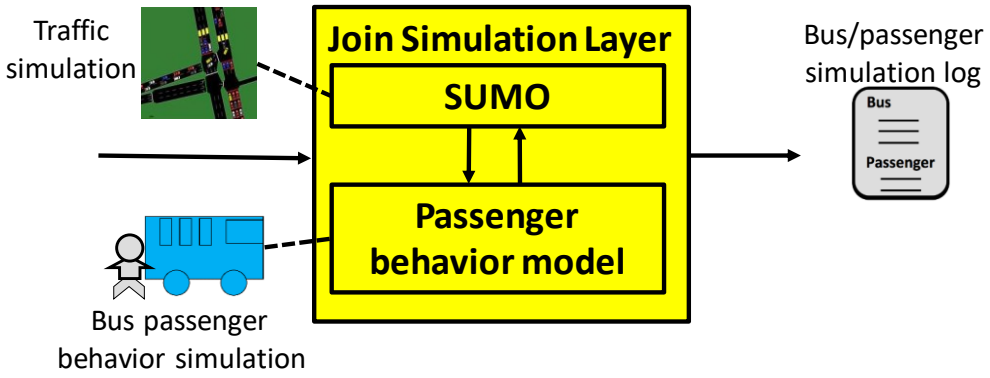

Figure 2.7: Block diagram of joint-simulation layer.

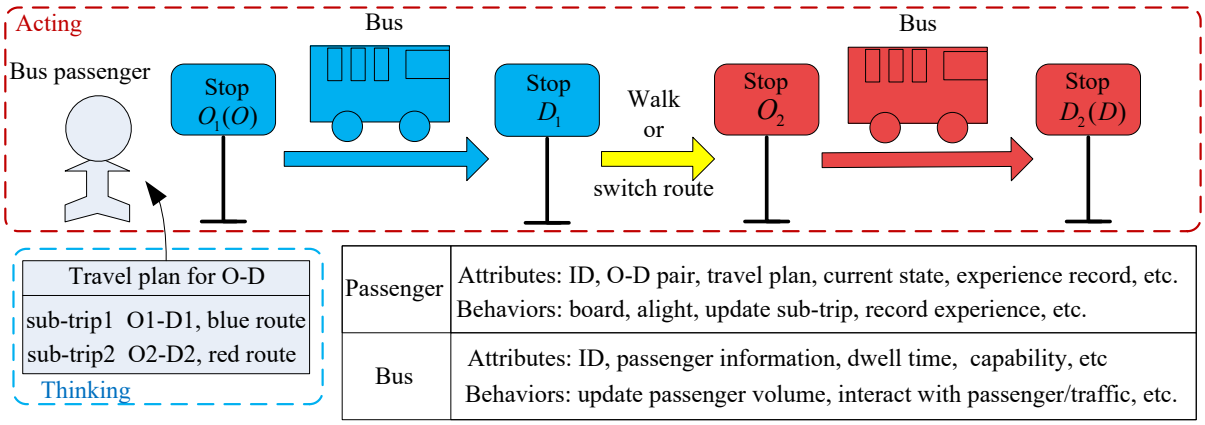

Figure 2.8: Passenger behavior model.

always make a trade-off between simulation speed and granularity depending on research topics. In most cases, to gain macroscopic insights into city-scale passenger behavior patterns, a reasonable strategy is to begin with a basic model, and gradually increase the subtlety according to computing capacity and needs.

In the joint simulation layer of P-SUMO, the traffic simulator SUMO and passenger behavior model communicate with each other to modify and update the states of the passengers and buses in real time. There are influences from SUMO to the passenger as the bus arrival events trigger the passengers' actions to get on and off, and the states of relevant passengers are updated. Also, there are influences from the passenger behavior model back to the SUMO as passengers' on/off actions affect the bus dwell time at the stop and thus the traffic conditions on the roads.

In our work, we introduce a bus passenger behavior model. Figure 2.8 illustrates this model. According to the travel plan, the passenger starts at the origin stop O_1 and takes a bus on the blue route to D_1 to complete sub-trip 1. Then, he or she gets to O_2 by walking (if $D_1 \neq D_2$)

or route switching (if $D_1 = D_2$) to start sub-trip 2. Finally, the passenger gets to the final destination D_2 via the red route, and the travel demand is fulfilled. Interactions between buses and passengers take place at each stop, where the number of boarding and alighting passengers affects the bus dwell time. According to STCP public vehicle descriptions, most buses in Porto have independent channels for boarding and alighting, respectively, and thus, the dwell time t_{dwell} is formulated as $t_{dwell} = max(t\_on, t\_off)$, which is the maximum of the boarding time t_on and alighting time t_off. SUMO simulates the interactions between buses and traffic, where the travel time t_bus varies according to traffic conditions on the roads. The time the passenger spends from stop i to stop $(i+1)$ is

$$t_i = L + max(t\_on_i, t\_off_i) + t\_bus_i, \tag{2.3}$$

where L is a constant of lost time, including the pulling up, door-open-close time, etc. With this model, the simulation captures primary interactions among passengers, buses, and traffic. All bus passengers in the city are treated as agents who follow both the travel planning model and the behavior model defined in previous sections.

Loading the passenger travel demands to the joint simulation layer, we use P-SUMO to simulate the city-wide bus passenger behaviors in Porto for 90 Wednesdays. For each day, the simulation log stores detailed passenger behavior information and bus state information. For example, the passenger log records the time spent waiting at the stop, boarding, and alighting at the destination stop. The bus log includes the bus arrival time at each stop, on/off passengers' IDs at each stop, stop dwell time, and passenger volume after prompting passengers to get on/off.

Some examples of the simulation output log are provided in Figure 2.9. The bus operation log includes the transit operation details. From the example of one bus trip on the Route 23, we can extract the dynamics of seat availability on the bus. Furthermore, we obtain the bus arrival time at each stop, the number of on and off passengers, and the time the bus stays at the stop. For example, the first row in the bus log example indicates that at the time 41409, the bus arrives at Stop C24A4 and 50 seats are available on the bus. There are no passengers getting on and off and the dwell time of the bus at the stop is 1 second.

Another log is the experience log of Passenger 131. One of the passenger's properties is the travel plan, which is generated by the pre-processing layer. The experience of the passenger is another important property that provides the timing of each sub-trip (i.e., when the passenger

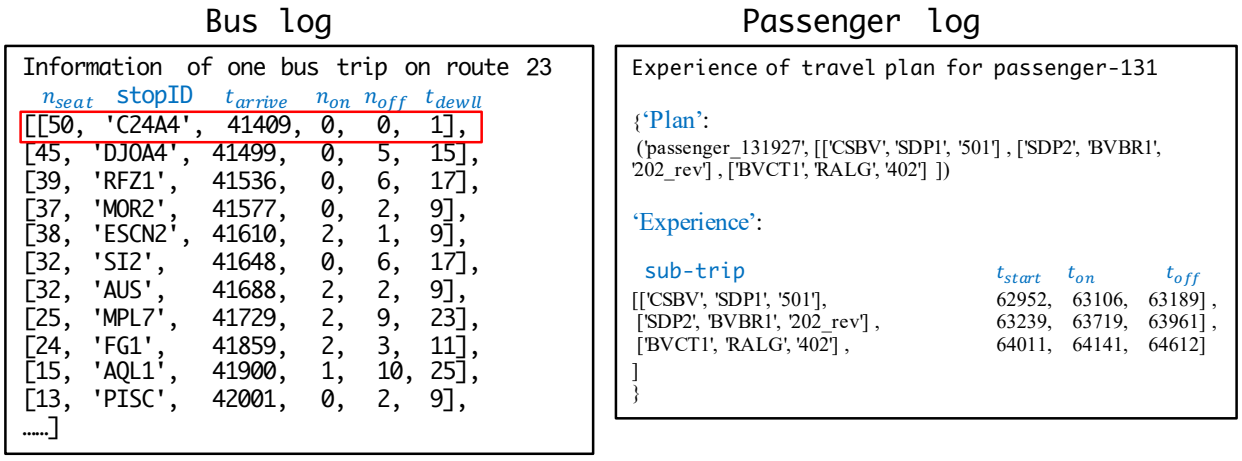

Figure 2.9: Examples of simulation output log.

starts waiting for the bus at the stop and when he or she actually gets on the bus to start the trip).

The bus log along with the passenger log constitute the simulation output log, serving as the synthetic data for post-processing.

2.2.4 Evaluation of Synthetic Data with Real Passenger Data

In order to evaluate P-SUMO, we compare the synthetic bus passenger simulation data to the real bus AFC data collected from Porto. The basic idea is to quantify the difference between the synthetic data and real passenger data in terms of spatial-temporal distribution, using the *Kullback-Leibler* (KL) divergence [22].

Real AFC Bus Passenger Data

The AFC dataset is the set of bus passenger transaction records that occurred in January, April, and May of 2010. They are collected by the AFC system installed in buses operated by STCP in Porto. The AFC system called "Andante" is an entry-only system – a transaction record is generated once a passenger taps the travel card on his or her AFC reader. No additional action is needed when the passenger alights, so the information about destinations is unavailable.

Each transaction record contains several attributes in which we are interested: the 1) ID, 2) transaction timestamp, 3) bus stop at which the transaction occurred, 4) route, and 5) route direction. We fuse the Andate AFC data with additional data sources to obtain the route structure (sequence of stops in a route) and geographical location of each stop. There are 2,374 bus stops and 66 bidirectional bus routes within the area of interest. The raw data have about 3% fault

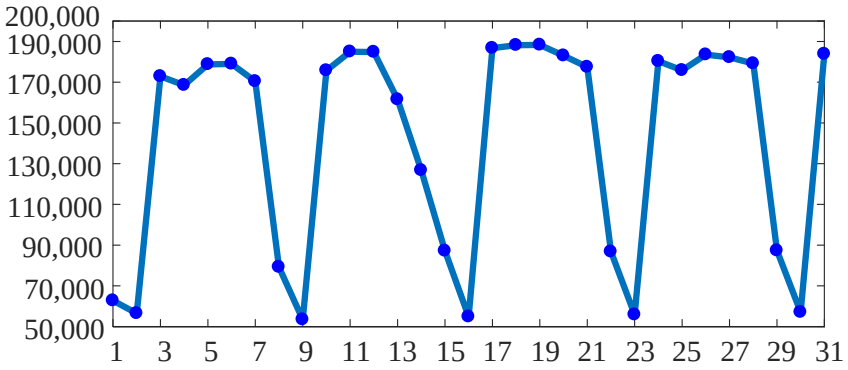

Figure 2.10: Number of transactions vs. date in May for AFC data.

samples that have illogical or missing attributes. After data recovery, a proportion of 1% remains unsolved, and we remove those records.

Figure 2.10 shows a weekly repetitive pattern in passenger boarding activity in a month, based on which, it is reasonable to assume that the passengers' mobility trend during certain normal weekdays is consistent. We select the transactions on Wednesdays of the three months, totaling 12 Wednesdays with 2,422,079 records. Those Wednesdays are normal weekdays. Special local holidays are avoided.

Evaluation of Temporal Distribution

The goal of the simulation is to capture the underlying distributions from which the real observations are generated so that the simulation outcomes can be used as a reasonable approximation of the real passenger data. To this end, we quantify the difference between the synthetic passenger data and the real AFC data, and verify our method by comparing the synthetic-real data difference to the data difference of baseline methods (not using the simulation) from the real data. The measurement applied to quantify the difference between two distributions is called the KL divergence [22]. For discrete distributions, the KL divergence from distribution $Q(i)$ to distribution $P(i)$ is defined as:

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}. \tag{2.4}$$

The use of the KL divergence in measuring the difference between probability distributions is popular for the following reasons: 1) $D_{KL}(P||Q)$ is always non-negative; 2) $D_{KL}(P||Q) = 0$ holds if and only if $P = Q$; and 3) $D_{KL}(P||Q)$ is a convex function with respect to P and Q. The

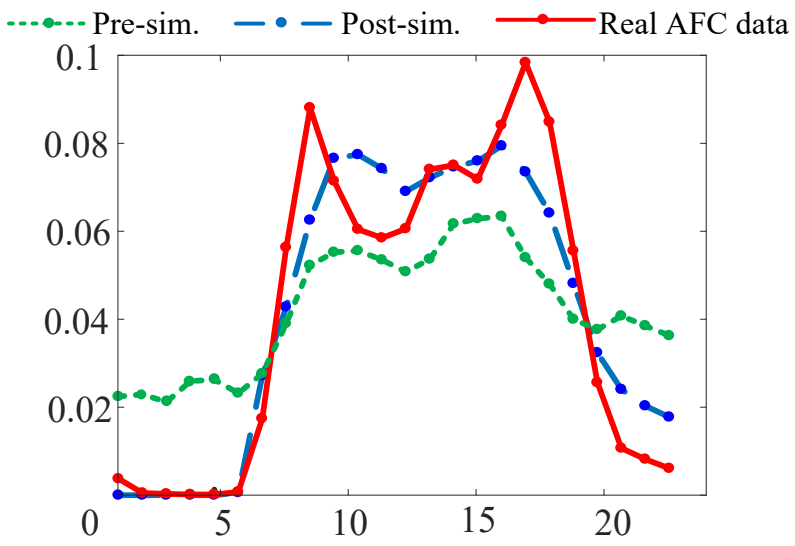

Figure 2.11: Demand occurrence probability vs. hour in a day.

convexity implies that the larger the $D_{KL}(P||Q)$ is, the larger the difference will be between P and Q.

We investigate the difference in *temporal passenger demand distributions* between the simulation data and the real data. Here, P and Q are the expected *temporal* passenger demand distributions for simulation data and real data, respectively. In this thesis, we focus on Wednesday data. Specifically, the distribution P is defined as $P(i) = \mathbb{E}(n_i) / \sum_{j=1}^{24} \mathbb{E}(n_j)$, where n_i is the number of passengers who get on a bus in period i (e.g., 10 am-11am), and $\mathbb{E}(n_i)$ is the average number (over all 90 simulated Wednesdays) of passengers who get on a bus in period i. In the same way, we can obtain the distribution Q for the real data. The shapes of both distributions are illustrated in Figure 2.11, where we can see a clear similarity between the simulation data (blue dashed) and the real data (red solid).

For comparison purposes, we consider two baseline distributions. The first one is simply *shuffled* from P, and we call it *shuffled distribution*. The shuffle means a random permutation of $P(i)$ with respect to period i. The second baseline distribution is the temporal distribution estimated directly from the taxi passenger data, which is the **green dotted** curve in Figure 2.11. We call such a distribution the *pre-simulation distribution* because the data have not been processed by the simulation. Note that only the first *trip demand starting time* t (see Table 2.1) contributes to the pre-simulation distribution. In contrast to the two baselines, P is called the *post-simulation distribution* (the **blue dashed** distribution in Figure 2.11). Be aware that both the first trip starting time t and the simulated timing $\{t_i\}$ of midway trips (i.e., sub-trips) contribute to the post-simulation

distribution.

Calculating the KL divergence from the real distribution Q to each of the three distributions, we obtain **1.418, 0.553, and 0.045** for shuffled, pre-simulation, and post-simulation distributions, respectively. Compared with the shuffled and pre-simulation distributions, the post-simulation distribution achieves the best similarity, reducing the distribution difference to 0.045. The information gain comes from the fact that the simulation can capitalize on the passenger behavior model to effectively fill the midway details (especially the timing $\{t_i\}$ of each midway trip) between the origin and the destination. In contrast, the raw taxi data only provide the O-D pair of the whole trip, failing to provide the midway details (especially the *timing* t_i of the occurrence of each event in the experience).

Evaluation of Spatial Distribution

We further investigate the difference in spatial distributions. Because the real data contain only boarding information, we should conceptualize the spatial distribution accordingly: Because a bus route R consists of a sequence of bus stops $\{s\}$, and because each stop corresponds to a spatial location, the spatial probability distribution of passenger boarding demands associated with route R is essentially the boarding probability distribution over bus stops $\{s\}$. Considering that the spatial distribution can vary in different periods, we focus on the periods of $\{T\} = \{$4-8, 8-12, 12-16, 16-20, 20-24$\}$ and omit the period of 0-4 because buses are mostly off-service during that time. Then, for simulation data, given period T and route R, the spatial distribution over $\{s\}$ is defined as $P_{R,T}(s) = \mathbb{E}(n_{R,T,s}) / \sum_k \mathbb{E}(n_{R,T,k})$, where $n_{R,T,s}$ is the number of passengers (on a Wednesday) who get on a bus in period T at stop s of route R, and $\mathbb{E}(n_{R,T,s})$ is the average number (over all 90 simulated Wednesdays) of passengers who get on a bus in period T at stop s of route R. In the same way, we can obtain $Q_{R,T}(s)$ for the real data. The spatial KL divergence for the period T is defined as:

$$D_{KL\_spatial}(T) = \mathbb{E}_R[D_{KL}(P_{R,T}||Q_{R,T})] = \frac{1}{N_R}\sum_R D_{KL}(P_{R,T}||Q_{R,T}), \tag{2.5}$$

where N_R is the number of routes in the area being investigated. This is the expected KL divergence in spatial distributions over all bus routes during a certain period T.

Based on Eq. 2.5, the spatial KL divergences from the real spatial distribution to the shuffled, pre-simulation, and post-simulation spatial distributions are calculated and illustrated in Figure 2.12. Note that the spatial pre-simulation distribution counts only on the trip demand

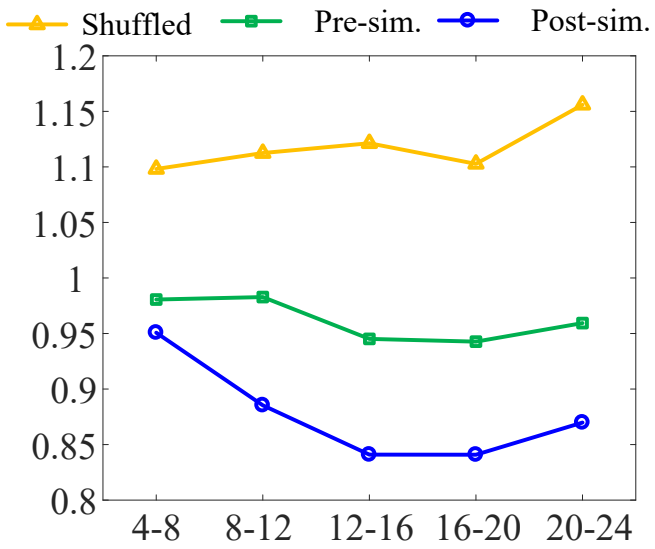

Figure 2.12: Spatial KL divergence $D_{KL\_spatial}$ vs. time period T.

(O, D, t) of each passenger; in contrast, the spatial post-simulation distribution counts on the whole synthetic experience $\{(\mathbf{O}, D_1, \mathbf{t}), (O_2, D_2, t_2), ..., (O_n, \mathbf{D}, t_n)\}$ of each passenger. From the figure, we can observe that from shuffled to pre-simulation and then to post-simulation distributions, there is a decreasing trend in the divergence. The experimental results support that the post-simulation data exhibit a higher degree of similarity to the real bus passenger data in terms of spatial and temporal activity. This experimental outcome also supports that the P-SUMO is an effective system that simulates the interactions between passengers and traffic, making it a reasonable platform for the passenger on-vehicle experience simulation.

2.3 Modeling the On-Vehicle Wi-Fi Service

The on-vehicle services are termed public *resources* because they are limited in amount and used by each individual passenger to improve his or her experience. This thesis considers two public resources: on-vehicle Wi-Fi service and space availability. The modeling of the on-vehicle Wi-Fi service is driven by the following concerns: In a city, buses can connect to any wireless backhaul channel to provide Internet connectivity to passengers, including cellular towers and city hotspots. The connection based on cellular networks is expensive because they are owned and operated by telecommunication companies (e.g., T-Mobile, AT&T). Those companies have to build and maintain the infrastructures, auction against peers to gain a spectrum license from the Federal Communications Commission (FCC) and support many other costs. In contrast, urban hotspot-based infrastructures are public facilities, and the wireless connection between a

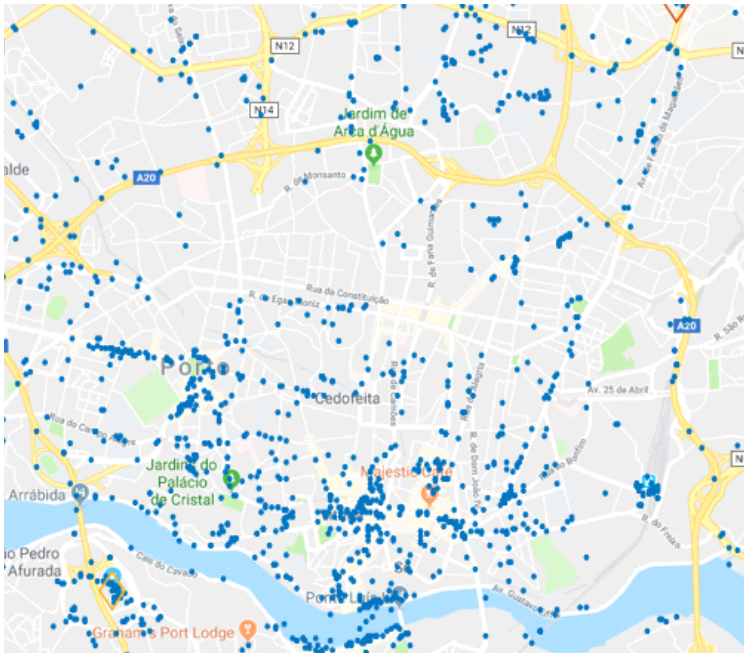

Figure 2.13: Map of Wi-Fi hotspot locations in Porto.

bus and a hotspot is free. The cost, which includes Ethernet fees, hotspot maintenance, etc., is much cheaper, and is affordable to urban governance. In this sense, in the thesis, the on-vehicle free Wi-Fi connection is assumed to be supported by the city hotspots deployed across the city and passengers are assumed to seek the bus Wi-Fi service first before using their own cellular data packages.

We construct the on-vehicle Wi-Fi service map based on the urban hotspot location data and the free-space-path-loss model; each bus will connect to the closest hotspot automatically (ignore the handoff delay between Wi-Fi hotspots). Without considering the surrounding obstacles, multipath interference, and other factors affecting the wireless signals, the capacity of a link in bits per second (bps) follows *Shannon's* Theory [23]:

$$C = B \cdot log_2 \left(1 + \frac{S}{N}\right), \tag{2.6}$$

where B = 40MHz for 801.11ac wireless networking protocol [24], the signal-to-noise-ratio (SNR) = 40dB for the good outdoor environment 2.5 meters from the hotspot [25]. The SNR is linear to $\frac{1}{distance^2}$ based on the free-space-path-loss model.

We use the hotspot location data to form the map of the on-vehicle Wi-Fi service. Specifically, for the city of Porto, the collection of hotspot location data is crowdsourced by mobile phone

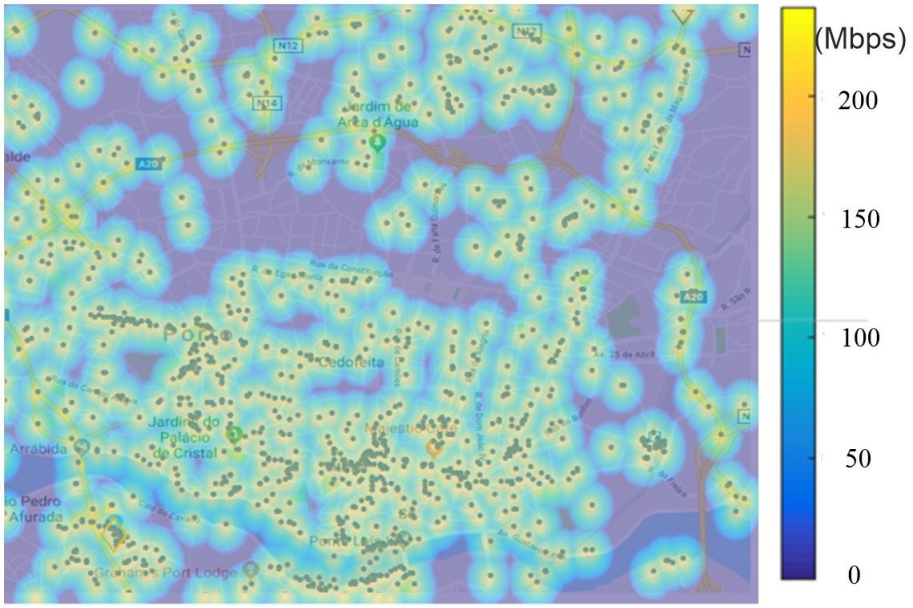

Figure 2.14: Map of the on-vehicle Wi-Fi capacity in Mbps.

apps, such as Wiman[3], through which users can share their free Wi-Fi hotspot location information to the apps. This way, the apps can help other users to automatically connect to the hotspot when they reach the covered area. The real Wi-Fi hotspot location data for Porto is presented in Figure 2.13, where demographically dense areas usually concur with dense Wi-Fi hotspot deployments. Figure 2.14 is the map of the on-vehicle Wi-Fi service capacity using Eq. 2.6, where we assume that the capacity is not affected by whether two or more buses are connecting to the same hotspot. We observe that the capacity decays with respect to the distance from the hotspot and that the areas with higher hotspot distribution density will provide higher and more robust quality of Wi-Fi service (up to 200 Mbps).

Combining the P-SUMO and the on-vehicle Wi-Fi service model, we record the simulated capacity of the Wi-Fi access rate (Mbps) on each bus trip and add the records to the simulation output log. Specifically, at every 10 seconds in the simulation of a Wednesday, we associate the geographical location of each bus trip to the closest Wi-Fi hotspot and calculate the Wi-Fi access rate. This capacity becomes an attribute of the bus state record, and the average Wi-Fi access rate allocated to each passenger on the same vehicle becomes an attribute of the passenger's experience record.

Figure 2.15 illustrates how the average on-vehicle Wi-Fi service allocated to each passenger is retrieved based on the simulation. We sample the Wi-Fi capacity along Trip 801-1, which

[3] https://www.wiman.me/portugal/free-wifi-porto

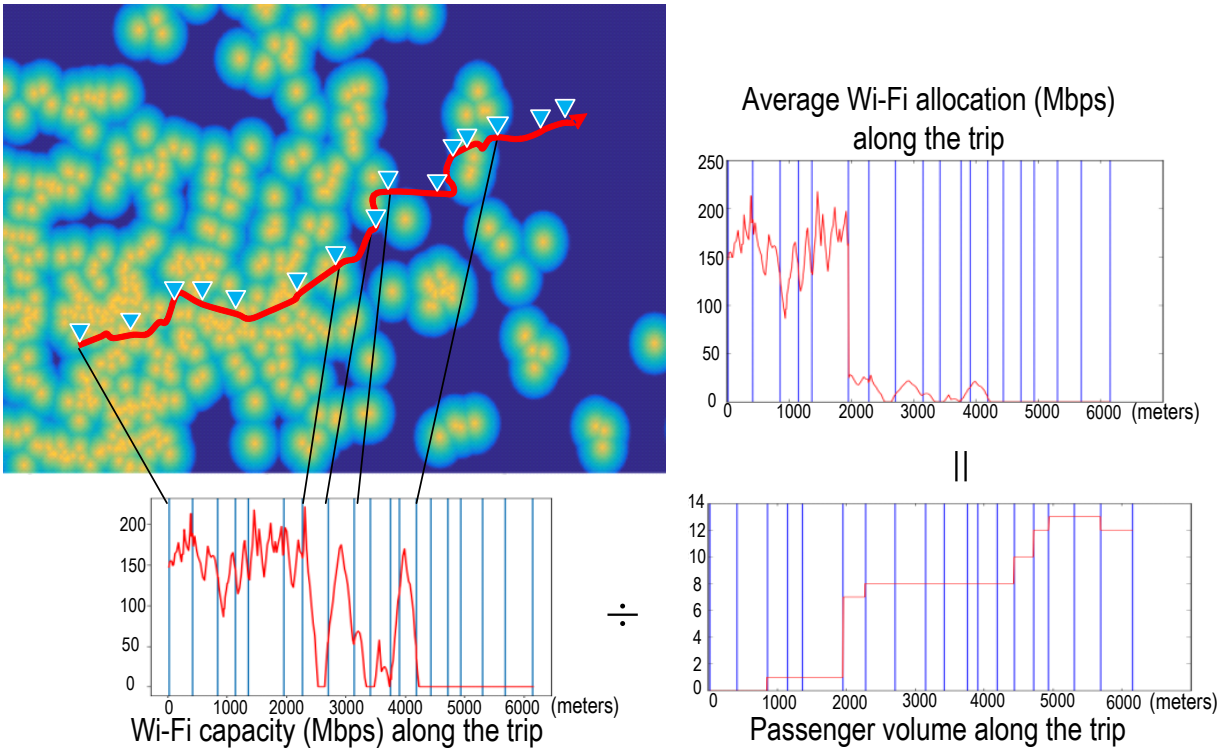

Figure 2.15: Example of the average Wi-Fi quality profile along Trip 801-1 on the Wednesday of May 5, 2010.

departed at 6 am on the Wednesday of May 5, 2010; collect the passenger volume; and take the ratio of the two. This trip starts from the central city area, and the Wi-Fi quality is high. However, at around 2000 meters into the trip, the passenger volume increases and the average Wi-Fi quality declines to around 25 Mbps. After 4000 meters, the bus moves into the poor Wi-Fi area while the passenger volume is large, and thus, the Wi-Fi service is low from the viewpoint of each passenger. Because the passenger volume will change from day to day, the average Wi-Fi profile is unique on a daily basis. However, instead of complete randomness, human trajectories show a high degree of temporal and spatial regularity [8], and thus, the Wi-Fi quality pattern along Trip 801-1 may still apply to a future normal Wednesday. Based on this dynamic pattern of the Wi-Fi quality on Trip 801-1, a planner could encourage the client passenger to take the segment of the trip before 2000 meters and avoid traveling through the segment after 4000 meters.

SUMO and the passenger behavior model, extended by the on-vehicle Wi-Fi service model, constitute the joint traffic-passenger simulator P-SUMO that enables on-vehicle experience simulation at the passenger level. The P-SUMO generates the synthetic data necessary for modeling the city dynamics as a prerequisite of our mobility planning research.

BusID	Stop	Time (mins)	Pre_vol	Target
207-12	6	498.13	5	7
207-17	10	554.95	16	20
207-15	2	529.97	5	10
207-12	9	500.55	10	12

Table 2.4: Feature and target of 4-D input data.

2.4 Usage of Synthetic Data

In this section, we demonstrate the application value of the P-SUMO by showing how researchers can rely on the synthetic data to move forward in the absence of complete data in the field of passenger-related research. As part of the IPTS research, people would like to apply ML approaches for predicting the passenger volume on buses. However, in practice, the developed approach is usually not verified due to the lack of the ground-truth validation data. The synthetic data generated by the P-SUMO can solve the lack-of-complete-data issue.

In this demonstration, we train an artificial neural network (ANN) using the synthetic data for predicting the bus passenger volume dynamics in real time. The hurdle to applying such a paradigm in practice is the unavailability of the real passenger volume information. The volume records in the simulation output log provide an opportunity to make a proof-of-concept of the ANN-based prediction.

2.4.1 Training and Testing Data Preparation

We extract the bus state information from the P-SUMO simulation output log and divide it into a training set (75%) and a testing set (25%). Before actually implementing an ANN, we first decide which features to use as the input. In this experiment, we use two versions of input. The first version (3-D input) has three features $X = [BusID, Stop, Time]$, where $Time$ is the bus arrival time at the stop. The second version (4-D input) has four features $X = [BusID, Stop, Time, Pre\_vol]$, where Pre_vol is the passenger volume at the previous stop. The passenger volume at stop i is defined as the number of on-bus passengers at the moment the bus leaves the stop. The 4-D version examples extracted from the simulation log are illustrated in Table 2.4, where the last column is the target value Y we want to predict. The data point in the first row means that the bus trip "207-12" arrives at the stop "6" at the time of 498.13 minutes (about 8:30 am), and the passenger volume changes from 5 to 7 after prompting passengers to get on and off. The ANN

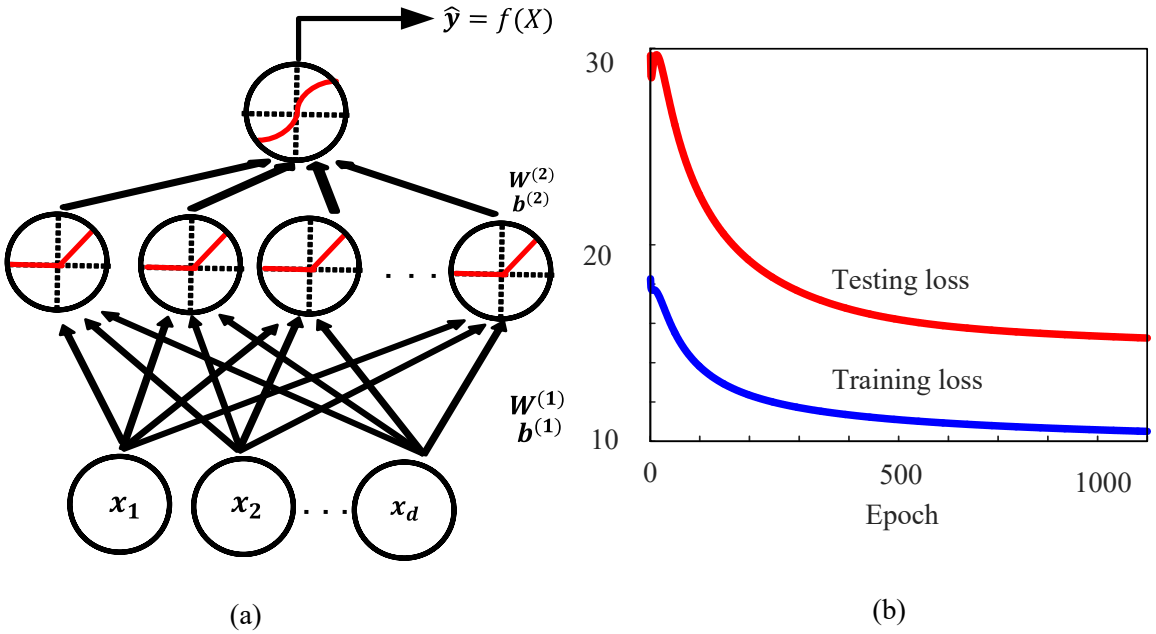

Figure 2.16: ANN structure and learning results.

aims to predict the volume 7 given the information in the first 4 columns.

2.4.2 Neural Network Implementation and Performance

We apply a three-layer neural network and the hidden neuron number is tunable, as shown in Figure 2.16 (a). The activation function of the hidden layer is the *ReLU* function: $g(a) = ReLU(a) = max(0, a)$. The output layer has a sigmoid activation function: $g(a) = c \cdot \sigma(a) = c/(1 + e^{-a})$. Here, c is the bus capacity. We have c = 60 in the simulation. The selection of the activation functions is based on simplicity and training efficiency. The derivatives of both *ReLU* and sigmoid are easy to solve, and the hidden *ReLU* layer can prevent the *vanishing gradient* issue when we increase the depth of the network.

We use the square error loss, $l(f(X), y) = \frac{1}{2}(y - f(X))^2$, for back-propagation training. We apply the stochastic gradient descent method to find the weights W and bias b that minimize the loss objective. The learning result of the 3-D input and 200-hidden-neuron ANN (training over 1000 epochs) is shown in Figure 2.16 (b), where we observe a consistent decrease in the average loss of both training and testing sets as training proceeds. We implement four ANNs with different input versions and hidden sizes. Their performance is listed in Table 2.5. The average prediction error in the fourth column indicates the average difference between the target value of passenger volume and the corresponding ANN prediction on the testing set. The average

NN ID	Input	Hidden neurons	Ave. error
NN1	3-D	200	±5.58
NN2	4-D	200	±2.83
NN3	3-D	1000	±4.74
NN4	4-D	1000	±2.76

Table 2.5: ANN structure and average prediction error.

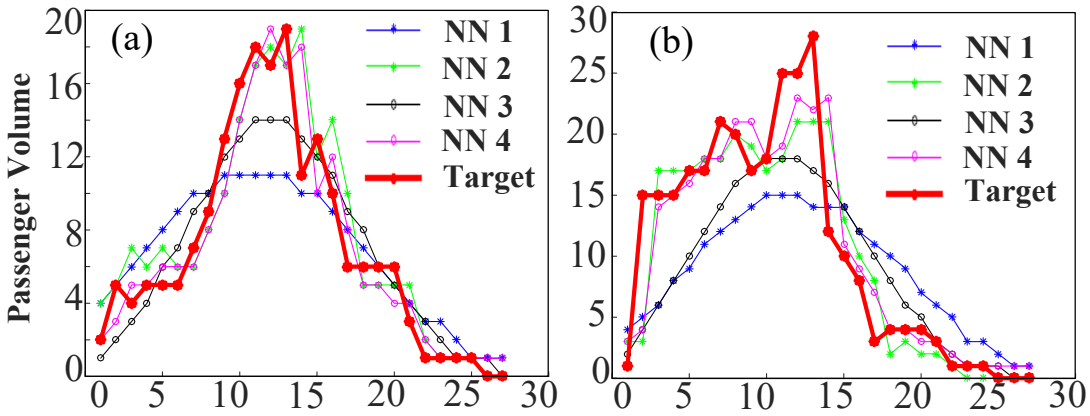

Figure 2.17: Passenger volume prediction at each stop: Trip (a) 207-21 (b) 207-35.

error decreases as the hidden size and input dimension increases.

Figure 2.17 presents the prediction for bus trips 207-21 and 207-35 to visualize how the prediction performs for the two bus trips on a future Wednesday. With 3-D input (NN1 and NN3), the prediction can capture the shape of the ground-truth curve. We observe that the large hidden size improves the prediction, but not significantly. This is because the information provided by the 3-D input is insufficient for an accurate prediction. In contrast, NN2 and NN4 can improve their prediction by using the latest passenger volume information and eventually fit well with the irregular changes of the day. The experiments verify that structures with 4-D input usually lead to a significant improvement in prediction.

The experimental results demonstrate the feasibility and effectiveness of applying ANNs to passenger volume prediction. The daily pattern of the passenger volume dynamics is a valuable reference for city planners to improve the bus transit systems and their infrastructures. For example, to make a strategic arrangement of service resources (e.g., on-vehicle Wi-Fi bandwidth assignments) for Trip 207-21, more service resources should be distributed to the bus trip between Stop 5 and 20 as large passenger volumes are predicted. For a real-time passenger volume-monitoring purpose, under a condition of limited funding or limited number of sensor

devices, deploying more sensors to Trip 207-35 is a more reasonable choice than 207-21 because more uncertainties are predicted.

In this demonstration, using synthetic data, the ANN-based bus passenger volume predictor has been effectively verified as it can perform decently on the synthetic passenger data, making it a potentially powerfully tool for real-world prediction.

2.5 Summary

In this chapter, we introduce the joint traffic-passenger simulator, P-SUMO. P-SUMO integrates SUMO, a passenger behavior model, and an on-vehicle Wi-Fi service model for simulating the passenger's on-vehicle experience. We develop a joint simulator for the bus transit system in Porto, Portugal.

The synthetic bus passenger data has high similarity in terms of spatial-temporal distributions to the real-world data collected by the AFC system in the same city. We show the application value of the synthetic passenger data in a demonstration of predicting the passenger volumes in real time.

Chapter 3

Infer the Missing Features in Incomplete Data

For the on-vehicle experience simulation, the specific design of the pre-processing layer in Figure 2.1 should configure the passenger flow based on real data. The Porto Andate AFC data are a valuable resource that is directly suitable for such a configuration. Unfortunately, the AFC system collecting the passenger data is entry-only, and the unavailability of destination information limits the data's use in specifying the passenger flow in the simulation. This chapter describes our contribution to this issue, which is also a long-standing challenge in the O-D matrix estimation field. To find the passengers' destinations, traditional approaches rely on heuristics that can match a limited proportion of the missing features. To fill the remaining missing features, our work proposes a semi-supervised learning methodology, called *Self-Training*.

The purpose of self-training is to address the destination inference problem for the remaining unmatched data by taking advantage of the heuristic-based matched data. We formulate the problem as a semi-supervised learning problem and leverage the self-training methodology to infer the remaining missing destinations. As part of our contribution, the proposed method is evaluated using two data sources: the simulation log generated by the joint traffic-passenger simulator introduced in Chapter 2 and a subset of the heuristic-based matched AFC data. Experiments are designed to compare our approach to baseline methods in terms of inference performance. Much of the work presented in this chapter was also presented in [26].

3.1 Motivations and Challenges of Filling the Missing Features

To infer the missing features in real data is important because many data recording systems only save the start of a process, not the end.

3.1.1 Destination Inference for Passenger Data

The real passenger data accessible to our research have the missing-feature issue. In general, the data collected by AFC systems and many other automatic data collection (ADC) systems are incomplete because these systems are entry-only. As shown in Figure 3.1, the entry-only system only requires passengers to use their transaction cards and identify themselves when entering a public vehicle; no action is needed from passengers when they alight (i.e., getting off). Entry-only systems are popular because they reduce the number of transaction devices that must be supported, and no additional action is needed when passengers leave. As a result, the passenger transaction data collected by these systems do not provide passengers' destination information, limiting direct use of the data to construct the O-D matrices important for the scheduling and adjustment of public transit service.

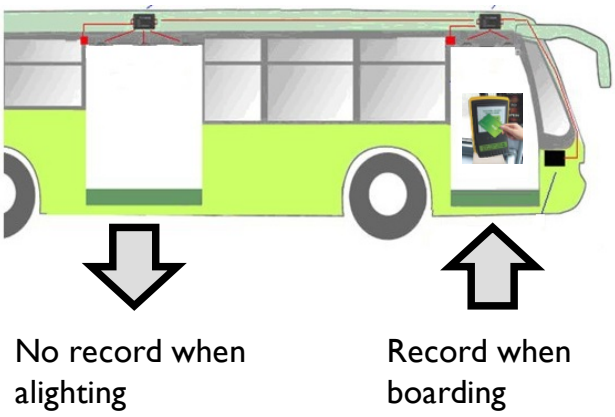

Figure 3.1: Entry-only automated transaction system.

Passenger O-D matrices provide important information for operation scheduling and service adjustment [5]. Generally, the O-D matrices of passengers' journeys can be obtained in four ways:

- The first is to conduct manual travel surveys, which are infrequent, expensive, and prone to response bias [27].
- The second way is to build entry-exit systems on all mass transit vehicles in the city to accurately record boarding and alighting behaviors for each passenger. As mentioned previously, for internal mass transit in a city, there is no simple solution to applying such systems due to hardware costs and passenger inconvenience. In fact, only a few cities are equipped with an AFC system that can provide passenger information about both boarding

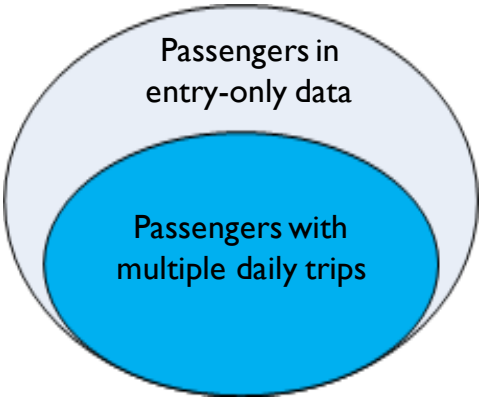

Figure 3.2: Traditional methods can apply to passengers who have multiple daily trips.

and alighting locations. To the best of our knowledge, the first passenger-related research using complete entry-exit AFC data was conducted by Jung *et al.* [28] in 2017. In their work, they trained deep artificial neural networks with a large amount of O-D trips provided by the entry-exit bus fare system in Seoul, Korea and predicted destinations given the entry information. Though this method is a decent starting point to applying the ML method to passenger data, its generality to other cities is limited due to the lack of funds for building a destination collection system.

- As a workaround, the third way is to develop a passenger behavior simulator to generate simulation data. As seen in Chapter 2, the traffic-passenger joint simulator P-SUMO has been developed. This work 1) learned a passenger demand model from *indirect* people mobility data, 2) developed a passenger behavior model to jointly run with SUMO, and 3) generated synthetic passenger data for researchers to work with. The generated data can be used – to a limited extent – to justify potential ML applications to passenger-related research. However, without being driven by direct and complete real passenger data, the simulation results may deviate from the real world, limiting its referential value to researchers.

- The fourth way is to estimate O-D pairs from either incomplete but direct data or indirect but relevant data, such as cell phone data [2] and parking-sensing information [29]. Due to practical efficiency, the fourth path becomes a necessary intermediate step in most public transit analysis [30]. Over the past two decades, estimation of O-D matrices directly from entry-only passenger data has attracted substantial research interest. Traditional approaches rely on heuristics to estimate passenger destination information. Specifically, as

shown in Figure 3.2, a passenger might have multiple trips in a day, and the next trip constrained by time, distance, and other validation rules lets us match a destination for that passenger. The work of Nunes *et al.* [15] estimates the destinations based on two heuristic assumptions: 1) the most likely destination of the trip is the origin of the next trip, and 2) the most likely final destination is the origin of the first trip of that day. They constrained their estimations by transit distance, travel zone limitation, and other validation rules to ensure accuracy. However, there might still be a considerable amount of entry-only data in which we are not able to match the destinations, either because we do not have a subsequent trip or the subsequent trip does not meet the validation rules. In general, the remaining entry-only data that cannot be matched using heuristic approaches (e.g., the gray area in Figure 3.2) account for 40-50% of the total entry-only data.

Thanks to previous research, the heuristic-based matched data cover the majority of the passenger population with reasonably high accuracy. There is useful information in the matched data with which the remaining unmatched data can be inferred, and it is feasible to further improve the inference of the remaining data based on the knowledge obtained from the heuristic-based matched data. The rest of this chapter details our solution to the destination inference. We use Nunes *et al.*'s approach [15] to match some of the entry-only data and infer the remaining unmatched data using our proposed method.

3.2 Heuristic Method

The heuristic method estimates destinations directly from the entry-only data. We refer to the destination estimation using heuristics as *destination matching*. This section introduces a commonly used heuristic method upon which our work is based.

For general discussion, the entry-only passenger data have the following four attributes:

- t boarding time,
- r route code,
- o index of the boarding stop, and
- h passenger ID.

We also conceptualize:

- d index of the alighting stop, and
- n^r number of stops of route r.

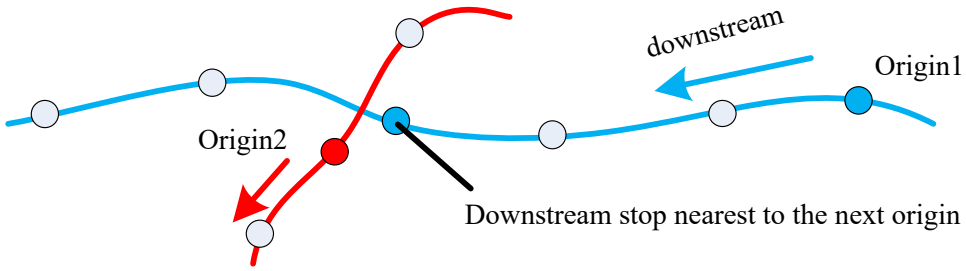

Figure 3.3: Heuristic assumption example.

The heuristic method is conducted based on passengers' travels that have multiple stages in a single day. This implies that the data are pre-grouped by date before applying the heuristic method. Unless otherwise noted, the data in this section are assumed to be same-day. The objective of matching the destinations of passengers' travels is to determine the alighting stop of each travel stage:

$\hat{d}_{h,s}$ index of the matched alighting stop of the s-th stage of passenger h.

The heuristic method applied in this paper is based on two key assumptions: 1) the most likely destination of a travel stage is the downstream stop nearest to the origin of the next travel stage (see Figure 3.3), and 2) the most likely destination of the final travel stage is the downstream stop nearest to the origin of the first daily travel stage.

Let $dist[stop1, stop2]$ be the Euclidean distance between two stops. Define

$$A_{h,s} = \{i | o_{h,s} \leq i \leq n^{r_{h,s}}\} \tag{3.1}$$

as the set of downstream alighting candidate stops, where $r_{h,s}$ is the route code of the s-th stage of passenger h and $o_{h,s}$ is the index of the boarding stop of the s-th stage of passenger h. Define $stop^{r_{h,s}}(i)$ as the code of the i-th stop along route $r_{h,s}$. The inference method is formulated as

follows:

$$\hat{d}_{h,s} = \underset{d_{h,s} \in A_{h,s}}{\operatorname{argmin}} \{ dist[stop^{r_{h,s+1}}(o_{h,s+1}), stop^{r_{h,s}}(d_{h,s})] \}, \quad \text{for } 0 < s < m_h \ , \quad \text{(a)}$$

$$\hat{d}_{h,s} = \underset{d_{h,s} \in A_{h,s}}{\operatorname{argmin}} \{ dist[stop^{r_{h,1}}(o_{h,1}), stop^{r_{h,s}}(d_{h,s})] \} \ , \quad \text{for } s = m_h \ , \quad \text{(b)}$$

$$s.t. \tag{3.2}$$

$$o_{h,s} \neq d_{h,s} \ , \quad \text{(c)}$$

$$dist[stop^{r_{h,s+1}}(o_{h,s+1}), stop^{r_{h,s}}(d_{h,s})] < c \ , \quad \text{for } 0 < s < m_h \ , \quad \text{(d)}$$

$$dist[stop^{r_{h,1}}(o_{h,1}), stop^{r_{h,s}}(d_{h,s})] < c \ , \quad \text{for } s = m_h \ . \quad \text{(e)}$$

Here, m_h is the number of travel stages of passenger h; $d_{h,s}$ is the index of the alighting stop of the s-th stage of passenger h; c is the cutoff distance. The first constraint, Eq. 3.2.c, is to make sure that the origin and destination of a travel stage are two different locations. The second constraint, Eq. 3.2.d and the third constraint, Eq. 3.2.e, provide a reasonable walking distance threshold within which passengers can travel from one stop to another on foot.

Two aspects of the heuristic method are worthy of notice:

- The first aspect relates to how to process the final travel stage: If the travel has more than two stages, it is arguable that the last travel stage was to reach a destination other than the first daily origin, and simply accepting that the second heuristic assumption is true brings a great risk of incorrect matching [15]. Therefore, we restrict the use of Eq. 3.2.b to two-stage travels with $m_h = 2$, and the final destination of a multiple-stage travel is not matched even if it meets the constraints.

- The second aspect is about making the method more sophisticated. It is possible to make use of timing (e.g., a combination of vehicle dwell time and passenger boarding/alighting time) and vehicle speed in addition to distance to determine the potential destination of a travel stage or create extra constraints. However, based solely on the entry-only data, it would be complex and less reliable to handle the noise incurred by temporal traffic jams, traffic light variations, and other uncertain conditions. Thus, we proceed with the basic heuristic method first and gradually increase the subtlety when extra relevant information (such as AVL data, road conditions, etc.) is available.

In the output of the heuristic method, the entry-only data in which the destinations are successfully matched are called *heuristic-based matched data*, and the data that remain unmatched are called *remaining unmatched data*.

3.3 Methodology

The problem of focus in this section is as follows:

> *Given the heuristic-based matched data and the remaining unmatched data, how can we infer the destination information of the remaining unmatched data?*

To deal with the remaining unmatched data, we propose a semi-supervised learning method to draw probabilistic conclusions about destinations in the presence of the statistics obtained from the heuristic-based matched data.

3.3.1 Semi-Supervised Learning Problem Setting

We realize that the above destination inference problem can be formalized as a semi-supervised learning problem. If there is no special notice, the term "stop" refers to "stop index" in this section.

In data science, the set of the heuristic-based matched data is known as the *labeled set* and the remaining unmatched dataset as the *unlabeled set*. Each data point is a vector $x_k = [t_k, r_k, o_k, h_k]$. In our problem, a data point is also called a *transaction*. We have the labeled data points $X_l = (x_1, x_2, \ldots, x_l)$, of which the labels $Y_l = (d_1, d_2, \ldots d_l)$ are known, and the unlabeled points $X_u = (x_{l+1}, x_{l+2}, \ldots, x_{l+u})$ of which the labels are unknown. Here, l is the size of labeled data, and u is the size of unlabeled data. Note that $d_k \in \{1, 2, \ldots, n^{r_k}\}$ is the alighting stop of the k-th transaction. r_k is the route code of the k-th transaction. We assume that both the labeled and unlabeled data are drawn from the same distribution.

The goal of semi-supervised learning is to make use of both labeled and unlabeled data to better infer the label of each unlabeled data point. In our case, we attempt to use this method to find the missing destinations of the remaining unmatched data.

3.3.2 One-Time Inference Method

We first utilize the one-time inference method, which is a straightforward way to infer the destination of a transaction based on the boarding and alighting statistics of the labeled set. In this method, we select proper labeled data to answer the question of how, given boarding time t, boarding route r, and boarding stop o, the probabilities of a passenger in general alighting from the public vehicle are distributed over downstream stops in the rest of the route. This probability distribution is called the *general alighting distribution*.

Because passenger activity will change from hour to hour in a day, the general alighting distribution could be inhomogeneous and vary over time. However, rather than complete randomness, human trajectories have a high degree of temporal and spatial regularity [8]. Thus, according to periodic nature of human activity, it is reasonable to assume that the general alighting distribution of a given boarding mode (i.e., a fixed route and a fixed boarding stop) is stable within a small period of time on a certain weekday. Based on this assumption, we infer the destinations of transactions on a weekday basis and construct the O-D matrices for each hour on a certain weekday, using which, we derive the hourly general alighting distributions.

Specifically, we select all the labeled transactions of a certain weekday (e.g., Wednesday) and reorganize them according to route. For each route, we count the boarding and alighting records hourly and store the counts in the O-D matrix of that hour. In this way, each route has 24 matrices. All the O-D matrices are stored into and accessed from dictionary $\Lambda = \{\Lambda_{T,r}\}$, where r indices a route code and T indices the clock hour. For route r in hour T, the corresponding O-D matrix $\Lambda_{T,r}$ is as follows:

$$\Lambda_{T,r} = \begin{pmatrix} 0 & \lambda_{1,2} & \lambda_{1,3} & & \lambda_{1,n^r} \\ 0 & 0 & \lambda_{2,3} & \cdots & \lambda_{2,n^r} \\ 0 & 0 & 0 & & \lambda_{3,n^r} \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \tag{3.3}$$

where

$$\lambda_{i,j} = \Lambda_{T,r}^{(i,j)} = \sum_{k=1}^{l} \mathbf{1}\left(t_k \in T,\ r_k = r,\ o_k = i,\ d_k = j\right) \tag{3.4}$$

indicates that in the labeled set, during the hour T, there are $\lambda_{i,j}$ passengers boarding on the route r at stop i and leaving the vehicle at stop j. Obviously, only the matrix's upper-right entries have non-zero values, because under normal conditions, no one can go back to any upstream stops or leave the vehicle at the same boarding stop after check-in.

After normalization, each row in $\Lambda_{T,r}$ presents the alighting probability distribution of a passenger in general. Thus, given an unlabeled transaction with the trip starting at stop o at time t, the probability of the passenger in general alighting from the public vehicle at the

candidate downstream stop d is as follows:

$$p\left(d|t,r,o;t\in T\right)=\frac{\Lambda_{T,r}^{(o,d)}}{\sum_{j=1}^{n^r}\Lambda_{T,r}^{(o,j)}} \tag{3.5}$$

This is the definition of the *general alighting distribution* mentioned above. It is a probability function of destination candidates given the statistics in relevant labeled transactions. Based on this probability distribution, the inference of the destination can be drawn via:

$$\hat{d}=\underset{d}{\text{argmax}}\{p\left(d|t,r,o;t\in T\right)\}. \tag{3.6}$$

With this framework, the one-time inference can be described as the following procedure:

- Given the route r and the boarding time $t\in T$, access the O-D matrix $\Lambda_{T,r}$ from Λ;
- Given boarding stop o, pick out the o-th row;
- Normalize the row vector to obtain the general alighting distribution; and
- Infer the destination $\hat{d}$ using Eq. 3.6.

3.3.3 Self-Training (ST)

The one-time inference can be viewed as a preliminary supervised learning method and the accuracy is low when the size of unlabeled data is comparable to the size of labeled data. In our case, the size of the unlabeled data is considerable and fully supervised learning is therefore not feasible. Many ML researchers have found that the unlabeled data, when used in conjunction with the labeled data, can produce considerable improvement in learning accuracy [31]. This property also applies to our passenger destination inference problem – by using the information from unlabeled data to improve the general alighting distribution in Eq. 3.5, the inference Eq. 3.6 will become more reliable. Self-training [32] is a semi-supervised methodology that makes full use of the unlabeled data to *iteratively* improve accuracy. Figure 3.4 illustrates the idea of self-training.

A self-training method involves incrementally increasing the labeled set as learning proceeds. Specifically, the method uses a *base predictor* trained from labeled data to predict the labels of unlabeled data with a confidence score associated to each prediction. Then, certain newly-labeled data with top-ranked prediction confidence are selected to be added to the original labeled set; the rest of the newly-labeled data will replace the original unlabeled set. This

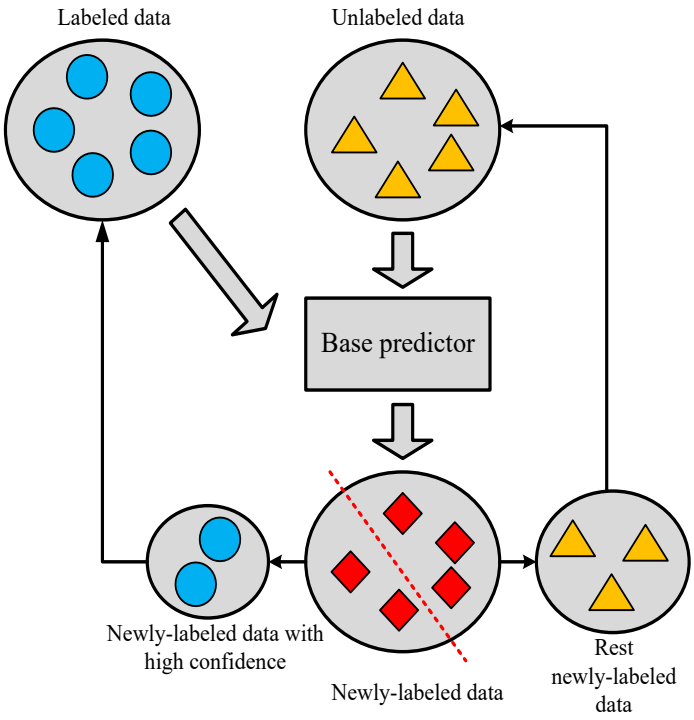

Figure 3.4: Diagram of semi-supervised self-training.

process repeats until some termination conditions are met (i.e., all unlabeled data are labeled or the iteration exceeds a threshold number). The way we select newly-labeled data to update the labeled set is called *selection strategy*. It is crucial that both the base predictor and selection strategy are well-designed to ensure reliable inference. To implement self-training that applies well to our passenger data, the base predictor and selection strategy of newly-labeled data are designed as follows.

Base Predictor

The performance of self-training strongly depends on the confidence of the prediction made by the base predictor. Thus, we want to quantify this confidence to correctly reflect the reality. The one-time inference is a good starting point that provides a nice framework for the design of our base predictor. The base predictor considered in this chapter consists of two parts:

- To construct the general alighting distributions based on the statistics of existing labeled data; and
- To draw probabilistic conclusions about destinations using the constructed distributions.

The inference method in Eq. 3.6 still applies here, as it determines the optimal destination given the statistics of similar observations. Following this framework, the key factor that affects inference quality is the quality of the general alighting distributions. Different from the one-time inference method, which simply relies on the direct statistics of the labeled data of a fixed hour, in order to develop a more reliable approach, we must consider: 1) what kind of labeled data should be selected, and 2) how many of them are sufficient for constructing a good distribution.

To this end, we design a base predictor that predicts the missing destination labels based on the statistics of a similarity-based "training set" selected from the labeled set. Specifically, we use a time window centered at the input boarding time t to define the temporal similarity. The half-length of the time window is called the *bandwidth parameter* Δ (note the bandwidth is 2Δ). Then, the time window T^t centered at boarding time t is:

$$T^t = \{\tau | t - \Delta \leq \tau \leq t + \Delta\} \tag{3.7}$$

On the other hand, we modify the O-D matrix dictionary $\Lambda = \{\Lambda_{T,r}\}$ by increasing the temporal granularity by constructing a base O-D matrix every 6 minutes, and in this case, T in Eq. 3.3 is a predetermined 6-minute interval. Then, for a transaction of route r at time t, the customized O-D matrix is

$$\Lambda_{t,r} = \sum_{\{T|T\cap T^t \neq \emptyset\}} \Lambda_{T,r}. \tag{3.8}$$

This process accumulates the O-D pairs in the period of T^t to construct an O-D matrix customized on the boarding information (t, r, o). Based on Eq. 3.8, the corresponding alighting distribution is:

$$p\left(d|t,r,o\right) = \frac{\Lambda^{(d)}_{t,r,o}}{\sum_{j=1}^{n^r} \Lambda^{(j)}_{t,r,o}}, \tag{3.9}$$

where $\Lambda_{t,r,o} = \Lambda^{(o,:)}_{t,r}$ is the o-th row of the matrix $\Lambda_{t,r}$. This is the general alighting distribution based on the selected similarity-based labeled set. Using this distribution, we apply the base prediction:

$$\hat{d} = \underset{d}{\text{argmax}}\{p\left(d|t,r,o\right)\}. \tag{3.10}$$

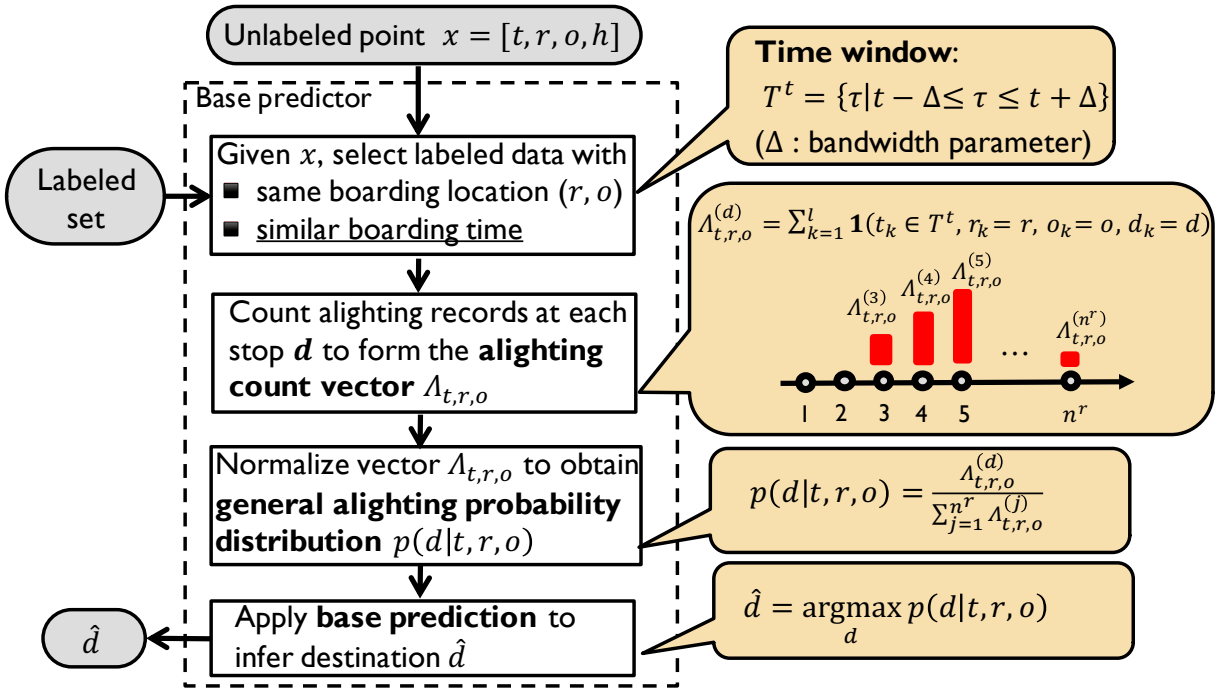

Figure 3.5: The design of base predictor.

Note that Eq. 3.10 is different from Eq. 3.6. The latter has to choose a matrix from the dictionary Λ, while the former constructs a customized matrix $\Lambda_{t,r}$ centered at the input's boarding time. Figure 3.5 shows the pipeline of the base predictor.

Selection Strategy

The design of selection strategy takes advantage of the base prediction in Eq. 3.10 and uses the probability $p(\hat{d}|t,r,o)$ as the prediction confidence, then selects the newly-labeled data with top-ranked prediction confidence to update the labeled set. Specifically, we sort the newly-labeled data in a descending order with respect to confidence $p(\hat{d}|t,r,o)$ and select the first N data – along with their labels – to add to the original labeled set. The rest of the newly-labeled data will replace the original unlabeled set. The value N is a parameter called the *threshold of selection*. The number of the iteration of the entire inference highly depends on N. Note that the selection of labeled data for the training set in Eq. 3.8 is NOT part of the selection strategy, but a part of the base predictor.

Here we need to clarify a point related to the trade-offs between training effort and performance. In many cases, the use of an ill-designed training selection may eventually cause a decrease in inference accuracy. In general, the more relevant labeled data used for training, the

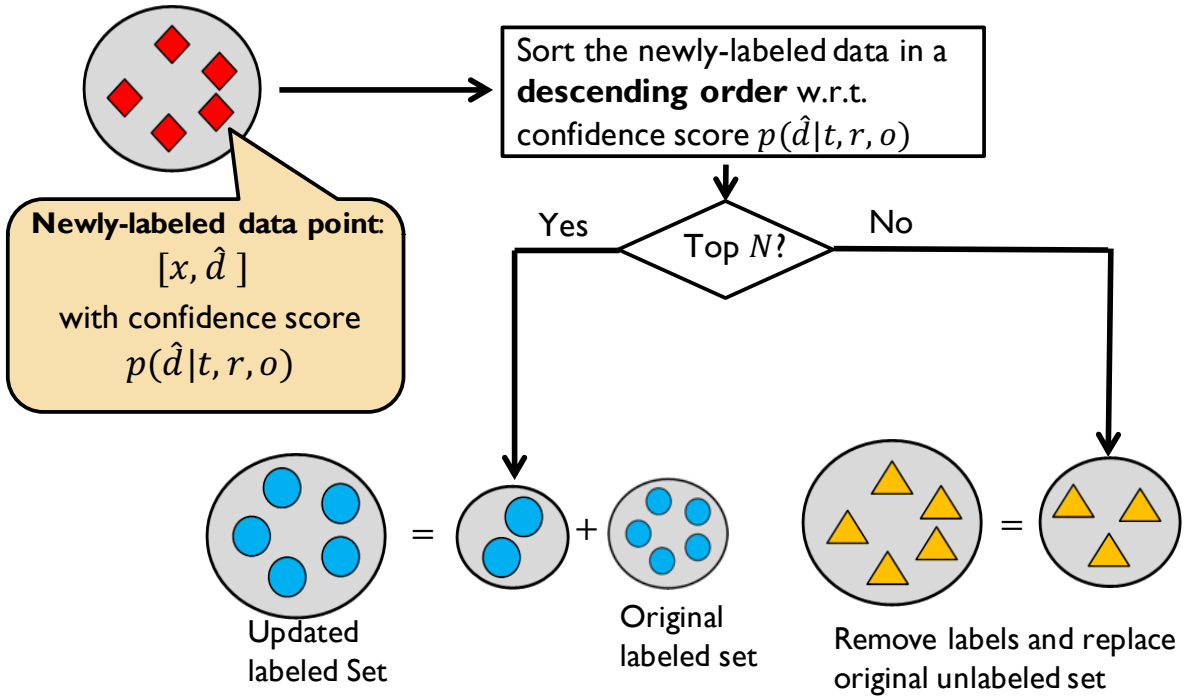

Figure 3.6: Selection strategy.

more reliably the base predictor will perform. However, in a scenario involving large data size, training effort is a critical concern. When processing EEG-based brain-computer interface data, Li *et al.* proposed an idea: Using a proper selection metric, designed for learning the classifier from a limited number of training data, the training effort could be significantly reduced while keeping competitive accuracy [33]. Our design of the selection metric achieves this advantage, and the use of the training data selection in Eq. 3.8 is flexible to trade off inference accuracy against the algorithm's scalability. As will be seen later in the experiments, by tuning the time window bandwidth and threshold of selection, it is possible to reduce the training effort without seriously deteriorating the inference accuracy. The pipeline of the selection strategy designed in this section is shown in Figure 3.6.

3.3.4 Self-Training using Personal Historical Information (STP)

We notice that many passengers may appear and take buses consistently in different days, and this inspires us to incorporate personal historical information to modify the base predictor and improve the inference performance. Trepanier *et al.* proposed an idea to incorporate passengers' daily and weekly travel patterns to complete the missing destinations [34]. This is a feasible insight, because passenger ID is usually available in entry-only data. Given a passenger ID,

the passenger's multiple daily travel stages can be retrieved and corresponding destinations can be estimated via the heuristic method. In this section, we extend the scope further: Once the transactions of the same passenger are recorded continuously over a long time (e.g., several weeks or months), the passenger's travel pattern on a daily, weekly, and even monthly basis can be estimated by looking into his or her personal historical records. In statistical practice, the personal mobility pattern information can be formulated as *prior* (i.e., probability distribution that would express one's beliefs about the destination of a specific person before some boarding information is taken into account). It is expected that the prior can increase the reality level of the alighting distribution of passengers who show up consistently on different days.

Personal alighting distribution q

To be specific, we assume that in the entry-only data, parts of passengers (identified by attribute h) appear repetitively and consistently in different days. Given an unlabeled data point with passenger ID h, from the labeled set, we search for the passenger's historical transactions with the same boarding location (r and o). The boarding time information is not involved here due to concerns of potential data scarcity. With the assumption, we construct a personal O-D matrix for each passenger ID and save them in dictionary $\Omega = \{\Omega_{r,h}\}$. We count the alighting records of passenger h at each downstream candidate stop d to form the *personal alighting count vector* $\Omega_{r,o,h}$, of which the d-th element is:

$$\Omega_{r,o,h}^{(d)} = \sum_{k=1}^{l} \mathbf{1}\left(r_k = r,\ o_k = o,\ h_k = h,\ d_k = d \right). \tag{3.11}$$

$\Omega_{r,o,h}$ is the o-th row of matrix $\Omega_{r,h}$. The normalized personal alighting count vector $\Omega_{r,o,h}$ defines the *personal alighting distribution* of the passenger h. Then, for passenger h getting on a mass transit vehicle at stop o of route r, the probability of the passenger getting off at stop d is:

$$q\left(d|r,o,h\right) = \frac{\Omega_{r,o,h}^{(d)}}{\sum_{j=1}^{n^r} \Omega_{r,o,h}^{(j)}}. \tag{3.12}$$

The pipeline for constructing the personal alighting distribution is illustrated in Figure 3.7.

Modified alighting distribution p'

Both the personal alighting distribution (Eq. 3.12) and general alighting distribution (Eq. 3.9) are generative probabilistic models of destination candidates, from which we can draw destination samples. To make full use of both distributions in order to make a reliable decision about the

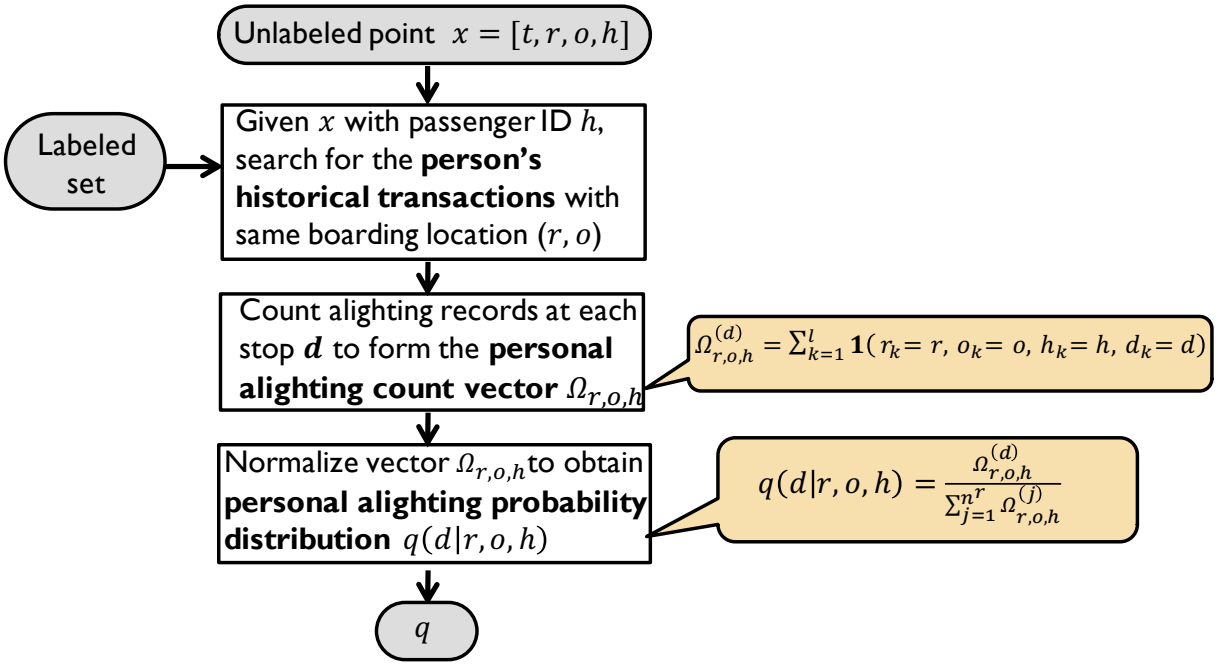

Figure 3.7: Construction of personal alighting distribution q.

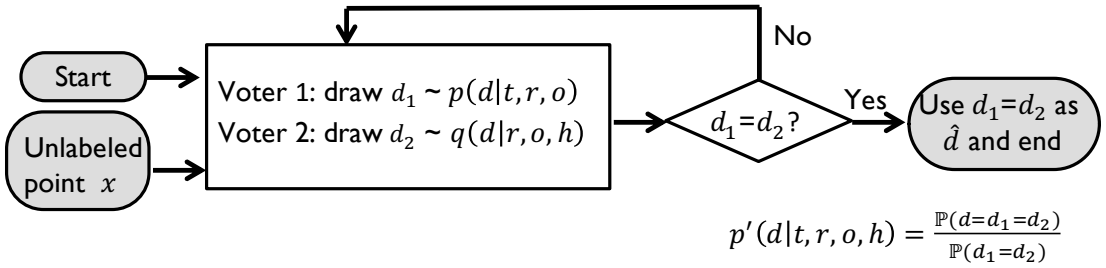

Figure 3.8: Voting process behind the modify alighting distribution p'.

destination, we propose a voting process. In this process, we have two independent voters. One of them has a voting distribution that follows the personal distribution q in Eq. 3.12 and the other follows the general distribution p in Eq. 3.9. Each time, each voter chooses a destination independently for the input unlabeled data point. Only when the two voters agree with each other do we make use of the chosen destination. Otherwise, the voting repeats until the two voters agree. Then, the *modified alighting distribution* is shown as follows:

$$p'(d|t, r, o, h) = p \odot q = \frac{\Lambda^{(d)}_{t,r,o} \cdot \Omega^{(d)}_{r,o,h}}{\sum_{j=1}^{n^r} \left(\Lambda^{(j)}_{t,r,o} \cdot \Omega^{(j)}_{r,o,h} \right)}. \tag{3.13}$$

This expression formalizes the probability of destination d being finally selected, conditioned on the two voters agreeing with each other. The voting process is presented in Figure 3.8.

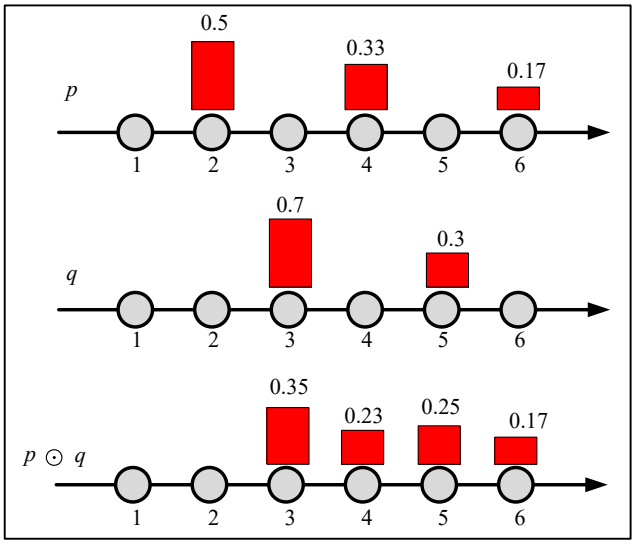

Figure 3.9: Modification of p using q under orthogonality condition.

p' under corner conditions

Under corner conditions where row vectors $\Lambda_{t,r,o}$ and $\Omega_{r,o,h}$ are *orthogonal* to each other, it is invalid to conduct the operation of Eq. 3.13. This issue becomes nontrivial when we are tuning the bandwidth parameter Δ. An alternative operation for handling this corner condition is to use the cumulative distribution function (CDF). Let $P(d|t,r,o)$ and $Q(d|r,o,h)$ be the CDFs of Eq. 3.9 and Eq. 3.12, respectively. We propose the following modified alighting distribution under the corner conditions:

$$\begin{aligned} p'(d|t,r,o,h) &= p \odot q \\ &= P(d|t,r,o) \cdot Q(d|r,o,h) - P(d-1|t,r,o) \cdot Q(d-1|r,o,h), \end{aligned} \tag{3.14}$$

where $p'(d=1|t,r,o,h) = P(1|t,r,o) \cdot Q(1|r,o,h)$. This is equivalent to form a modified CDF through the element-wise multiplication of P and Q. As can be seen in Figure 3.9, by following Eq. 3.14, the $p \odot q$ operation effectively modifies the alighting distribution of p using q. In particular, probabilities at stops 3, 4, and 5 are properly balanced. In contrast, the calculation of the distribution using Eq. 3.13 is invalid due to the orthogonality between p and q.

After construction of the modified alighting distributions, the inference procedures are the same as those in section 3.3.3. The STP inference is conducted using the following expression:

$$\hat{d} = \underset{d}{\operatorname{argmax}}\{p'(d|t,r,o,h)\}. \tag{3.15}$$

Algorithm 2 Self-training using personal historical information

Input: L, U, N, Iter; L: Labeled set; U: Unlabeled set;
N: Threshold of selection; Iter: Max number of iteration;
Δ: Bandwidth parameter
Output: L,U

1. $\{\Lambda_{T,r}\}$ ←ODMatrixDisctionary(L)
2. $\{\Omega_{r,h}\}$ ←PersonalODMatrixDisctionary(L)
3. n←1
4. **while** (U != empty) and (n< Iter) **do**
5. S= {}
6. n← n+1
7. **for** each x_i =(t_i, r_i, o_i, h_i) ∈ U **do**
8. p ← alightingDistribution($\{\Lambda_{T,r}\}$, t_i, r_i, o_i, Δ)
9. q ← personalPrior($\{\Omega_{r,h}\}$, r_i, o_i, h_i)
10. **if** q is valid **do**
11. p' ← $p \odot q$
12. (d_i, $Prob\_b_i$) ← Estimator(p')
13. **else do**
14. (d_i, $Prob\_b_i$) ← Estimator(p)
15. **endif**
16. S← S+ (x_i, d_i, $Prob\_b_i$)
17. **endfor**
18. S_{label}←select N highest $Prob\_b$ points{(x_i, d_i)} from S
19. S_{rest} ← the remaining less confident points {x_i} in S
20. L← L ⋃ S_{label}
21. U← S_{rest}
22. $\{\Lambda_{T,r}\}$ ←ODMatrixDisctionary(L)
23. $\{\Omega_{r,h}\}$ ←PersonalODMatrixDisctionary(L)
24. **endwhile**

return L,U

The algorithm of the destination inference using self-training with personal historical information is presented in Algorithm 2. The algorithm pre-processes the labeled set L to construct the general O-D dictionary Λ and the personal O-D dictionary Ω (lines 1 to 2). Variable n tracks the iteration number (line 3). In each iteration (lines 4 to 24), it uses a temporary set S to collect the newly labeled data. Specifically, for each data point in the unlabeled set U (line 7), it constructs the general alighting distribution p and personal alighting distribution q (lines 8 to 9). Then, it infers the destination d_i and the corresponding conference score $Prob\_b_i$ using the modified alighting distribution p' (constructed by Eq. 3.14 and Eq. 3.15) if the personal information q is available (lines 10 to 12). Otherwise, it uses the general alighting distribution p to conduct the inference (lines 13 to 15). Line 16 collects the newly labeled data point $(x_i, d_i, Prob\_b_i)$ to S. Lines 18 to 21 execute the selection strategy to add the first N (i.e., threshold of selection) most

Passenger ID	Boarding code	Alighting code	Route	Boarding time (s)	Alighting time (s)
119959	CSXS2	CVI2	602_rev	60132	60322
119959	NAT2	ACN1	204_rev	60502	60719
119959	AQL1	SPTO1	203	61430	61532

Table 3.1: Simulation data examples.

confident newly-labeled data in S to the labeled set L and replace the unlabeled set U with the rest S_{rest}. The algorithm updates the Λ and Ω using the new labeled set L and turns to next iteration. This process repeats until U is empty or the iteration n is beyond the termination number $Iter$ (line 4).

3.4 Data Pre-Processing for Experiments

Two data sources are used for verifying the self-training based methodology: the simulation data generated by the joint traffic-passenger simulator P-SUMO and a subset of the heuristic-based matched AFC data.

3.4.1 Passenger Simulation Data of Porto City

Chapter 2 introduced the P-SUMO for the city of Porto, Portugal. The simulation covers the main bus transit system (136 routes, 855 bus stops, and 5,723 trips in a normal weekday). The P-SUMO generates bus passenger travel demands and simulates the passengers' behavior (in terms of waiting at the stop, boarding, alighting, and transiting until arriving at the final destination). From the simulation output log, we extract relevant passenger attributes: passenger ID, route, boarding time (in seconds), and the boarding/alighting stops of each transaction. The records of ten weekdays (i.e., Wednesdays), totaling 635,111 transactions, are selected as the first dataset for the experiments. We remove the destination information and apply the inference methodology to the simulation data. Examples of simulation data for "Passenger 119959" are presented in Table 3.1.

3.4.2 AFC Passenger Data of Porto City

The second dataset is the real bus AFC data, which contain transaction records in three months of 2010 (see Section 2.2.4 for descriptions). For each transaction record, we extract six attributes:

Passenger ID: 20029975520					
Route	Boarding code	Dirt.	Date	Vehicle start time (s)	Boarding time (s)
801	ATSR2	2	19/5	30547	30977
801	CB1	1	19/5	31174	32849
801	ATSR1	1	19/5	44252	46210
801	SPC	2	19/5	49739	49793

Table 3.2: Porto AFC data examples.

the 1) travel card serial number; 2) transaction timestamp; 3) bus stop where the transaction took place; 4) route number; 5) route direction; and 6) vehicle trip start time.

We reorganize the data according to our needs. Specifically, the transaction records in all Wednesdays of the three months are selected for the experiments; there are 12 Wednesdays, totaling 2,422,079 transactions in the areas of interest. Those Wednesdays are normal weekdays and any local holidays are avoided. Within each day, the transactions are grouped on a passenger basis. The reorganized samples of a passenger with multiple travel stages are presented in Table 3.2.

3.5 Experiments

We conduct experiments using both the simulation and real AFC data and compare performance of different approaches. This section describes the experimental results.

3.5.1 Experimental Setup and Evaluation Metrics

Preparation of Datasets

We apply the heuristic method described in section 3.2 to obtain the labeled and unlabeled sets. Following best practices, we control the cutoff distance to a conservative level. For the simulation data from P-SUMO, the cutoff distance c is 500 meters. With this setting, of all 635,111 transactions, 353,123 are matched with destinations. The remaining 281,988 transactions are placed in the unlabeled set; thus, we state that the heuristic-based inference rate is 55.60%. Because we have the ground-truth destinations for the simulation data, we can calculate the accuracy of the labeled set as 86.23% (the number of correct labels divided by the size of the labeled set).

For the AFC data, the cutoff distance c is 600 meters. The heuristic method is able to match

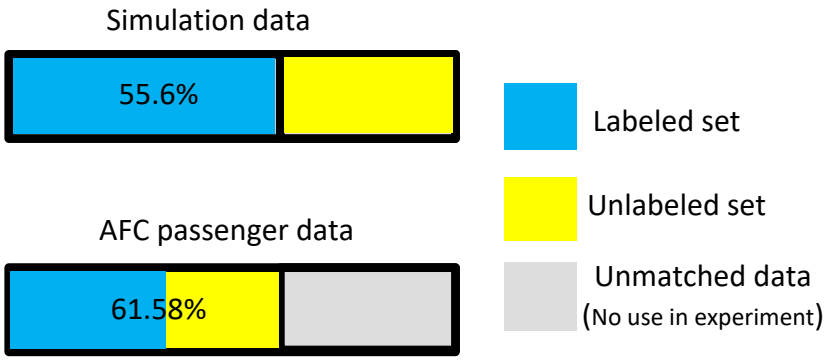

Figure 3.10: Breakdowns of the data used in experiments.

about 61.58% of the total transactions, so that 1,491,416 data have labels. Because the ground-truth destinations are not available, we select the matched data of three Wednesdays (20/1, 14/4, 12/5) as the unlabeled data, totaling 379,676 data points. The rest of the matched data, totaling 1,111,740 data points, serves as the labeled set.

The data breakdowns are shown in Figure 3.10.

Methods for Comparison

Four different types of destination inference methods are implemented. The **random inference method** assigns a destination randomly drawn from the downstream candidates to an unlabeled data point. The **one-time inference**, **self-training (ST)**, and **self-training with personal historical information (STP)** methods conduct inference in the way described in section 3.3. The proposed ST/STP methods are designed with tunable parameters. We are particularly interested in studying the impact of the time window bandwidth parameter Δ and threshold of selection N on inference accuracy.

Because different remaining unlabeled sets have different sizes, we need a common indicator to represent the threshold. We introduce a normalized selection threshold $N_k = N/u$, where u is the size of the remaining unlabeled set. In practice, we use its reciprocal N_k^{-1} which is called the *selection threshold coefficient*. Using N_k^{-1} is beneficial because it implies the rigorous level of the selection strategy and the approximate number of iterations the algorithm has to execute before completion. The larger the N_k^{-1} is, the more selective the selection strategy will be, and the more iteration times we will have.

Evaluation Metrics

The performance of different inference methods is compared based on two evaluation metrics: the mean square error (MSE) and inference count accuracy. The definition of MSE is as follows:

$$MSE = \frac{\sum_{i=1}^{u} (d_i^{true} - \hat{d}_i)^2}{u}, \tag{3.16}$$

where d_i^{true} is the true stop index of test set and $\hat{d}_i$ is the inferred stop index. The inference count accuracy is defined by the following value:

$$Accuracy = \frac{\sum_{i=1}^{u} \mathbf{1}(d_i^{true} = \hat{d}_i)}{u}. \tag{3.17}$$

In this chapter, the terms "inference accuracy" and "accuracy" all refer to this value. It is necessary to note that the accuracy of simulation data is the ground-truth accuracy because the true destinations are available. On the other hand, the accuracy of AFC data inference is based on the labels provided by traditional methods.

3.5.2 Results of Parameter Sensitivity

We first study the influence of the time window bandwidth and selection threshold on accuracy.

Figure 3.11 illustrates that wider bandwidths bring improvements in accuracy given a fixed selection threshold coefficient. Both ST and STP perform more reliable destination inference as more relevant labeled data are used for learning a base predictor. An interesting property observed here is an elbow region beyond which the accuracy is noticeably less sensitive to the bandwidth. This implies that by selecting a bandwidth in the elbow region, we may reduce the training set without significantly deteriorating the accuracy. For instance, according to Figure 3.11.a, the accuracy of $\Delta = 3$ hours (bandwidth is 6 hours) is reasonably close to that of $\Delta = 10$ hours (bandwidth is 20 hours), but the training effort of processing the labeled data of 20 hours is way heavier than the former. In experiments, the algorithm run time of 10-hour Δ is reduced to less than 43% when using 3-hour Δ.

In big data scenarios, training effort is a primary concern. The process of properly selecting a limited number of training data to reduce the training effort without seriously reducing accuracy is important for the scalability of any approach to large-scale datasets. The proposed self-training methods can meet this end. The red markers in Figure 3.11 are suggested elbow points that provide reasonable trade-offs between accuracy and training effort.

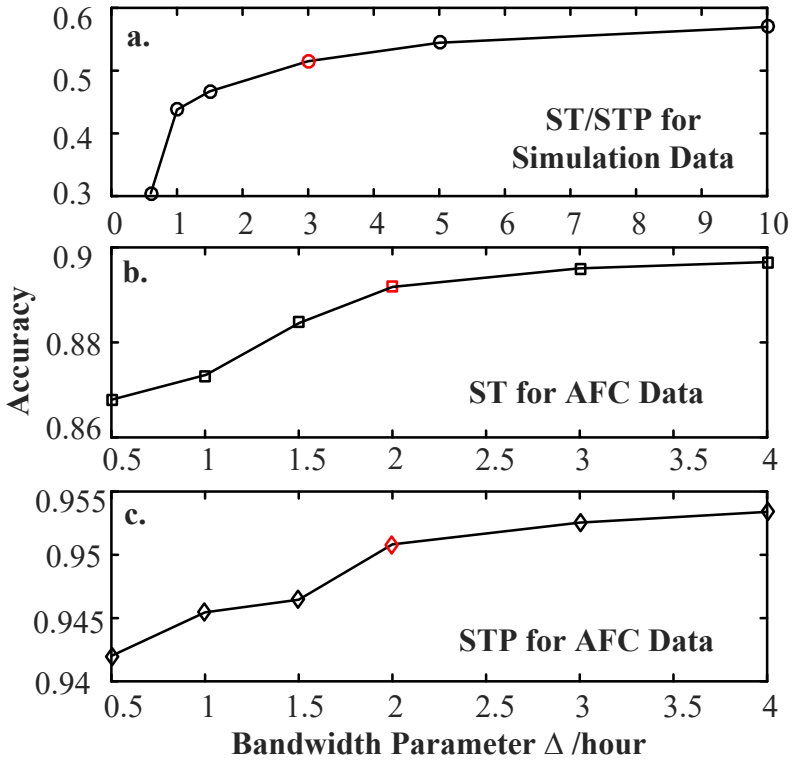

Figure 3.11: Impact of the bandwidth on accuracy ($N_k^{-1} = 10$).

Figure 3.12 shows the performance of ST/STP on both datasets as the selection threshold increases for certain bandwidths. A larger selection threshold coefficient N_k^{-1} implies that a smaller number of newly-labeled data are selected to update the labeled set. In this way, more iterations are executed before the inference algorithm completes. A large N_k^{-1} is beneficial: With a smaller newly-labeled set in each iteration, the base predictor experiences a more precise training. As a result, we obtain a more reliable base predictor to deal with the remaining unlabeled data of which the confidence is very low.

Figure 3.12 also shows that the STP approach can effectively take advantage of personal historical information to further improve the inference accuracy. For the AFC data, STP is generally superior to ST under the same conditions and achieves a higher accuracy. However, this improvement does not appear in the simulation data because the simulator P-SUMO generates and simulates passengers on a daily basis, with the passengers simulated in different days being completely independent. As a result, no personal historical information about the simulated passengers can be extracted from the simulation data. Fortunately, most real data contain identification information linked to each passenger, and thus, the practical value of STP in the real world is high.

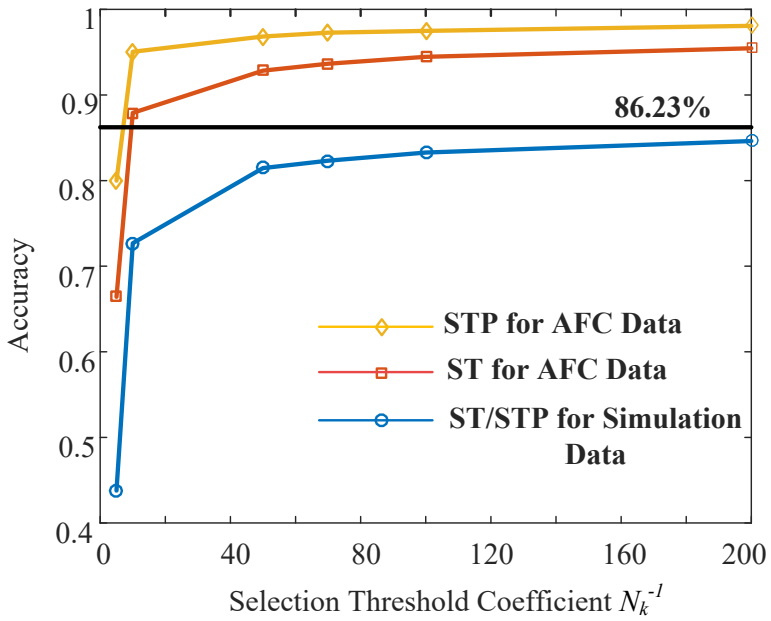

Figure 3.12: Impact of the selection threshold on accuracy ($\Delta = 2$ hours for AFC data, $\Delta = 3$ hours for simulation data).

One more discussion on Figure 3.12 is about how the wrong labels in the original labeled set affect the overall accuracy. Because we have the ground-truth labels for the simulation data, this issue can be studied. The benchmark line in the figure indicates the proportion of correct labels in the original labeled set. This value implies that 86.23% is the upper bound on the ST/STP inference accuracy; by selecting proper parameters and conducting inference through more precise training, the ST/STP method is shown to approach and converge to that limit. At the parameter setting of $\Delta = 3, N_k^{-1} = 200$, the self-training based methods achieve a competitive accuracy of 83.27%. This result demonstrates the robustness of the proposed self-training based methods against wrong labels in the original labeled set.

3.5.3 Comparison among Methods

Table 3.3 provides the performance of different destination inference methods. Compared to the random method, the one-time inference method reduces the inference MSE to a noticeable extent. However, the overall inference quality is still unsatisfactory: By making the inference only once based on the statistics of the labeled set, the one-time inference would be seriously affected by noise in the alighting distributions when relevant samples are sparse. In contrast, the ST and STP methods incrementally make use of the highly-confident newly-labeled data and filter out the noise when constructing the alighting distributions, leading to a more reliable

	Simulation Data		AFC Data	
	MSE	Accuracy	MSE	Accuracy
Random	248.09	3.12%	241.46	2.98%
One-Time Inf.	25.89	17.76%	63.27	26.15%
ST	**2.21**	**83.27%**	**3.758**	**94.46%**
STP	**2.21**	**83.27%**	**1.964**	**97.49%**

ST/STP parameters: N_k^{-1}=100 for all; Δ=2 hours for AFC data, Δ =3 hours for simulation data.

Table 3.3: Destination inference performance of different methods.

inference. Eventually, the ST method significantly reduces MSE and achieves an accuracy of about 83% for the simulation data and 94% for the AFC data. Furthermore, the STP method proves its ability to incorporate the passengers' historical information to further improve the base predictor, achieving an impressive accuracy of 97%.

Finally, the inference rate of ST/STP is 91.85% for simulation data and 99.96% for AFC data, respectively. Sample deficiency is the major cause of non-inferable data, especially for the simulation data. The deficiency is due to the fact that some passengers' boarding scenarios in the unlabeled set do not appear in the labeled set. In the P-SUMO, the passengers generated on a weekday are relatively small compared to the urban area in SUMO, leading to sample deficiency. Due to the lack of reference, some data in the remaining unlabeled set cannot be inferred. Another cause of non-inferable data is that, in our implementation, we set a confidence threshold (0.55), and the inference confidence must exceed this threshold for the results to be accepted. This acceptance rule could cause consistent rejection to some inferred destinations. We believe that the acceptance rule can ensure the reliability of our approach for more real-world data inference problems.

The experimental results support our claim that the semi-supervised self-training methodology is an effective solution to inferring the destinations of the remaining unmatched data, and that the inference performance is significantly improved when compared to baseline methods that do not use self-training.

3.6 Summary

This chapter describes a semi-supervised self-training method for filling the missing destination features in the entry-only data. We show that with proper design of the base predictor and selection strategy, we can achieve significant improvement in the inference accuracy. Furthermore,

tuning the built-in parameters can effectively reduce the training effort. We conduct experiments to demonstrate that our proposed approaches performed better than baseline methods that do not use self-training.

The self-training inference method is extendable, as the ST/STP framework can incorporate any kind of pre-knowledge (e.g., human mobility patterns, weather, road conditions, etc.) to improve inference; it is also possible to apply the self-training method to other types of destination inference problems once fragments of O-D knowledge about human mobility are available.

The processed bus AFC data will be used for configuring the passenger flow in the joint simulator P-SUMO for on-vehicle experience simulation.

Chapter 4

SC-M*: A Multi-Agent Path Planning with Soft-Collision-Free Constraint

The focus of the thesis includes optimizing the on-vehicle experience via mobility planning on a city scale. Different from traditional planning research for individual passengers, our research attempts to simultaneously plan for multiple passengers while limiting the level of interference among them. We consider the total set of passengers to be divided into two subsets: the *client passengers* who request planning service from our planner, and *non-client passengers* who do not.

In addition to optimizing over a linear combination of multiple criteria, we formalize the passenger mobility planning as a multi-agent path planning (MAPP) problem, where multiple client passengers are coupled and may interfere with one another on some services. State-of-the-art MAPP solvers, such as M*, cannot scale well to such a city-scale MAPP task. One bottleneck of M* is that the hard-collision-free constraint is too restricted, incurring a surge in the size of the search space when collisions among many agents happen.

To handle this bottleneck, we introduce the concept of *soft collision* to the MAPP setting: A collision among client passengers is soft, quantified using a collision score, and different client passengers have different scores according to their *dissatisfying experience*, in which the quality of some services is low due to interference. We then develop a generalized version of the M*, the *soft-collision-free M** (SC-M*), for solving the MAPP problem under the soft-collision context. In the field of intelligent planning and scheduling technology, a client passenger can be viewed as an agent and the mobility planning can be viewed as the path planning, leading the agents to move from the origin to the destination with constraints.

The SC-M* extends M* by taking the perspective of soft collision. Specifically, the SC-M*

tracks the collision score of each agent and places agents whose collision scores exceed certain thresholds into a soft-collision set for *sub-dimensional expansion*, a technique that limits the search space while maintaining the cost optimality of the algorithm. In this way, SC-M* achieves improved scalability to handle a larger number of agents while limiting the probability of collisions to a bound.

Later in this chapter, we demonstrate that the SC-M* has advanced flexibility and scalability for efficiently solving MAPP problems under the soft-collision context and can handle complex environments (e.g., with multiple types of agents requesting multiple types of resources). Furthermore, we compare SC-M* to other SC-based MAPP solvers and study the advantages and trade-offs of SC-M* against baselines in terms of path cost, success rate and run time.

As an outline, this chapter details and theoretically studies the framework of the SC-M*, while the next chapter focuses on the application of SC-M* to a real public transit system. The paper [35] covers most of the work in this chapter.

4.1 Background of MAPP

MAPP is a problem involving finding the set of least-cost paths for a set of agents co-existing in a given *graph* such that each of the agents is free from collision (i.e., at least two agents move to the same location at the same time). MAPP attracts increasing attention due to its practical applications in multi-robot systems for surveillance automation, video gaming, traffic control, and many other domains [36, 37, 38, 39]. This problem is, however, difficult because the configuration space grows exponentially with the number of agents in the system, incurring extremely heavy computational efforts. It is an NP-hard problem to find optimal solutions for MAPP in its general form [40].

Approaches to solving MAPP fold into three main categories: *coupled*, *decoupled* and *intermediate* [41]. *Coupled* approaches search the *joint configuration space* of the multi-agent system, which is the *Tensor product* of the free configuration spaces of all the individual agents. A popular coupled planner is the A* algorithm that directly searches the whole joint configuration space, making such an approach computationally infeasible when the number of agents is large. Furthermore, enhanced variants of A*, such as operator decomposition (OD), enhanced partial expansion A* (EPEA*), and iterative deepening A* (IDA*), can – to some extent – mitigate the exponential growth in the number of neighbors by improving the admissible heuristics [42, 43, 44]. Coupled approaches are optimal and complete, but usually at high computational cost. *Decoupled* ap-

proaches plan for each agent separately and then adjust the path to avoid collisions. Algorithms from this category are generally faster because the graph search and collision-avoidance adjustment are performed in low-dimensional spaces. However, optimality and completeness are not guaranteed [45, 38]

Intermediate approaches lie between coupled and decoupled ones because they dynamically couple agents and grow the search space during the planning. In this way, the search space is initially small and grows when necessary. A few intermediate MAPP algorithms can guarantee optimality and completeness. State-of-the-art examples include Conflict-Based Search (CBS) [41, 46]. CBS is a two-level algorithm. At the high level, conflicts are added into a *conflict tree* (CT). At the low level, solutions consistent with the constraints given by the CT are found and updated to agents. CBS behaves poorly when a set of agents is strongly coupled. Meta-agent CBS (MA-CBS) is then proposed by merging strongly coupled agents into a meta-agent to handle the strongly coupled scenarios.

The M* algorithm is a state-of-the-art coupled approach. It starts with decoupled planning and applies a strategy called *sub-dimensional expansion* to dynamically increase the dimensionality of the search space in regions in which agent collisions occur. In this way, an efficient graph search with a strict collision-free constraint can be achieved, while minimizing the explored portion of the joint configuration space. M* identifies which subsets of agents can be safely decoupled and hence plans for multi-agents in a lower-dimensional space. Compared to CBS and its variant MA-CBS, M* and its variants (e.g., recursive M* [rM*]), have much more fine-grained control over some technical details, such as the management of conflict sets for better scalability. The fine-grained nature of M* allows it to be integrated into MA-CBS to take advantage of both [47]. Recent work extended both M* and CBS algorithms to handle the imperfect path execution due to unmodeled environments and delays [48, 49].

Most fundamental MAPP approaches assume hard collisions, not allowing any two agents to co-exist at the same node or edge. This chapter extends collisions to the concept of soft collisions in which agents may softly collide when their satisfaction conditions are reduced. Based on this concept, we introduce the soft-collision-free M* (SC-M*), which generalizes the M* framework to efficiently handle MAPP problems under the soft-collision context.

4.2 Motivations of SC-M*

SC-M*, introduced in this chapter, generalizes M* to handle a subset of real MAPP problems under the soft-collision context. Problems of this type have some properties in common: 1) Agents can softly collide when their satisfaction conditions are reduced; 2) the extent of reduction in satisfaction depends on how long the dissatisfying situation lasts in terms of distance or time.

In addition to public transit scenarios focused in this thesis, other examples where SC-M* can apply include: network traffic engineering, where multiple data streams can route through a router. Long streams will have a higher chance of being blocked when unexpected traffic spikes pop up, exceeding the link capacity [50]. How to maximize the throughput with a bounded chance of blocking is of great interest to researchers in the field of communications and computer networks. Another example is planning for large-scale self-driving cars, where multiple cars can share the same lane, but the number of cars on a road will influence the chance of crashes among autonomous vehicles [51, 52]. Scholars and engineers dealing with the fundamentals of autonomous vehicles in unstructured and dynamic environments aim to increase the road traffic while bounding the crash risk.

Thus, introducing the soft collision property to agents and developing SC-based MAPP planners are important for many practical problems. Our work is the first attempt to generalize the M* framework to this type of problem and introduce SC-M*. The value of this research is twofold. On one hand, the planning and scheduling academic community would like to see 1) how much flexibility and scalability the M* solver can have when applying to the soft-collision context, 2) whether and how much the completeness and optimality of M* will be affected under the soft-collision setting, and 3) how the design of some components in SC-M* (e.g., the design of functions mapping agents' experience to the collision probability) will affect the performance. On the other hand, in the field of intelligent transportation systems, urban transportation planning practitioners would like to see how M* can be applied to the public transit domain and settle the passenger mobility on a large scale to help with the congestion and resource management issues, given the challenging surge in city population and mobility needs. This chapter focuses on the first aspect, while the next chapter deals with the second.

4.3 Technical Briefing of M*

Before introducing the SC-M* method, this section briefs the traditional MAPP problem and the M* algorithm [41].

4.3.1 MAPP Problem Definition

In this problem, we have m agents indexed by the set $I = \{1, \ldots, m\}$. Let the free configuration space of agent j be represented by the directed graph $G^j = \{V^j, E^j\}$. For any agent j, graph G^j is the same. The joint configuration space, which describes the set of all possible states of the multi-agent system, is represented by the *Tensor product* of the graphs of all individual agents: $G = G^1 \times \cdots \times G^m$. G consists of a *joint vertex* set V and a *joint edge* set E. As an example, in a 2-D joint configuration space given by the agents j and k, the two 2-D joint vertexes $v_p = (v_p^j, v_p^k)$ and $v_q = (v_q^j, v_q^k)$ is connected by the joint edge (e_{pq}^j, e_{pq}^k). Note that $v_p^j \in V^j$ and $e_{pq}^j \in E^j$. Let $\pi^j(v_p^j, v_q^j)$ denote a sequence of joint vertexes, called a *path* in G^j from v_p^j to v_q^j. The cost of a path $\pi(v_p, v_q)$ in G is defined as

$$g(\pi(v_p, v_q)) = \sum_{j=1}^{m} g(\pi^j(v_p^j, v_q^j)), \tag{4.1}$$

where $g(\pi)$ is the sum of all edge costs involved in the joint path π.

The problem of MAPP is to find a collision-free path, which is optimal with respect to minimal cost, from the source configuration $v_s = v_s^1 \times \cdots \times v_s^m$ to the destination configuration $v_d = v_d^1 \times \cdots \times v_d^m$. To determine the collision between agents, a collision function $\psi(v_p)$ is defined to return the set of conflicting agents at v_p.

Most fundamental MAPP approaches usually default hard collisions that no intersection is allowed between every two agents in terms of the occupation of any *resource*, such as a workspace. This implies that the capacity of each resource can support only one agent at a time (i.e., a collision happens immediately once agents intersect at any resource). Suppose we have a set of resources $A = \{A_1, \ldots, A_L\}$ requested by each agent in the multi-agent system, where A_i is the set of resource of Type i on all edges or vertexes and it is a continuous set because only continuous resources are considered in the thesis. A traditional hard-collision-free constrained MAPP problem can be formulated as follows:

$$\begin{aligned} &\min_{\pi} \; g\left(\pi\left(v_s, v_d\right)\right) \\ &s.t. \\ &\bigcup_{\forall i \neq j \in I} \left(A_k(v_p^i) \cap A_k(v_p^j)\right) = \emptyset, \; \forall A_k \in A, \; \forall v_p \in \pi, \end{aligned} \tag{4.2}$$

where $A_k(v_p^j)$ denotes the subset of resource A_k occupied by the agent j at the joint vertex v_p. One state-of-the-art solver to this problem is M*, which uses the sub-dimensional expansion

strategy to dynamically increase the dimensionality of the search space in regions featuring some agent collisions. M* enables a relatively cheaper graph search under the strict hard-collision-free constraint.

4.3.2 Graphic-Centric Description of M*

This section uses the graphic-centric description introduced by [41] to illustrate M*. M* is a complete and optimal MAPP algorithm. The main idea of M* is to iteratively construct and update a so-called *search graph* G^{sch} (i.e., to iteratively remove the collision configuration vertexes and expand necessary neighbors) and apply the A* algorithm on the new G^{sch} until the optimal collision-free path to v_d exists in the G^{sch} and is found by the A* search. Specifically, G^{sch} is a sub-graph of G and consists of three other sub-graphs: the *expanded graph* G^{exp}, *neighbor graph* G^{nbh}, and *policy graph* G^{ϕ}. The expanded graph G^{exp} is the sub-graph of G that has been explored by M*. G^{nbh} contains the *limited neighbors* of all the joint vertexes in G^{exp}. The definition of limited neighbors will be given later. G^{ϕ} consists of the paths induced by the *individual optimal policy* ϕ that connects each joint vertex in $G^{nbh} \cup G^{exp}$ to v_d without the collision-free constraint. Specifically, ϕ^j is the individual optimal policy for the agent j that leads any v^j in $G^{nbh} \cup G^{exp}$ to v_d^j without considering collisions. Examples of policy ϕ include the standard *Dijkstra's* algorithm and A*. Using the above graphic concepts, we can define the *collision set* C_p as

$$C_p = \begin{cases} \psi\left(v_p\right) \bigcup \{\bigcup_{v_q \in \boldsymbol{V_p}} \psi\left(v_q\right)\}, \ for\ v_p \in G^{exp} \\ \varnothing \qquad\qquad\qquad\qquad\qquad\quad , \ for\ v_p \notin G^{exp} \end{cases}, \tag{4.3}$$

where $\boldsymbol{V_p} = \{v_q | \exists \pi(v_p, v_q) \subseteq G^{exp}\}$ is the set of the joint vertexes to which there exists a path from v_p in G^{exp}. Let $\phi^j(v^j)$ be the *successor vertex* of v^j in the policy path, then the *limited neighbors* V_p^{nbh} of the joint vertex v_p in G^{nbh} are defined as

$$V_p^{nbh} = \left\{ v_q \middle| \begin{cases} e_{pq}^j \in E^j, \ for\ \ j \in C_p \\ v_q^j = \phi^j\left(v_p^j\right), \ for\ j \notin C_p \end{cases} \right\}, \tag{4.4}$$

where $e_{pq}^j = edge\left(v_p^j, v_q^j\right)$. The definition of the limited neighbors implies the sub-dimensional expansion strategy: We only expand the search space at the dimensions where the collision occurs ($j \in C_p$), otherwise for collision-free dimensions ($j \notin C_p$), M* will not expand, confining the unexpanded search space to the individual optimal path induced by the policy ϕ.

4.3.3 Algorithm Description of M*

The high-level description of M* is as follows [41]: Initially, M* computes the individual optimal policy ϕ for each agent from source v_s to destination v_d. The initial search graph G^{sch} only consists of an individual optimal path: Initial G^{exp} contains v_s only; initial G^{nbh} contains $\phi(v_s)$ only, which is the successor of v_s along the individual optimal policy; and initial G^{ϕ} contains the optimal policy path from the vertex in G^{nbh} and G^{exp} all the way to v_d. $C_p = \emptyset$ for all v_p in initial G^{sch}. Given the initial G^{sch}, the A* algorithm is applied using the following admissible heuristic

$$h\left(v_p\right) = g(\pi_{\phi}(v_p, v_d)\ ) \leq g(\pi_*(v_p, v_d)), \tag{4.5}$$

where π_{ϕ} is the individual optimal path induced by policy ϕ, and π_* is the ground-truth optimal multi-agent path we want to find. The initial *open list* (aka. *priority queue*) contains v_s only, with zero cost. The open list is ranked according to $v_p.cost + h\left(v_p\right)$, where $v_p.cost$ is the current cost of v_p from the source.

In each iteration, M* expands the first-ranked vertex v_p from the open list to G^{exp} and investigates each joint vertex v_q in the limited neighbors of v_p (i.e., $v_q \in V_p^{nbh}$) if no collision occurs at v_p; otherwise, it jumps to the next iteration. If there exists a collision (i.e., $\psi\left(v_q\right) \neq \emptyset$), M* will update the collision set C_q with $C_q \cup \psi\left(v_q\right)$, and this update will back-propagate from v_q to its immediate predecessor v_p AND all the way back to any ancestors that have at least one path inside of G^{exp} leading to v_q (see Eq. 4.3 for details). After this pre-processing, the algorithm

- Investigates and updates the cost of the vertex v_q and records its corresponding predecessor; and
- Adds v_q and all its predecessors/ancestors, of which the collision sets are changed, to the open list.

This process is repeated until v_d is expanded.

The critical point is that: only when a collision set C_p is changed will the search graph G^{sch} change. It is the operation of updating the collision set in a back-propagation way that makes the story different: By including $\psi\left(v_p\right)$ to C_p, M* can tell which agents are *immediately* collided at the current v_p; by including all $\psi\left(v_q\right)$ for $v_q \in V_p$ to C_p (i.e., the collision information of all the expanded downstream successors from v_p), M* can preview which agents will collide in the future, making it possible to pre-plan to avoid that. Therefore, using the limited neighbor

set in Eq. 4.4 makes sense: It advises M* to only expand the dimensions where there exists an immediate collision at v_p OR there will be collisions in the future, starting from v_p, in the current expanded graph G^{exp}. See Figure 4.1 for an example of how M* solves the optimal collision-free path planning for the two agents.

From Figure 4.1, we can visualize the evolution of the search graph G^{sch} of Agent 2, which consists of an expanded graph G^{exp} (circle), a neighbor graph G^{nbh} (diamond), and a policy graph G^{ϕ} (square). Edge cost and direction-changing cost are considered during planning. Yellow zones are preferred areas with lower edge cost. In M*, individual optimal paths are induced by ϕ for each individual agent (Figure 4.1.a). We can observe that there will be a collision at vertex s_{10}, which is ignored by ϕ. For Agent 2, M* searches in the subspace, and the most promising vertex is expanded at each iteration (Figure 4.1.b and Figure 4.1.c). Then, a collision occurs at vertex s_{10} and triggers the removal of the rest of G^{sch} (Figure 4.1.d), which is equivalent to jumping to the next iteration. Following the sub-dimensional expansion strategy, M* extends the search space to include the limited neighbors, and a new G^{sch} is obtained (Figure 4.1.e). By searching in the new G^{sch}, M* finds the optimal collision-free path for Agent 2 (Figure 4.1.f and Figure 4.1.g). On the other hand, the planning for Agent 1 is conducted simultaneously, and finally, the collision-free optimal paths for both agents are found by M* (Figure 4.1.h).

Figure 4.2 shows an example of applying M* for four agents, where the A* individual optimal planning causes Agent 2 and 3 to collide with Agent 1 along their individual optimal paths, while only Agent 4 is free from collision. To handle the collisions, M* adjusts the paths of Agent 2 and 3 to go upward and rightward, respectively at the first step. In this way, newly planned paths are collision-free and all the four agents' mobility demands are accommodated. Agent 4 maintains its optimal path.

4.3.4 M* Challenges and Improvements

Using the sub-dimensional expansion strategy, M* gains some planning efficiency in terms of success rate and run time while maintaining the cost optimality. Generally, in a typical four-connected grid world with 32x32 cells, M* can handle fewer than 40 agents, compared to 10 with the standard A* [41]. However, the curse of dimensionality (CoD) challenge remains an issue for M*. There are several bottlenecks blocking M* from further scalability. Some of them were researched and mitigated.

The first bottleneck is rooted in the property of M* to accumulate the collision set from

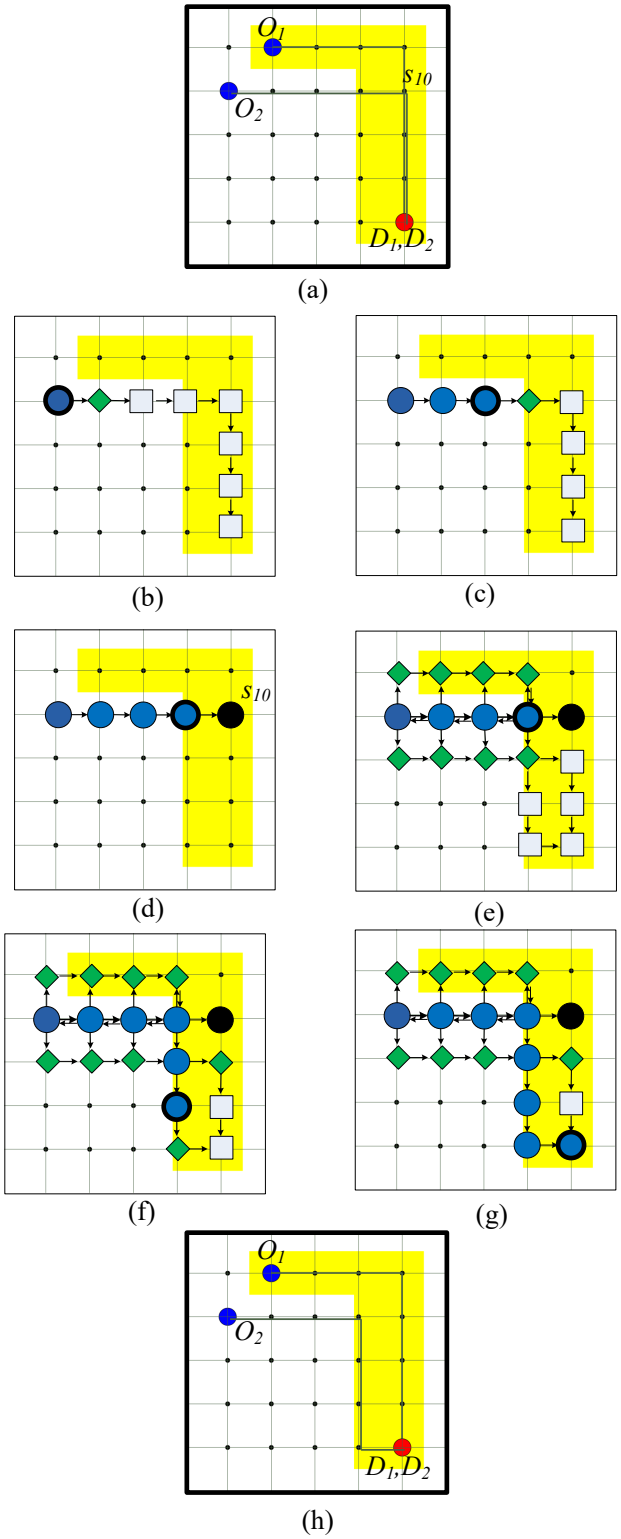

Figure 4.1: Illustration of traditional M* for two agents, where we show the evolution of the expanded graph G^{exp} (circle), neighbor graph G^{nbh} (diamond), and policy graph G^{ϕ} (square) for Agent 2 as the M* algorithm proceeds.

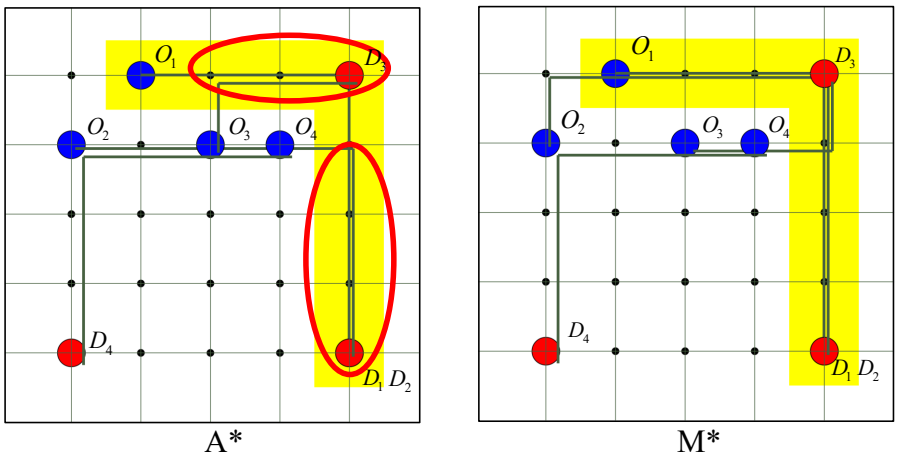

Figure 4.2: Planning for four agents with origins (O_1, O_2, O_3, O_4) and destinations (D_1, D_2, D_3, D_4).

the current joint vertex back to all its predecessors/ancestors in the expanded graph. This property guarantees the completeness and optimality of M* but on the other hand incurs a surge in the size of the search graph when collisions among too many agents happen. Previous approaches to this issue include the recursive M* (rM*) [39, 41], which decouples the colliding agents into multiple disjoint subsets. Each subset contains the mutually colliding agents only. In this way, rM* can find a path for each independent set of mutually colliding agents in a lower dimension and use the resulting paths to constrain the exploration in future iterations. This strategy brings in efficiency – the computation is exponential with respect to the size of the largest set of mutually colliding agents, instead of the total number of colliding agents. Around 10 more agents can be handled using rM* than using the basic M*.

The second bottleneck of the M* lies in using A* as the basic component in searching the high-dimensional graph G^{sch}, which adds all the *out-neighbors* of the expanded vertexes to the open list, limiting the efficiency of planning. Replacements of A* for searching in the search space can bring some efficiency to M*. Successful examples include EPEM* and ODM*, which replace the A* component in M* with the EPEA* [43, 53] and OD [42], respectively. Integration of the rM* technique to these algorithms results in advanced variants, EPErM* and ODrM*. Around 30 to 40 agents can be handled by using these methods.

The third bottleneck is related to the choice of individual policies when constructing the policy graph G^{ϕ}. The performance of M* is very sensitive to the individual policies because more than one optimal path may exist for each agent. Improper choice of individual policies will result in a large number of agents colliding at one vertex, preventing a solution from being found in a reasonable run time. Researchers have made great efforts to develop *policy optimization* for

optimizing the choice of individual policies before starting an M* search. The MA-CBS algorithm [46] is a good fit for this role because it iteratively adds constraints on individual agents, rather than the joint configuration space, and searches for paths consistent with the set of constraints, until the destination configuration is reached without any collisions. Using MA-CBS as the policy generator and combining it with ODrM* leads to the MA-CBS+ODrM* algorithm [47], which achieves the state-of-the-art performance to handle more than 60 agents.

The final bottleneck is due to the fact that the collision-free constraint is too restrictive for many applications. Our contribution in the thesis is to mitigate the fourth bottleneck by introducing the perspective of soft collisions and show that proper slackness in the collision constraint (i.e., the soft-collision-free constraint) is rewarding without losing much of the optimality.

4.4 Soft-Collision-Free M* (SC-M*)

In this section, we formulate the concept of soft collisions, generalize M* for planning in the soft-collision context, and extend our approach to a more complex environment with multiple types of agents requesting multiple types of resources.

4.4.1 Soft-Collision-Free Constraint

Inspired by some real-world scenarios, we introduce the soft collision property to the model of an agent and develop the soft-collision-free M* (SC-M*), a generalized version of M* to efficiently solve the MAPP in the soft-collision context.

We define that all the agents have the following properties: 1) a collision among agents is soft, quantified using some collision scores and 2) different agents have different collision scores, according to their individual experiences through the paths. We suppose that each agent cares about a set of resources $A = \{A_1, \ldots, A_L\}$. To obtain the properties, we introduce to each agent an additional attribute called *resource experience* (for each resource) and use the resource experience to calculate the *collision score*.

In doing so, this section first uses the *resource experience* (will be defined in Definition 1), to indicate how unpleasant the agent is about the resource allocated to it. Then, we combine this information of all the resources into a *collision score* (will be defined in Definition 2), that indicates the probability of the agent announcing a collision given its resource experience. Thresholds of collision are used to limit the collision score, implying to what degree of unpleasant we want to pursue the solution. The agent, of which the collision score exceeds the threshold, will be

placed into a soft-collision set via the *soft collision function* for sub-dimensional expansion (will be defined in Definition 3).

Definition 1 (Resource Experience)

We use *resource experience* to quantify the dissatisfying experience per resource about which an agent cares.

Let

- $\pi = \pi\left(v_s, v_b\right)$ be a path from the source v_s to some v_b;
- $v_q = \pi\left(v_p\right)$ be the immediate successor of v_p along the path π;
- $A_k(e_{pq}^j)$ be the capacity (amount) of the subset of the resource A_k on the edge e_{pq}^j, given by the graph model;
- $A_k^j(e_{pq}^j)$ be the amount of the subset of the resource A_k *actually allocated* to the agent j on the edge e_{pq}^j, called the *allocated resource value*.

The *resource experience* is then defined as the *dissatisfying* experience of agent j on resource A_k along the path π^j:

$$D\left(\pi^j, A_k\right) = \sum_{v_p | v_p \in \pi_{/v_b}} \mathbf{1}\left(A_k(e_{pq}^j) \geq \varepsilon_k \wedge A_k^j(e_{pq}^j) < \varepsilon_k\right) \cdot g(e_{pq}^j), \tag{4.6}$$

where $\mathbf{1}(\cdot)$ is the indicator function; $\varepsilon_k \in \varepsilon = \{\varepsilon_1, \ldots, \varepsilon_L\}$ is the *satisfying value* regarding the resource A_k, which is a positive real value; $g(e_{pq}^j)$ is the edge cost regarding travel time/distance given by the graph model; $A_k^j(e_{pq}^j)$ is formulated as:

$$A_k^j(e_{pq}^j) \ = \frac{A_k(e_{pq}^j)}{\sum_{k \in I} \mathbf{1}\left(e_{pq}^k = e_{pq}^j\right)}. \tag{4.7}$$

Obviously, $A_k^j(e_{pq}^j) = A_k(e_{pq}^j)$ if and only if no other agents are physically moving along with agent j on the edge e_{pq}^j. The allocated resource value $A_k^j(e_{pq}^j)$ quantifies the level of interference incurred by other agents when they physically move together. In contrast, the traditional hard-collision setting will always label a collision to the agent j and all other involved agents whenever $A_k^j(e_{pq}^j)$ is (even slightly) smaller than $A_k(e_{pq}^j)$. The resource experience is used as an attribute of a joint vertex and can be calculated incrementally using Algorithm 3.

Algorithm 3 Function: experience(v_k, v_l, A)

Input: v_k: base vertex;
v_l: immediate successor of the base;
A: list of resource
Output: v_l with updated experience
1: **for** A_p **in** A **do**
2: **for** j **in** I **do**
3: $v_l.\exp[A_p][j] \leftarrow v_k.\exp[A_p][j] + D\big(\pi(v_k, v_l)^j, A_p\big)$
4: **endfor**
5: **endfor**
6: **return** v_l //the successor with updated experience

Combined with the allocated resource value, which serves as a proxy of the interference level, the definition of resource experience in Eq. 4.6 actually defines a property of an agent: Only the situation in which the resource allocated to an agent is dissatisfying because of the co-existence of other agents, will contribute to the dissatisfying experience of that agent. Furthermore, each dissatisfying condition is weighted by the edge cost $g(e^j_{pq})$. In this way, we can quantify the resource experience in terms of how long such a dissatisfying condition lasts in travel time or distance. As will be discussed later, the resource experience of an agent will determine its collision score, which is defined from a probabilistic point of view.

Definition 2 (Collision Score)

We use the resource experience results from Definition 1 to calculate the *collision scores*. This is defined from the view point of collision probability, that must be constrained below some threshold.

Let

- Col_j be the event that agent j announces a collision (i.e., when agent j calls for one of the resources, the allocated resource is less than satisfying);
- $D^j = \{D^j_1, \ldots, D^j_L\}$, where $D^j_k = D\left(\pi^j, A_k\right)$, be the set of dissatisfying experiences of agent j along path π^j on the resource A_k;
- $f_k \in f = \{f_1, \ldots, f_L\}$ be a customized cumulative distribution function (CDF) defined on $[0, +\infty)$, mapping the resource experience D to a probability of collision on the resource A_k.

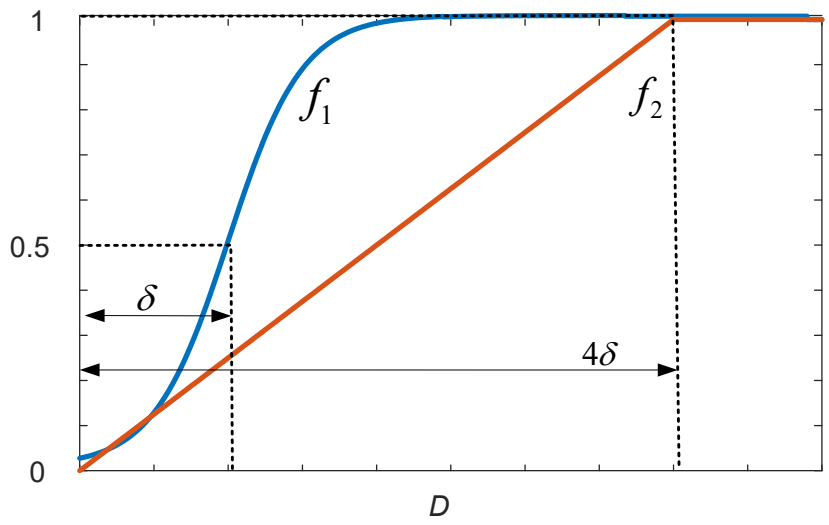

Figure 4.3: Example designs of CDFs, mapping the resource experience of an agent to a collision probability on certain resource. f_1: sigmoid-based CDF for important (sensitive) resources. f_2: linear CDF for trivial (insensitive) resource. δ: offset parameter adjusting the *tolerance level*.

The *collision score* of the agent j is defined as the probability of how likely a collision occurs to the agent j on at least one of the resources given its resource experience D^j:

$$P\left(Col_j \middle| D^j\right) = 1 - \prod_{k \in \{1,\ldots,L\}} \left(1 - f_k(D_k^j)\right). \tag{4.8}$$

Note that $P\left(Col_j \middle| D^j\right)$ calculates the *complement* of the success probability – the joint probability of being collision-free on all resources.

Figure 4.3 shows two example designs of f. $f_1(D) = sigmoid(D - \delta)$, with a discontinuity point $f_1(0) = 0$, is a sigmoid-based CDF function, featuring a surge in the collision score (the derivative is bump-shaped) at the experience value around δ. This function is suitable to important resources that are sensitive to the agent. $f_2(D) = min(1, D/(4\delta))$ is a linear CDF with a shallow slope (the derivative is flat). This function can apply to trivial resources that are not very sensitive to the agent but still accumulate to contribute to the collision score. We use the offset parameter δ to adjust the *tolerance level* of the dissatisfying experience. With larger δ, the agent will tolerate a longer unpleasant experience before announcing a collision.

Although the definition of the collision score can be customized according to different practices, the probabilistic definition of collision score introduced here is a general one: Different types of resources may have different value ranges, and Eq. 4.8 standardizes the resource ranges, mapping them to a value between $[0, 1]$ and enabling an efficient integration of different types of resources to the framework.

Definition 3 (Soft Collision Function $\widetilde{\psi}$)

Now, according to the collision scores from Definition 2, we want to pick out the above-threshold agents and place them into the soft-collision set via the *soft collision function* for the purpose of applying the sub-dimensional expansion.

Given a path $\pi = \pi(v_s, v_b)$ and corresponding resource experience D^j for the agent j, the *soft collision function* of the agent is

$$\widetilde{\psi}^j(v_b) = \begin{cases} \{j\}, & for\ P\left(Col_j \middle| D^j\right) \geq T \\ \emptyset, & otherwise \end{cases}, \tag{4.9}$$

where T is the *threshold of collision*. The definition of the *global soft collision function* is then defined as

$$\widetilde{\psi}(v_b) = \bigcup_{j \in I} \widetilde{\psi}^j(v_b). \tag{4.10}$$

Based on Definition 3, we can formally construct the soft-collision-free constraint and obtain the soft-collision-free constrained MAPP problem:

$$\begin{aligned} &\min_{\pi} g(\pi(v_s, v_d)) \\ s.t. \quad & \\ &\widetilde{\psi}(v_p) = \emptyset, \quad \forall v_p \in \pi. \end{aligned} \tag{4.11}$$

SC-M* is a general solver to this problem by adjusting M* to the soft-collision-free constraints. The pseudocode for SC-M* is presented in Algorithm 4, where critical commands relative to the soft-collision-free constraint are underscored. In this algorithm, lines 1 to 7 initialize each vertex v in the vertex set V with infinite cost from the source v_s (the cost of v_s itself is zero), zero dissatisfying experience and empty collision set C_k. The initial open list contains v_s only (line 8). In each iteration, SC-M* expands the first-ranked vertex v_k in the open list ordered by the total cost $v_k.cost + heuristic[v_k]$ (lines 10 to 11). The algorithm terminates and returns the result if the expanded v_k is the destination v_d (lines 12 to 14) or jumps to the next iteration if immediate collision occurs at v_k (i.e., $\widetilde{\psi}(v_k) \neq \emptyset$ [lines 15 to 17]). Line 18 constructs the *limit neighbors* V_k^{nbh} of v_k using Eq. 4.4. For each vertex v_l in V_k^{nbh} (line 19), it adds v_l to the descendant set V_k of v_k (line 20), updates the dissatisfying experience of v_l using Algorithm 3 (line 21), and merges the immediate collision at v_l to its soft-collision set C_l (line 22). On top of the new collision set of v_l,

Algorithm 4 Soft-collision-free M*

Input: v_s: source joint vertex; v_d: destination joint vertex;
$\{V, E\}$: joint configuration graph; A: list of resources
Output: Path finding results

1: **for** all v_k **in** V **do**
2: v_k.cost← $+\infty$
3: v_k.exp← all zero experience
4: $C_k \leftarrow \emptyset$
5: v_k.traceBack← $\emptyset$
6: **endfor**
7: v_s.cost← 0
8: open ←$\{v_s\}$
9: **while** open≠ $\emptyset$ **do**
10: open.sort() by v.cost+heuristic[v]
11: $v_k \leftarrow$ open.pop()
12: **if** $v_k = v_d$ **then** //optimal path found
13: **return** back_track_path[v_k]
14: **endif**
15: **if** $\tilde{\psi}(v_k) \neq \emptyset$ **then**
16: **continue** //skip the vertex in Collisions
17: **endif**
18: construct V_k^{nbh} using Eq. 4.4
19: **for** v_l **in** V_k^{nbh} **do**
20: add v_l to V_k //note $V_k = \{v_q | \exists \pi(v_k, v_q) \subseteq G^{exp}\}$
21: v_l←experience(v_k, v_l, A) //update experience using Algorithm 3
22: $C_l \leftarrow C_l \cup \tilde{\psi}(v_l)$
23: backpro_update(v_k, C_l, open)
// update all affected soft-collision sets using Eq. 4.3
//add all affected vertexes back to open list [40]
24: **if** $\tilde{\psi}(v_l) = \emptyset$ **and** v_k.cost+e_{kl}.cost< v_l.cost **then**
25: v_l.cost← v_k.cost+e_{kl}.cost
26: v_l.traceBack← v_k
27: open.add(v_l)
28: **endif**
29: **endfor**
30: **endwhile**
31: **return** no path exists

SC-M* backpropagates to update all the affected ancestor vertexes from v_l (see Eq. 4.3) and adds them back to the open list for re-expanding (line 23). After this collision set updating operation, if v_l is free from collision and has improved cost, the algorithm accepts the new cost by save the trace-back information and adding v_l to the open list for expansion (lines 24 to 28). This process repeats until the open list is empty (line 9) when no solution exists or the optimal solution is found (lines 12 to 14).

SC-M* can make a transition from a decoupled individual A* ($T = 1$) to a standard hard-collision-free constrained M* ($T = 0$), providing more flexibility to the performance of the algo-

rithm with bounded soft-collision scores.

4.4.2 Completeness and Cost-Suboptimality

A MAPP algorithm is complete if it guarantees that it will either return a path, or determine that no path exists in finite time. An algorithm is optimal if it guarantees returning an optimal path if such a solution exists. SC-M* is complete and suboptimal conditioned on the soft-collision-free constraint (i.e., $P\left(Col_j \middle| D^j\right) < T$, for a given collision threshold T).

Completeness

SC-M* inherits the sub-dimensional expansion from M* (i.e., it changes the G^{sch} only when one of the soft-collision sets C_p changes). The algorithm applies A* in the updated search graph. Due to the merging operation applied to collision set C_p, as shown in Eq. 4.3, C_p for each vertex will change finite times (at most m times, which is the number of agents). Because A* is complete, applying A* to a given G^{sch} takes finite time to return a result. Therefore, SC-M* is complete.

Cost-Suboptimality

Different from M* which is optimal, SC-M* is suboptimal because Eq. 4.9 and Eq. 4.10 only include the immediate conflicting agents to the soft-collision set; the agents that softly interfere with the conflicting agents in the upstream path are excluded. Those excluded agents also contribute to the announced collision (i.e., the collision score is above the threshold). Because of this property, SC-M* cannot guarantee the *inclusion property*, which is the basis to ensure the optimality in M* [41]: *The optimal path for some subset of agents costs no more than the joint path taken by those agents in the optimal joint path for the entire set of agents*. Because of losing the inclusion property, SC-M* may not guarantee cost optimality.

Figure 4.4 provides a counterexample of the inclusion property of SC-M* under the soft-collision MAPP context. Let ${\pi'}_{\Omega}(v_k, v_f)$ be the joint path constructed by combining the optimal path for a subset $\Omega \in I$ of agents with the individual optimal paths for the agents in $I \backslash \Omega$. The inclusion property is defined as follows: If the configuration graph contains an optimal path $\pi_*(v_k, v_f)$, then $\forall \Omega \subset I$, $g({\pi'}_{\Omega}(v_k, v_f)) \leq g(\pi_*(v_k, v_f))$. See *Lemma 6* in [41].

Under the soft-collision context, this inclusion property does not always hold. In Figure 4.4, we have a three-agent MAPP problem ($I = \{r1, r2, r3\}$) under the soft-collision context. The agents $r1$, $r2$, and $r3$ attempt to move from the vertex a, f, and h to the vertex e, g, and i,

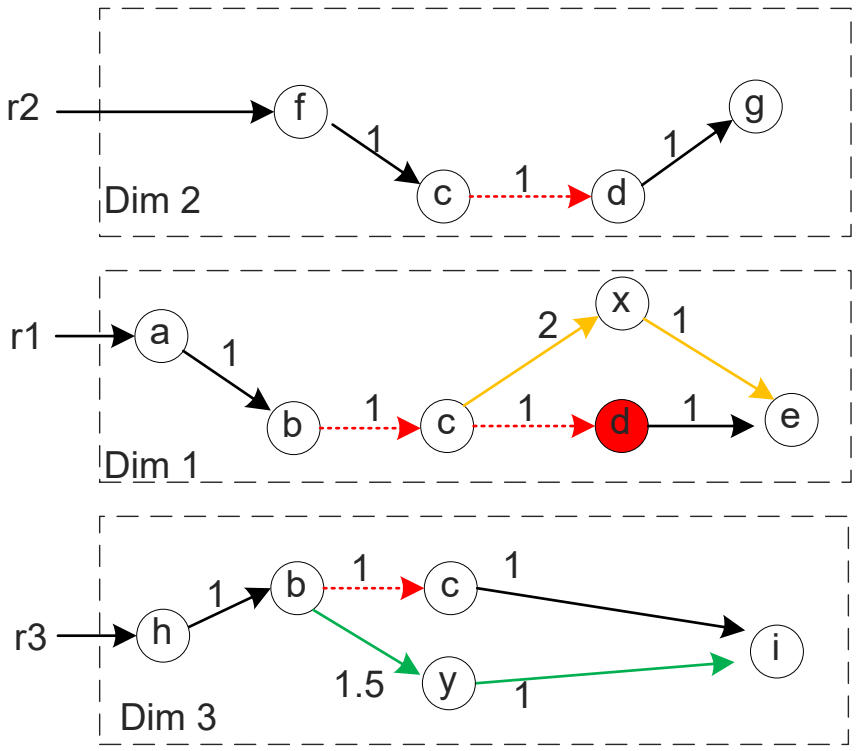

Figure 4.4: Counterexample of the inclusion property of SC-M* under the soft-collision context. Agent $r1, r2$, and $r3$ have the planning O-D demands (a, e), (f, g), and (h, i), respectively. Vertexes in the system are labeled as a, b, c, etc.

respectively. The individual optimal paths (lowest distance) are $a \to b \to c \to d \to e$ with distance 4 for $r1$, $f \to c \to d \to g$ with distance 3 for $r2$, and $h \to b \to c \to i$ with distance 3 for $r3$. The total cost of the joint individual optimal path is 10. However, assume that the agent can only tolerate a dissatisfying experience with distance 1, $r1$ will announce a collision at the vertex d because of the interference on the edge $b \to c$ and $c \to d$ from agent $r3$ and $r2$, respectively.

If we choose $\Omega = \{r1, r2\} \in I$, as can be seen in Figure 4.4, the only solution would be that $r1$ takes a detour through the vertex x to avoid the collision on the edge $c \to d$, resulting in a cost of 5 for $r1$, and the total $g(\pi'_\Omega(v_k, v_f))$ is 11 (3 for $r2$, 5 for $r1$ and 3 for $r3$). On the other hand, by searching through all three dimensions, a better solution would be that $r3$ detours through the vertex y, and $r1$ is free from collision because the interference on the edge $b \to c$ disappears. The total cost of this joint path is 10.5, and we have $g(\pi'_\Omega(v_k, v_f)) = 11 > g(\pi_*(v_k, v_f)) = 10.5$, which is contradictory to the inclusion property.

The reason for this phenomenon is that under the hard-collision context, only the immediate conflicting agent $r2$ contributes to the collision of $r1$ at the vertex d. However, under the soft-collision context, both $r2$ and $r3$ contribute to the collision of $r1$ at the vertex d, and thus, the

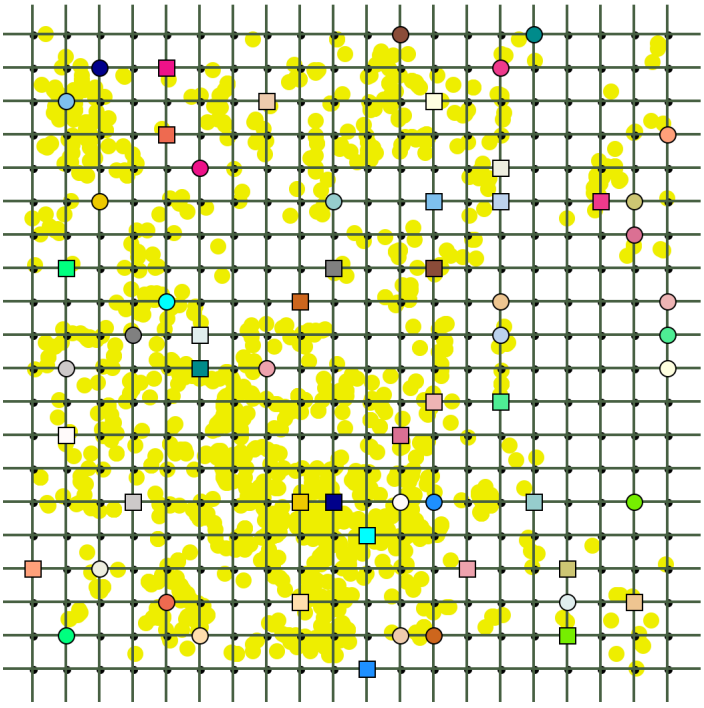

Figure 4.5: Grid system with 20x20 stops and 40 bidirectional lines.

inclusion property does not apply. Without this inclusion property, which is the basis of the optimality of M*, the optimality of SC-M* cannot be guaranteed.

However, we notice that suboptimal methods have long been used successfully to solve many interesting MAPP problems [54, 55, 48]. Given the fact that we will show in the next section that SC-M* is superior to other alternative SC-based MAPP solvers (e.g., SC-A* and SC-CBS) in terms of scalability, run time, and path cost, we believe that the proposed method, which is adjusted to MAPP under the soft-collision context, is a powerful tool in practice.

4.5 Experiments and Results

We evaluate SC-M* in simulation on a grid public mass transit network with an Intel Core i7-6700 CPU at 3.4 GHz with 16 GB RAM. As shown in Figure 4.5, the grid transit environment has 20x20 stops. There are 20 bidirectional horizontal lines. Likewise, 20 bidirectional vertical lines are deployed in the environment. At each stop, agents can switch lines. The yellow areas are covered by some resources, such as the on-vehicle free Wi-Fi in our experiments. Agents traversing those areas can enjoy high-quality on-vehicle Wi-Fi connections. A fully covered edge has a Wi-Fi resource value of 100, and the Wi-Fi value of an edge is proportional to the length of coverage. Each agent wants to move from its source (square) to its destination (circle) with

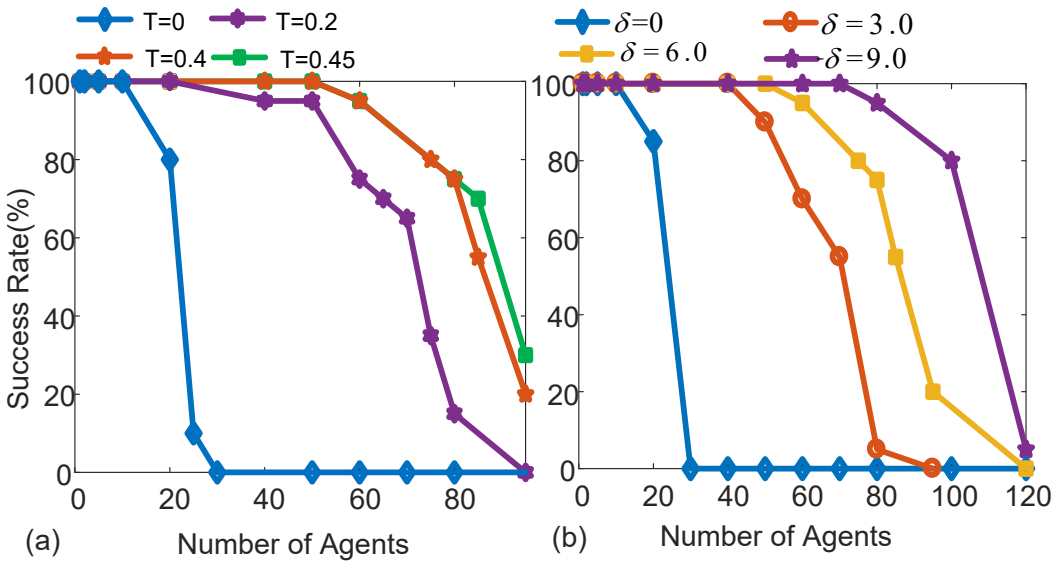

Figure 4.6: Impact of the collision threshold T (given $\delta = 6.0$) and offset parameter δ (given $T = 0.35$) on one-resource-one-type SC-M*.

m	T=0	T=0.2	T=0.4	T=0.45
5	0.556389	0.52489	0.5472	0.3616
10	1.25143	1.18687	0.7057	0.7965
20	403.3011	2.72513	1.4871	1.4488
40	>1000	56.1898	4.2336	4.4318
70	>1000	370.059	257.59	255.78
95	>1000	1000	951.34	774.40

Left: $\delta = 6.0$.

m	δ=3.0	δ=6.0	δ=9.0
5	0.5060	0.3616	0.5750
10	1.0765	0.7965	1.0427
20	2.2578	1.4488	2.1034
40	17.201	4.4318	4.5250
80	477.96	292.17	59.310
120	>1000	>1000	857

Right: T=0.35

Table 4.1: Run time of one-resource-one-type SC-M* under different parameters.

the lowest cost (i.e., a linear combination of distance cost and Wi-Fi cost) as well as bounded collision score. The second resource is the space on the edge, which is fixed at 5. The satisfying values are $\varepsilon_1 = 20$ and $\varepsilon_2 = 1$ for Wi-Fi and space resources, respectively.

We randomly generate a source-destination pair for each agent. Each trial is given a 1000-second run-time limitation to find a solution. For each configuration (including the number of agents, collision threshold T, and offset parameter δ), we run 20 random trials to calculate the average metrics (i.e., the success rate and run time). We use the standard A* as the coupled planner and policy generator in the SC-M* framework and compare our results to the baselines without integrating any advanced A*s.

4.5.1 Planning for the One-Resource-One-Agent-Type

The first experiment considers Wi-Fi as the only resource requested by agents (i.e., $A = \{A_1 : "WiFi"\}$). Only one agent type exists, and all agents use sigmoid-based function f_1 as the collision CDF.

We first study the influence of the collision threshold T and the offset parameter δ on performance. Figure 4.6.a shows the success rate of the one-resource-one-type SC-M* with different thresholds T =0 (equivalent to the basic M*), 0.2, 0.4, and 0.45, while the offset parameter is fixed to $\delta = 6.0$. Table 4.1.Left shows the run time in seconds for the experiment. The results clearly show that larger thresholds bring improvement in performance with a higher success rate and lower run time for a large system size ($m > 50$). The improvement in performance results from the property of SC-M* that larger thresholds render more relaxed constraints, and thus, agents are less likely to collide on resources.

Figure 4.6.b shows the success rate of the SC-M* with different offset parameters $\delta = 0$, 3.0, 6.0, and 9.0, with fixed $T = 0.35$. Table 4.1.Right shows the run time for the experiment. The results illustrate that SC-M* is sensitive to δ and can efficiently handle up to 100 agents with δ=9.0. These results are reasonable because the sigmoid-based CDF is used in the experiments, featuring a surge in the collision probability at the experience value around the offset, and the offset parameter poses a cutoff value on the resource experience, with collision always announced once the resource experience is larger than the offset. The standard M* ($T = 0$) can only scale to fewer than 30 agents. Taking advantage of this property, one can tune the parameters to trade off the scalability against the tightness of constraints.

4.5.2 Planning for the Two-Resource-Two-Agent-Type

We also conduct other experiments with SC-M* in more complex environments: two agent types requesting two resources. This experiment considers both Wi-Fi and space capacity (i.e., $A = \{A_1 : "WiFi", A_2 : "Space"\}$). Type I agents use f_1 in Figure 4.3 as the collision CDF for the Wi-Fi resource, and the linear CDF f_2 for the space resource, implying that they treat Wi-Fi and space as important and trivial, respectively. On the other hand, Type II agents use f_1 for space and f_2 for Wi-Fi. Each agent has a 50% chance of being Type I. Both CDFs are adjusted using the same δ at each trial, as illustrated in Figure 4.3.

Figure 4.7.a shows the success rate of the two-resource-two-type SC-M* with different thresholds $T = 0$ (equivalent to the basic M*), 0.2, 0.35, and 0.45, and with a fixed offset parameter

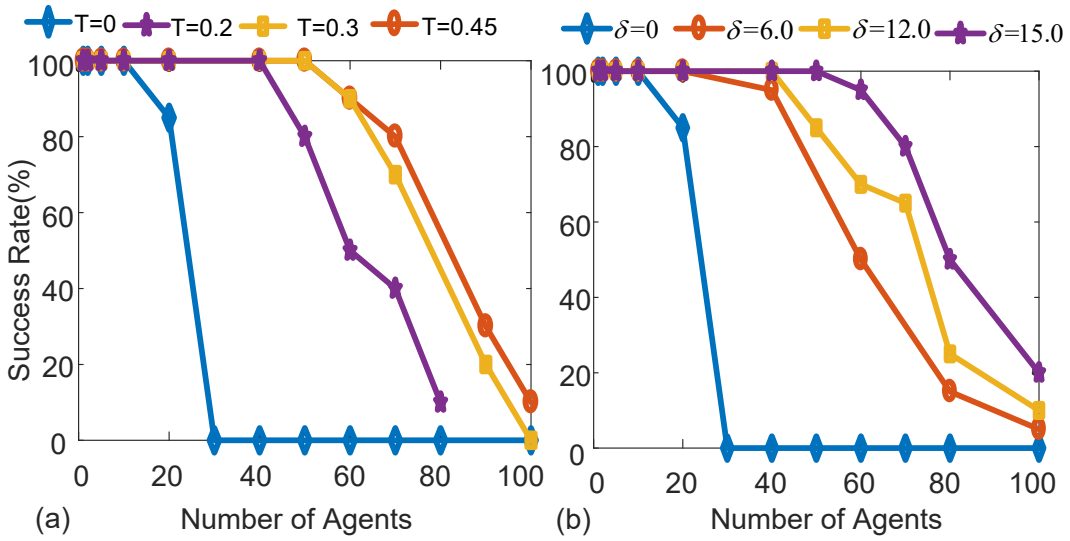

Figure 4.7: Impact of the collision threshold T (given $\delta = 9.0$) and offset parameter δ (given $T = 0.35$) on two-resource-two-type SC-M*.

m	T=0.2	T=0.35	T=0.45
5	0.3438	0.3493	0.363948
20	1.2485	1.8549	1.807993
40	10.102	3.2387	4.415021
60	503.94	106.02	104.6499
90	901.91	801.47	702.4526
100	>1000	909	901.1799

Left: δ=9.0.

m	δ=6.0	δ=12.0	δ=15.0
5	0.333498	0.3677	0.527464
20	1.479236	1.4032	2.369483
40	61.49792	4.5024	4.043310
60	521.2721	306.46	60.30944
70	627.0925	347.14.62	209.5725
100	901.91	751.78	606.6522

Right: T=0.35

Table 4.2: Run time of two-resource-two-type SC-M* under different parameters.

$\delta = 9.0$. Table 4.2.Left shows the run time for the experiments. As can be seen from the results, in general, SC-M* can handle the two-resource-two-type systems and plan for more than 80 agents. Because more resources contribute more factors to increasing the collision score, a relatively large offset (δ =9.0) is needed to achieve comparable performance to the one-resource-one-type SC-M*.

Figure 4.7.b and Table 4.2.Right present the impact of the offset parameter δ on performance. Different from the first experiment, SC-M* with the above configurations is less sensitive to δ, when compared to Figure 4.6. The reason is that 50% of the agents are insensitive to one of the resources because of the linear CDF f_2, and thus, increasing δ does not contribute to a significant reduction in collisions. This property implies that we can control the importance levels of resources efficiently through the design of collision CDFs. This experiment demonstrates that,

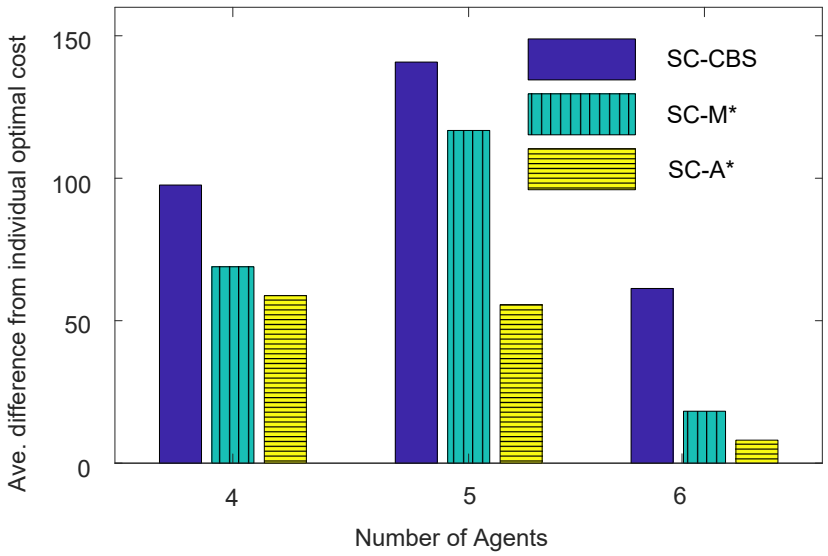

Figure 4.8: Average cost difference of SC-based MAPP solvers from the individual optimal cost under the one-resource-one-type context.

with the proper parameter settings, SC-M* can feasibly handle a complex environment with multiple resources and multiple agent types.

4.5.3 Comparison of SC-M* to Baselines

We now compare the SC-M* to other SC-based MAPP algorithms, including SC-A* (optimal) and SC-CBS (suboptimal), under the one-resource-one-type environment.

Path Cost

Firstly, we compare the path cost of the three algorithms. We design 18 planning tasks for environments with 4, 5, and 6 agents, in which agents will encounter at least one collision along the individual optimal paths under the $T = 0.05$, $\delta = 1$ setting. We start with small agent numbers because SC-A* cannot handle large agent numbers.

Figure 4.8 shows the average difference among the three SC-based solvers in terms of the individual optimal cost (i.e., the cost of the individual optimal path that ignores the constraint). We observe that SC-A* and SC-CBS have the lowest and highest additional cost, respectively. SC-M* solutions cost more than SC-A* but noticeably less than SC-CBS.

To be more detailed, in the experiments, we design 8, 5, and 5 MAPP tasks for environments with 4, 5, and 6 agents, respectively, in which all the tasks are designed to encounter at least one collision along the individual optimal paths under the abovementioned configuration. Thus,

		Diff. from individual optimal cost			Run-time		
m	index	SC-CBS	**SC-M***	SC-A*	SC-CBS	**SC-M***	SC-A*
4	1	67.09	**16.49**	16.49	1.62	**2.22**	4.22
4	2	48.39	**37.39**	37.39	1.93	**3.74**	18.64
4	3	18.69	**18.69**	17.59	0.77	**1.32**	5.37
4	4	320.09	**183.59**	183.59	4.87	**2.98**	19.39
4	5	240.89	**209.99**	178.19	2.53	**1.66**	310.96
4	6	14.29	**14.29**	4.39	1.15	**0.93**	6.00
4	7	49.49	**49.49**	27.49	1.21	**2.05**	8.62
4	8	21.99	**21.99**	5.49	1.66	**1.07**	5.59
5	1	328.79	**258.39**	41.79	1.48	**1.10**	6.02
5	2	56.09	**50.59**	50.59	2.63	**2.06**	60.23
5	3	128.69	**91.29**	83.59	1.42	**4.73**	1581.30
5	4	103.39	**96.79**	96.79	2.05	**5.03**	714.77
5	5	86.89	**86.89**	5.49	1.39	**3.61**	6.43
6	1	31.89	**9.89**	9.89	1.30	**2.26**	388.99
6	2	51.69	**51.69**	1.09	1.59	**1.83**	699.98
6	3	18.69	**13.19**	13.19	1.26	**2.50**	54.32
6	4	16.49	**5.49**	5.49	2.79	**4.12**	304.52
6	5	188.09	**10.99**	10.99	2.76	**4.12**	502.34

Table 4.3: Results of the cost optimality/suboptimality experiments.

additional costs from the individual optimal path are expected upon the three SC-based MAPP solvers. The detailed results are presented in Table 4.3, where we can observe that the additional cost of SC-M* from the individual optimal cost is consistently bounded between SC-A* (lower bound) and SC-CBS (upper bound). The run time of the SC-M* is longer than but competitive to that of SC-CBS and significantly faster than the SC-A*.

The reason for the result is that SC-A* is an optimal solver for this type of MAPP problem because it always explores cheaper paths in the entire multi-agent joint space before considering the paths that cost more [56]. SC-M* is suboptimal because of the process discussed in Section 4.3.2. Compared to SC-M*, SC-CBS suffers from more path cost due to the way it collects a collision: CBS collects collisions into a *conflict tree* and makes the collision into the form (agent j, vertex v, step s), indicating that the agent j collides at the vertex v at step s. In each iteration, CBS conducts decoupled planning to avoid the agent j reaching the vertex v at step s. This is unsuitable in the soft-collision context because there might exist another path that leads j to the vertex v at step s without announcing a collision, by avoiding one of the upstream vertexes

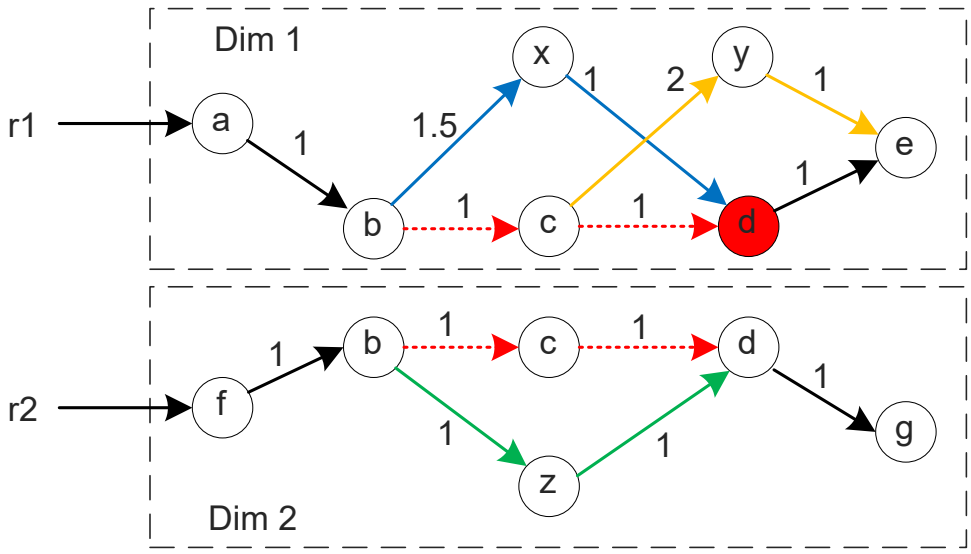

Figure 4.9: Illustration of the difference in planning among SC-A*, SC-M*, and SC-CBS.

involved in soft interference. In contrast, SC-M* can explore those paths excluded by SC-CBS because it searches the entire space of the immediate colliding agents. Figure 4.9 provides an example to visualize the difference in planning among the three SC-based MAPP solvers.

Figure 4.9 shows a two-agent MAPP problem under the soft-collision context. The agents $r1$ and $r2$ attempt to move from the vertex a and f to e and g, respectively. The individual optimal paths (shortest distance) for both agents are $a \to b \to c \to d \to e$ with distance 4 for $r1$ and $f \to b \to c \to d \to g$ with distance 4 for $r2$, respectively. The total cost of the joint individual optimal path is 8. $r1$ and $r2$ softly collide on the edge $b \to c$ and $c \to d$, where $r2$ can tolerate the dissatisfying experience with distance 2. However, $r1$ can only tolerate the dissatisfying experience with distance 1 and announces a collision at the vertex d.

When using SC-CBS, we will record the collision that occurred to $r1$ as ($r1$, d, 3), indicating that the agent $r1$ will collide at the vertex d at the third step. Then, SC-CSB will avoid any paths leading $r1$ to d at step 3 (including $a \to b \to x \to d \to e$ and $a \to b \to c \to d \to e$) and will end up with a longer detour through the vertex y. The SC-CBS solution has a cost of 5 for $r1$ and 9 in total.

When using SC-M*, the collision at d triggers the sub-dimensional expansion of the search graph in dimension 1, which includes both x and y. Thus, it can find a cheaper collision-free path through x and end up with a soft-collision-free path $a \to b \to x \to d \to e$ with a dissatisfying experience of distance 1 and a cost of 4.5 for $r1$ (8.5 in total). However, SC-M* does not expand dimension 2 because no collision has been announced by $r2$.

m	SC-CBS	SC-M*	SC-A*
4	1.971	2.002	47.35
5	1.798	3.312	473.7
6	1.942	2.969	390.0

Table 4.4: Average run time of SC-based MAPP solvers under the one-resource-one-type context.

When using SC-A*, the joint search space of both dimension 1 and dimension 2 is expanded and searched. Instead of the vertexes x and y, SC-A* will first investigate the vertex z in dimension 2 according to some heuristics. This process leads to another cheaper path $f \to b \to z \to d \to g$ with distance 4 for $r2$ (8 in total, which is the same as the individual optimal cost) and avoids all interference by moving through this path. As a result, SC-A* returns an optimal soft-collision-free solution at the expense of search space.

The example in Figure 4.9 illustrates the optimality of SC-A* and the advantage of SC-M* in path cost over SC-CBS. To be specific, SC-M* provides a better solution than SC-CBS by searching thoroughly through the expanded dimensions, whereas the way SC-CBS identifies collisions is inappropriate under the soft-collision context.

Run Time

Table 4.4 shows the average run time of the three SC-based MAPP solvers and we observe that both SC-M* and SC-CBS are significantly faster than SC-A* in terms of run time. This is reasonable because SC-A* always searches the global high-dimensional joint space, which is expensive. SC-CBS is faster than SC-M* because it always searches in one individual dimension at a time, whereas the SC-M* needs to occasionally deal with high-dimensional space when collisions occur.

Scalability

We compare the scalability of the three SC-based MAPP solvers in terms of planning for a large system size ($m > 50$). Figure 4.10 presents the success rate, average additional cost (i.e., how much more cost than the individual optimal path), and run-time ratio over SC-CBS under different thresholds T, where the run-time ratio of SC-CBS is compared to itself and thus is constant. SC-A* has the slackest constraint ($T = 0.35, \delta = 9.0$) but poorest performance because of the prohibitively large search space. SC-CBS has the best success rate because of the property of the decoupled searching. However, this is at the expense of path cost. SC-M* performs

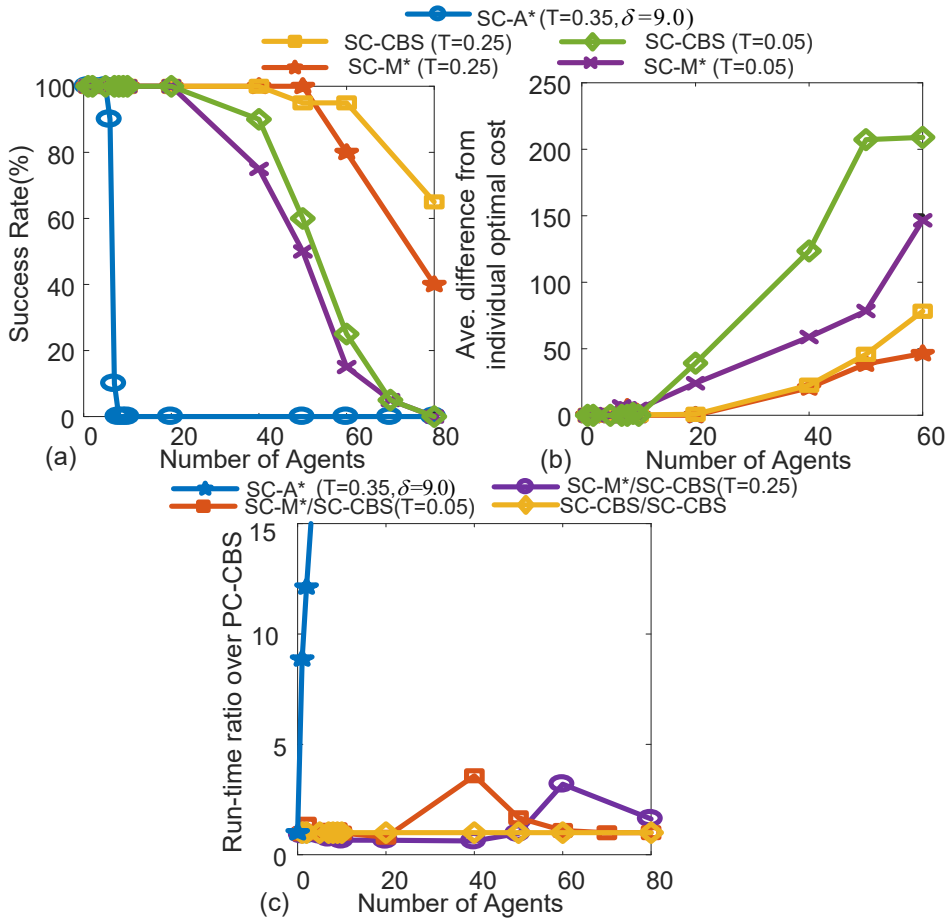

Figure 4.10: Success rate, cost, and run time ratio of the three SC-based MAPP solvers under different T.

decently in terms of both the success rate (significantly superior to SC-A*) and cost (noticeably lower than SC-CBS) as the number of agents increases.

The run time of the SC-M* is generally longer than that of SC-CBS. In Figure 4.10.c, we observe that the run-time ratio of SC-M* over that of the SC-CBS starts to decrease after a peak. This is because we force all algorithms to terminate after 1000 seconds, and both curves will converge to value one when their success rates decline to zero. We conduct another scalability experiment with different offsets δ (given $T = 0.25$) and observe the same results in terms of scalability. Figure 4.11 shows the experimental results.

In addition, we repeat all the experiments described above on another transit system featuring bottleneck areas. In terms of the success rate, run time, path cost and scalability among the three SC-based algorithms, the same phenomena are observed. Appendix A provides the structure of the transit system and main experimental results.

Considering the scalability and path cost altogether, SC-M* demonstrates its overall advan-

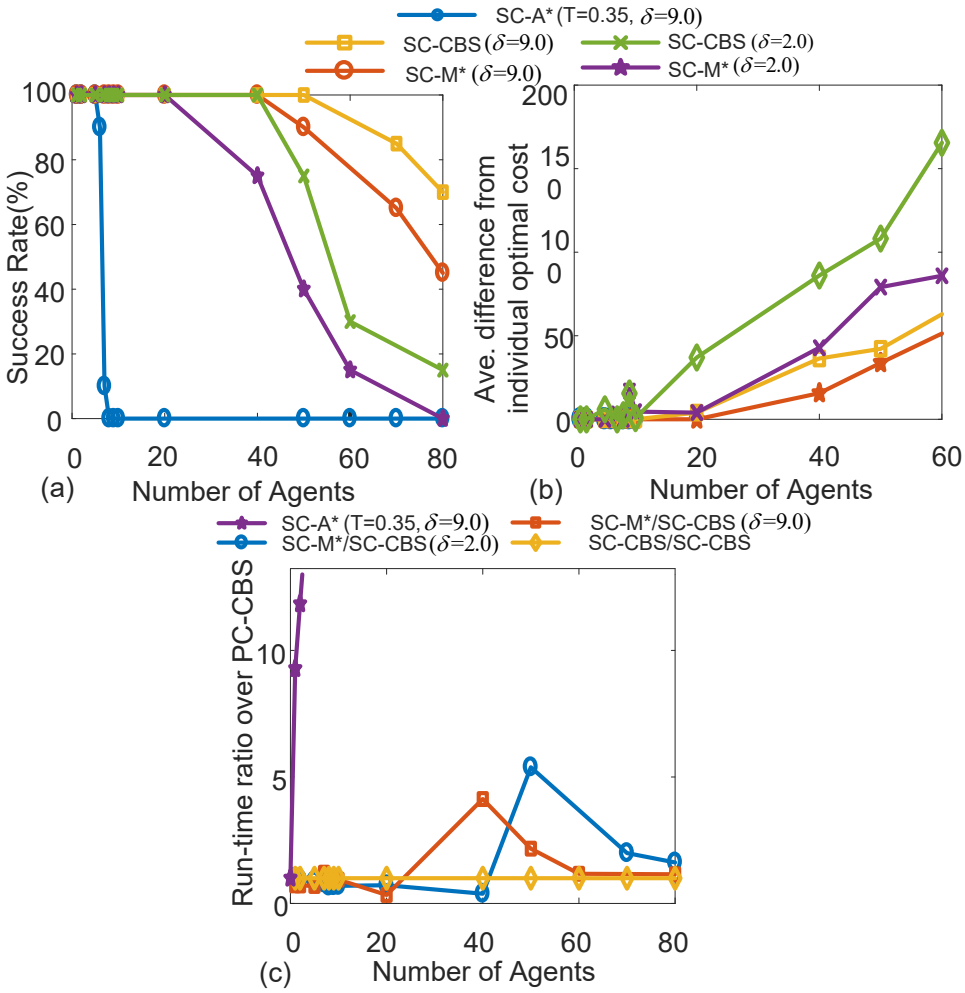

Figure 4.11: Success rate, cost, and run-time ratio of the three SC-based MAPP solvers under different δ.

tages over alternative SC-based solvers.

4.6 Summary

This chapter describes SC-M*, a generalized version of M*, which can handle MAPP problems under the soft-collision context. SC-M* is complete and suboptimal, constrained on the soft-collision-free condition. Experimental results demonstrate that SC-M* has improved scalability to plan for more agents and can be extended to handle more complex systems involving multiple resources and multiple types of agents. It is possible to further improve the performance by integrating advanced M* variants or other complete and optimal MAPP solvers, such as EPErM*, ODrM*, etc.

Chapter 5

SC-M* for Real-World Bus Transit System

This chapter applies the SC-M* algorithm presented in Chapter 4 to a real-world public transit system. The SC-M* should customize the planning for different types of client passengers and handle their mobility requests on a city scale.

In Chapter 2, we developed a data-driven on-vehicle experience simulator P-SUMO for the bus transit system of Porto, Portugal that simulates the passenger behaviors and the dynamics of certain on-vehicle resources (i.e., Wi-Fi quality and space availability). We apply the SC-M* to the Porto bus transit system to study the algorithm's performance. We generate the mobility plans based on the *expectation graph* (will be defined later), which is constructed using the logs of simulations driven by the historical data, and test the plans' execution based on the *testing simulations* driven by the future unseen data.

To mitigate the scalability issue in real systems, we tune the SC-M*'s build-in parameters. Similar to Chapter 4, we design experiments to evaluate the proposed method in terms of path cost, soft-collision-free constraint, and scalability in terms of run time and success rate. The experimental results show the advanced scalability and flexibility of the SC-M* for handling complex environments with multiple types of passengers requesting multiple types of service resources. Paper [57] covers most of the work in this chapter.

5.1 Multi-Passenger Multi-Criteria Mobility Planning in Real Public Transit

Most traditional research into planning for passenger-level mobility in the context of public transit networks aims to find the shortest path for independent individuals, also known as *route planning* [58]. When concerning passengers' experience, shortest path algorithms are not enough because they only consider distance or time as the optimization criterion. For public transit net-

works, however, other criteria are just as important, leading to a multi-criteria planning problem. A popular planning method is *scalarization*, which uses a linear combination of criteria as the optimization objective. This approach transforms the multi-criteria search into a single-criterion search using the criteria's weighted sum, which is practical and computationally feasible.

In addition to multi-criteria mobility planning, multi-passenger mobility planning is necessary for intelligent public transportation systems, in which passengers have various preferences, even necessities, in terms of *public resources*, such as seat availability (necessary for seniors) or on-vehicle Wi-Fi supply (preferred by video viewers and game players during the trip). When concerning multiple passengers' mobility on a city scale, it is not practical to plan for each passenger independently because passengers may interfere with one another regarding public resources. Individual optimal paths can cause serious interference, leading to low-quality experiences. Interference between passengers is soft because it is possible that they do not call for the same resource when they are on the same public vehicle. Also, they are able to tolerate each other over a short time and distance. Intuitively, how likely a collision (intolerable interference) actually happens depends on 1) whether the resource supply is less than the demands and 2) how long the lack-of-supply condition lasts in terms of the time and distance that the passengers stay together.

Client passengers can be viewed as agents moving through the transportation network. When we plan for all the agents, eliminating any hard collision is neither necessary nor feasible. Thus, we are more interested in another problem: how to maximize the public resources received by all client passengers such that the probability of collision of each client passenger is less than a bound?

We formalize the problem as a MAPP problem under the soft-collision context, and we optimize the mobility of each client passenger over multiple public resources. To solve the practical problem efficiently, for the first time, we generalize the SC-M* to plan for and accommodate multiple client passengers simultaneously. Our work contributes to showing that a MAPP solver framework can be extended to the public transportation domain and accommodate the passengers' mobility demands on a city scale. The SC-M* is expected to help with the congestion and resource management issues in the context of a surge in city's population and mobility needs.

The work in this chapter is based on the building blocks in previous chapters. Specifically, Chapter 2 and Chapter 3 enable the on-vehicle experience simulation which provides synthetic passenger data and bus transit data that are necessary to build a model of the urban public transit network. On top of the model, the proposed SC-M* in Chapter 4 can apply.

5.2 Briefings of Relevant Building Blocks in Previous Chapters

In previous chapters, we developed three building blocks needed for the development and demonstration of the multi-passenger multi-criteria mobility planner for the Porto bus transit system.

5.2.1 SC-M*

The development and fundamental analysis of the proposed SC-M* were conducted on a small grid environment in Chapter 4, and the obtained theoretical results show that SC-M* is:

- Complete – given a finite graph, it guarantees either returning a path in finite time or determining that no planning solution exists;
- Suboptimal – it may not return the optimal path because the soft collision set defined in Eq. 4.9 and in Eq. 4.10 ignores the soft conflicting agents in the upstream path in return for improving the scalability;
- Superior in path cost to other SC-based solvers, such as SC-CBS (i.e., to combine the soft collision free constraint with the state-of-the-art Conflict-Based-Search algorithm [46]). This low path cost results from that the SC-M* explores the expanded dimensions thoroughly whereas the SC-CBS conducts decoupled searching for each individual agent.

SC-M* demonstrates its significant overall advantages in terms of cost optimality (i.e., path cost) and scalability (i.e., run time and success rate) over other alternative MAPP solvers under the soft-collision context, making it a promising candidate for solving the city-scale passenger mobility planning.

5.2.2 Retrieving the Missing Destination in Real Data

Chapter 3 inferred the missing destination information from the entry-only AFC passenger data by applying the semi-supervised self-training approach. Specifically, based on the self-training using personal historical information (STP), we retrieved >90% of the missing O-D demands with high inference confidence. The post-processed O-D demands will serve as the input configuration of the non-client passenger flow in the on-vehicle experience simulation.

5.2.3 On-Vehicle Experience Simulator

Chapter 2 presented the joint traffic-passenger simulator P-SUMO combined with the on-vehicle Wi-Fi service model. Important points about the joint passenger behavior modeling and simulation in P-SUMO are as follows:

- The Porto bus transit system is established in SUMO and the on-vehicle Wi-Fi service map based on real urban hotspot location data is constructed;
- The interactions between buses and passengers take place at the bus stop, and the number of boarding and alighting passengers affects the bus dwell time;
- The interactions between buses and other road traffic are simulated by SUMO, in which a bus's travel time on the road varies according to the traffic conditions and the bus dwell time at stops;
- The simulation log records detailed passenger experiences (waiting/boarding/alighting time at a stop, Wi-Fi quality on buses, travel distance, etc.) and bus state information (bus arrival time at each stop, on/off passengers' ID, bus stop dwell time, Wi-Fi quality samples on each route, passenger volume, etc.).

With the joint traffic-passenger modeling, the simulator P-SUMO captures primary interactions among passengers, buses, and traffic. The simulation log will be used to construct the *expectation graph*. As will be discussed later in this section, the expectation graph is a model of the public transit network, which is a prerequisite for the planning. Based on model, the SC-M* application is applied. We run the simulation for all four Wednesdays in May of 2010, using the simulation logs of the first three Wednesdays (May 5, 12, and 19) as the historical data for constructing the expectation graph, based on which we apply the SC-M* to generate mobility plans. The generated plans are tested by executing them in the *testing simulation* of the unseen Wednesday on May 26, 2010.

5.3 Additional Stepping Stones Prior to Applying SC-M*

Before applying the SC-M* to generate mobility plans, the algorithm needs to 1) pre-process the simulation-log data to build the *expectation graph* of the bus transit network and 2) customize certain technical details of the SC-M* to the bus transit system because the real systems differ

from the traditional MAPP environments. To this end, this section introduces the *time-expanded expectation graph* model and the *forward-the-tail-agent* approach.

5.3.1 Expectation Graph of Bus Transit Network

One key point to planning for improved experience with soft-collision-free constraints is to construct a model of the urban public transit network with necessary information. In addition to the static infrastructure information about the bus routes and stop locations, dynamic information obtained from the simulation (e.g., the timing of the bus trips, Wi-Fi quality, and passenger volume) is needed. We use a *time-expanded model* to model the static and dynamic information about the public transit system in a weekday, and the average of the time-expanded models of several weekdays produces the *expectation graph*, upon which we apply the SC-M* during the planning phase.

Time-Expanded Model (A Model for One Weekday)

The public transit network is based on a *timetable*, consisting of a set of stops, routes, and trips. A trip of a route corresponds to a vehicle visiting a list of stops along the route at specific times of day [58]. The main approach for constructing a graph of the public transit network from the timetable is *time-expanded* modeling [59]. The time-expanded model creates a vertex for every event (e.g., a bus arriving at a stop) in the timetable and uses edges to connect subsequent events in the direction of time flow. In our research, we replace the timetable events with the bus-arrival events in the simulation log to build the time-expanded model.

As illustrated in Figure 5.1, we consider the bus transit network as a *directed graph* G (note that the graph G here is the same as G^j in Section 4.3.2, for simplification). Each vertex corresponds to an event in the simulation log. An event is defined as a bus arriving at a stop during a bus trip of a certain route. In Figure 5.1.a, for example, the blue route (with blue *solid* arrows) has three bus trips, and v_{2-1} corresponds to a bus arriving at the bus stop V_1 during the blue bus trip 2 at the time t_{2-1}. V_1 is defined as the set of the arrival events occurring at the same stop (i.e., Stop 1). The arrow edge connects two subsequent bus-arrival events along a bus trip. The color of the edge indicates the route the bus trip belongs to. For example, the blue edge from v_{1-1} to v_{1-2} indicates that, during the blue bus trip 1, the arrival event at the stop V_1 at time t_{1-1} precedes the arrival event at the successor stop V_2 at time t_{1-2}. In the graph, passengers can transfer to another bus trip of the same route. The transfer connections are represented by

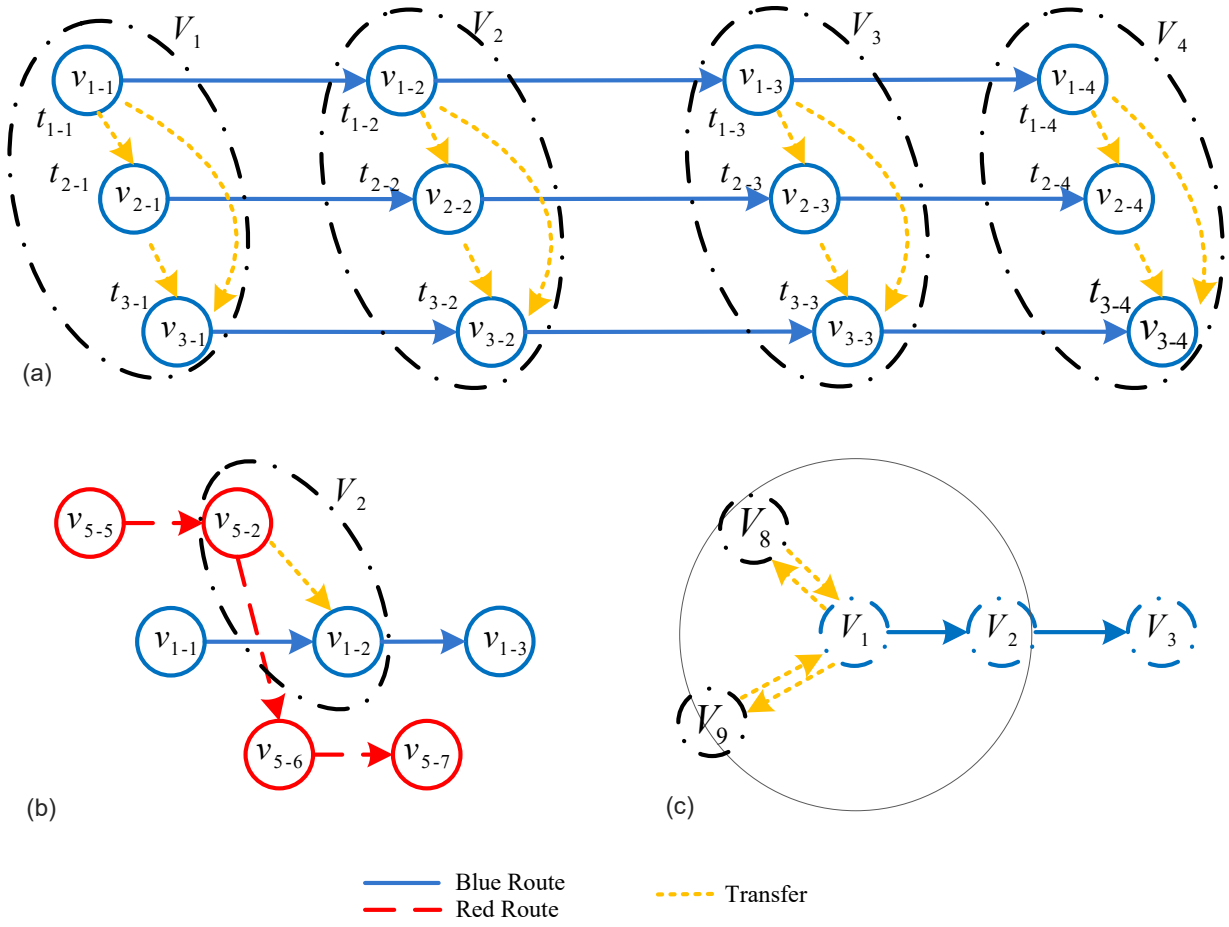

Figure 5.1: Time-expanded graph for the bus transit network of one weekday. (a) shows the time expanded graph for the blue route and (b) shows how passengers can transfer between two different routes. (c) shows the transfer links between walking-reachable stops. All dashed vertexes/ovals indicate the set of events occurring at certain stop V.

the yellow *dotted* arrow edges. The events at the same stop are fully connected, and any pair of subsequent events has a yellow dotted connection.

For the time-expanded graph, Figure 5.1.b illustrates that the red route (with red *dashed* arrows) and the blue route (with blue *solid* arrows) share the same stop, V_2, and that passengers can transfer from the red route to the blue route at this stop ($t_{5-2} < t_{1-2}$ is a prerequisite for creating such a transfer edge). In addition to transfer edges at the same bus stop, walking transfer edges are introduced into the graph, as illustrated in Figure 5.1.c. Stops within a short geographical distance (e.g., 640 meters) are considered as reachable via walking, and passengers would be willing to walk a few more meters to transfer routes there. The transfer edge between bus stops (black dashed circles/ovals) in Figure 5.1.c indicates the full connection between any pair of subsequent events occurring at the two stops.

Expectation Graph Construction (Average Model of Several Weekdays)

After obtaining the simulation log of a weekday from the P-SUMO, we extract the simulated travel time, average Wi-Fi quality, and passenger volume as each edge's attributes. The yellow dotted transfer edges are outside of the vehicles; thus, their Wi-Fi and passenger volume attributes are set to zero. We construct the *time-expanded models* of three simulated Wednesdays in May (i.e., May 5, 12, and 19). By averaging the attributes over the three models, we obtain the *expectation graph* of the bus transit network. SC-M* uses the graph to plan for the client passengers in the planning phase. Using this graph, passengers who reach one of the events at the destination stop fulfill their individual travel plan. One disadvantage of the time-expanded model is that the resulting graph is quite large. To handle this issue, the expectation graph only contains bus trips of which departure times are within a 30-minute time window. Specifically, our experiments use a time window between 14:00 and 14:30.

5.3.2 Forward-the-Tail-Agent Approach

In contrast to the MAPP settings, in which the agents are assumed to reach certain vertexes at a time step (i.e., the time spent on each edge is the same), the agents in real-world networks do not synchronize because of the various travel times spent on different edges. As shown in Figure 5.2.a three agents in the traditional MAPP context are assumed to synchronize at the joint vertexes $(v_1^1,\ v_4^2,\ v_7^3)$, $(v_2^1,\ v_5^2,\ v_2^3)$, and $(v_3^1,\ v_6^2,\ v_3^3)$ at the times t_1, t_2, and t_3, respectively. Therefore, the planner will know that the Agent 1 and Agent 3 meet up at v_2 and interfere on the edge from v_2 to v_3. However, when traditional MAPP planners are applied directly in real-world networks, as shown in Figure 5.2.b, agents will move from the joint vertex $(v_1^1,\ v_3^2,\ v_5^3)$ to $(v_2^1,\ v_4^2,\ v_1^3)$ after the first step, and the interference on the edge from v_2 to v_3 cannot be detected in real time. Thus, without this synchronization property, SC-M* and other MAPP algorithms cannot be directly applied. Note that identical vertexes are shaded in colors and we have $t_1 < t_2 < t_3 < t_4 < t_5$ in Figure 5.2.

To deal with this practical issue, we introduce the *forward-the-tail-agent* approach. At each time step, this approach only forwards the agent at the tail (i.e., the last agent in the time flow). If multiple agents synchronize at the tail, they will be forwarded simultaneously. In Figure 5.2.b, three agents start with the joint vertex $(v_1^1,\ v_3^2,\ \underline{v_5^3})$ and the approach only forwards Agent 3 (Agent 3 is at the tail at that time, which is underscored) from v_5^3 to v_1^3 at the first step, and the joint vertex $(v_1^1,\ \underline{v_3^2},\ v_1^3)$ is reached. At step 2, Agent 2 is at the tail and is forwarded from v_3^2 to v_4^2.

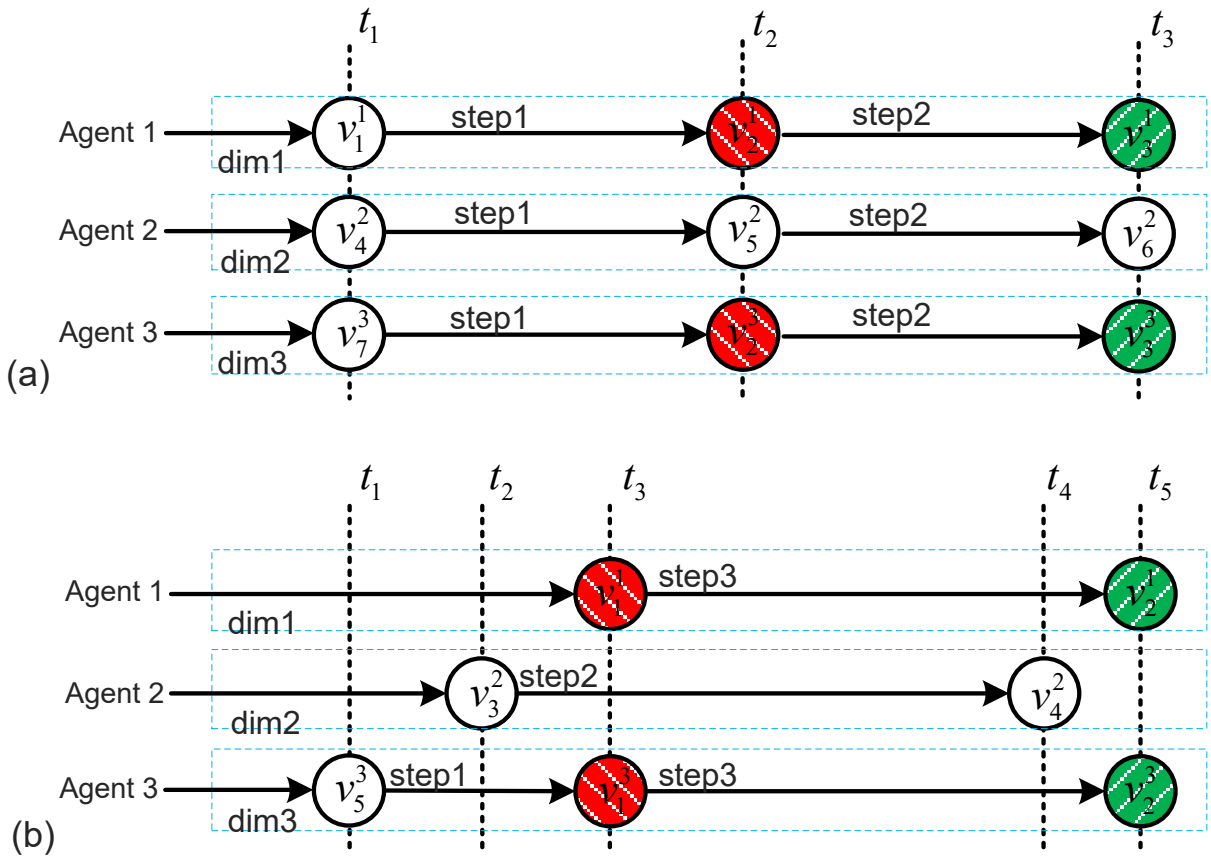

Figure 5.2: Forward-the-tail-agent approach to handle non-synchronization in real systems.

Then, the joint vertex $(\underline{v_1^1}, v_4^2, \underline{v_1^3})$ is reached, in which both Agent 1 and Agent 3 are at the tail. At step 3, both Agent 1 and Agent 3 are moved to v_2^1 and v_2^3, respectively, and the interference on the edge from v_2 to v_3 is detected at this step. Compared to the three joint vertexes in the traditional MAPP setting in Figure 5.2.a, Figure 5.2.b shows a sequence of four joint vertexes by using the proposed approach: $(v_1^1, v_3^2, \underline{v_5^3})$, $(v_1^1, \underline{v_3^2}, v_1^3)$, $(\underline{v_1^1}, v_4^2, \underline{v_1^3})$, and $(v_2^1, \underline{v_4^2}, v_2^3)$. The forward-the-tail-agent approach can also be directly applied to the synchronization scenario in Figure 5.2.a, performing exactly the same as the traditional MAPP algorithms.

Using the expectation graph construction and the forward-the-tail-agent approach, the SC-M* planner can be generalized to the real Porto bus transit system to generate mobility plans for client passengers. The client passengers' mobility planning is based on the expectation graph in the planning phase and executed in the testing environment of a future unseen weekday (provided by the testing simulation) in the testing phase.

5.4 Experiments and Evaluations

This section evaluates the proposed SC-M* on the on-vehicle experience simulator P-SUMO. The evaluation is twofold.

- First, we start with the planning tasks for a small number of client passengers under the strict hard-collision-free constraint. On top of this planning, we can conduct a *zoom-in checking* on the quality of each generated mobility plan.
- Second, we then apply the SC-M* to a large number of passengers under the soft-collision context, in which we can conduct a *property checking* on the scalability (i.e., the run time and success rate under various client passenger sizes) and on the satisfaction of the soft-collision-free constraint of each generated mobility plan. The planning tasks are O-D pairs of client passengers randomly generated over the city.

We run the SC-M* with an Intel Core i7-6700 CPU at 3.4 GHz with 16 GB RAM. We consider two public resources: Wi-Fi quality and available space on public vehicles. The goal of SC-M* is to find a joint path in the public transit system for multiple client passengers, optimizing the overall multi-criteria objective constrained on the soft-collision-free condition. The cost objective, which is customized according to passengers' service preference, is defined in terms of three criteria: travel time, Wi-Fi cost, and space cost. The Wi-Fi cost of an edge is given by $max(0,\ 150 - ave\_WiFi\_data\_rate)$, which is a rectified linear function of the negative average Wi-Fi quality in Mbps (see Figure 2.15). The cost regarding the lack of space is quantified using the passenger volume on the edge. Note that the travel time, average Wi-Fi quality, and passenger volume are the three attributes of an edge given by the expectation graph, as described in section 5.3.1. We apply the scalarization method to define the cost objective in SC-M* and optimize over the linear combination of the three criteria. Therefore, the overall cost of an edge used in the SC-M* is given by:

$$cost_{edge} = \alpha \cdot cost_{time} + \beta \cdot cost_{WiFi} + \gamma \cdot cost_{space}, \tag{5.1}$$

where α, β, and γ are the scale coefficients.

5.4.1 Evaluations with Respect to Path Cost and the Hard-Collision-Free Constraint

This section introduces the experiment setup and then presents and analyzes the experimental results.

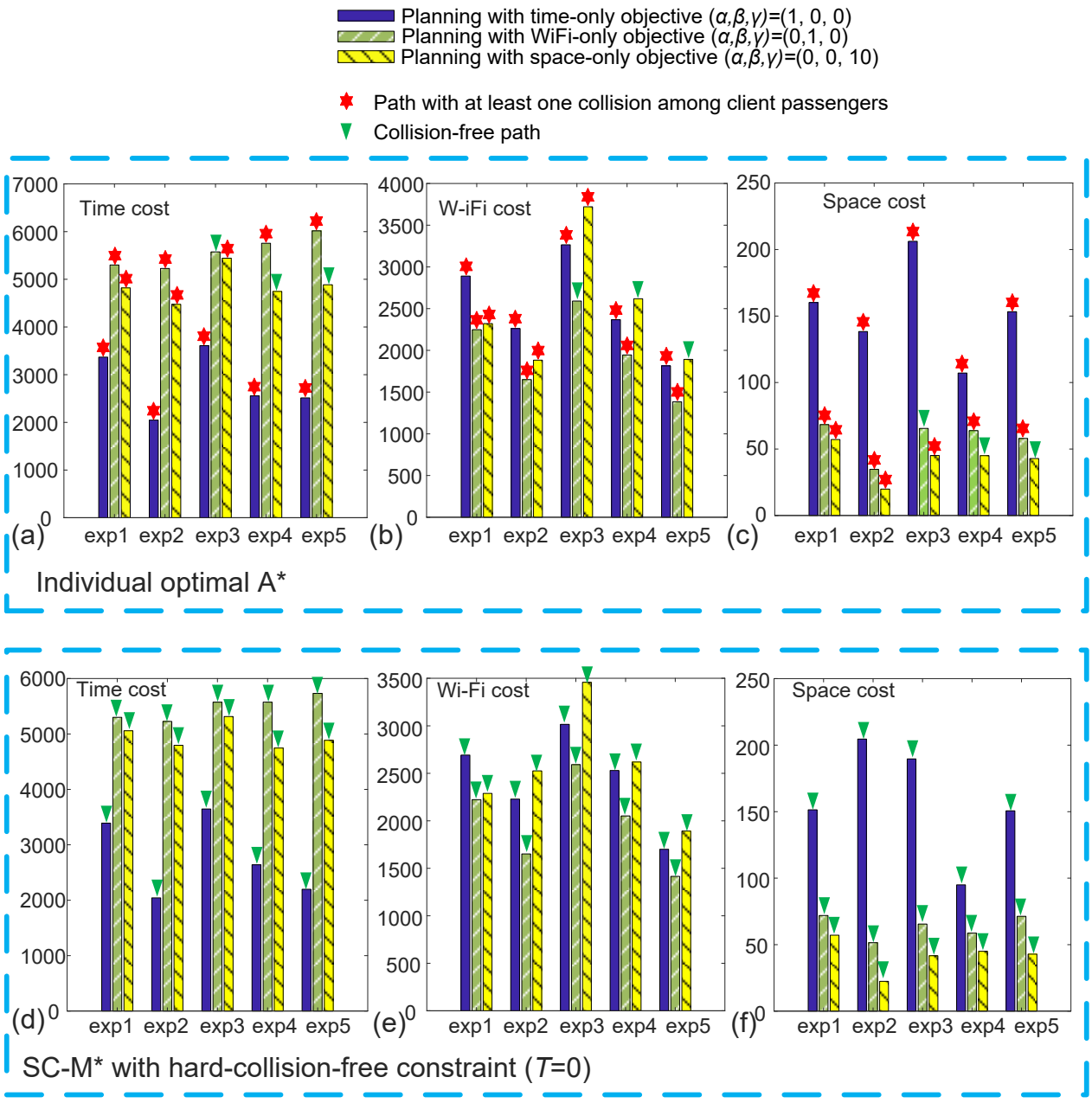

Figure 5.3: Comparison of the path costs between individual optimal A* and SC-M* ($T = 0$).

Experimental Setup

In this section, we apply the SC-M* to a small client passenger size and investigate each mobility plan's quality with respect to path cost and satisfaction of the hard-collision-free constraint. Specifically, each experiment corresponds to a planning task for a group of five client passengers under the strict hard-collision-free constraint (i.e., the collision threshold T in Eq. 4.9 is set to zero). To start with a small number of client passengers and the hard-collision context is

necessary because we can easily evaluate the results and verify the basic function of the SC-M* planner. The planning task of each experiment is designed to encounter at least one collision along the individual optimal path. Thus, the SC-M* paths must differ from the individual optimal paths. We use the standard A* as the individual optimal planner and evaluate each plan's quality via the *testing simulation* of the final Wednesday in May, which is unseen during the planning phase. Because the expectation graph is based on the other three Wednesdays in May, it is expected that the environment in the testing simulation is similar to the expectation graph in terms of the non-client passenger flow and road traffic, and thus, the planner should achieve a reasonably low path cost with the collision-free constraint satisfied for each planning task.

We design five planning tasks (one task per experiment). In each experiment, we

- Apply the two MAPP planners (individual optimal A* and SC-M* [$T = 0$]) separately to the task;
- Each planner generates three independent plans for each task using the time-only (α, β, γ = 1, 0, 0), WiFi-only (α, β, γ = 0, 1, 0) and space-only (α, β, γ = 0, 0, 10) objectives.

Experimental Results

We execute the planned paths in the testing simulations and obtain the path cost and collision information of the planned paths, as presented in Figure 5.3. From the figure, we observe that:

- The time cost (i.e., the travel time of the planned path) is minimized by both planners under the time-only objective (see Figure 5.3.a and Figure 5.3.d). In the five experiments, the same phenomenon consistently occurs to the Wi-Fi cost under the WiFi-only objective (Figure 5.3.b and Figure 5.3.e) and to the space cost under the space-only objective (Figure 5.3.c and Figure 5.3.f). These results verify the basic function of finding the shortest path for the customized cost objectives;
- Most individual optimal A* paths encounter collisions (marked with red asteroid indicators), whereas the paths from the SC-M* are all collision-free (marked with green triangle indicators).

To visualize whether the hard-collision-free constraints are satisfied by the SC-M*, Figure 5.4 shows an example of executing the planned paths from the individual optimal A* and from the

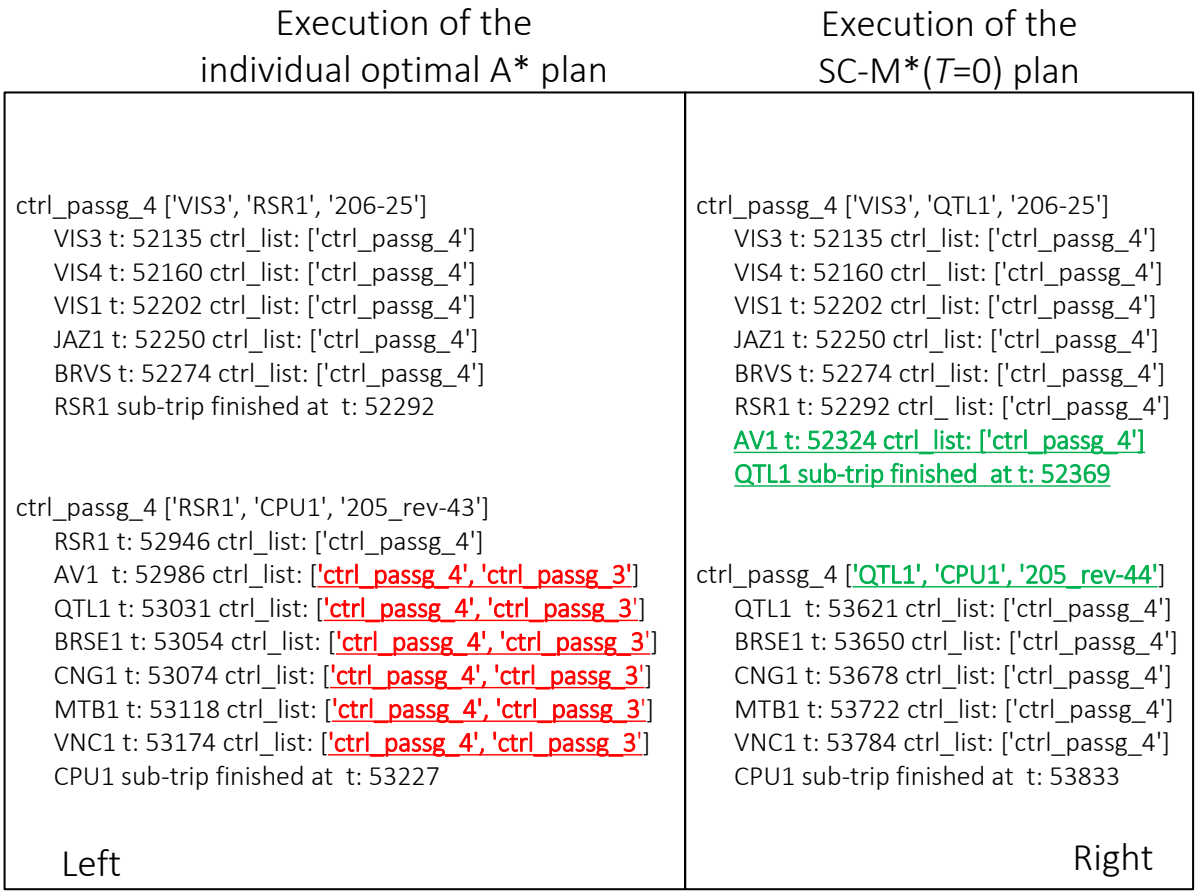

Figure 5.4: The simulation log of the execution of plans generated by individual optimal A* and SC-M* ($T = 0$). Note the SC-M* ($T = 0$) is equivalent to the traditional hard-collision-free M*.

SC-M* in the testing simulation. In this example, the client passenger "ctrl_passsg_4" attempts to move from stop "VIS3" to stop "CPU1". By following the plan from the individual optimal A*, "ctrl_passsg_4" collides with client passenger "ctrl_passsg_3" during the bus trip "205_rev-43" from stop "AV1" to stop "VNC1" (underscored in red in the left box), which is not allowed under the hard-collision context. In contrast, the SC-M* plan can avoid the collision by having "ctrl_passsg_4" move two more stops along the first bus trip, "206-25", to take a different bus trip, "205_rev-44", to reach the destination stop of "CPU1" (underscored in green in the right box). The arrival time of the SC-M* plan is 53833, which is about 10 minutes (i.e., 600 seconds) later than the individual optimal A* plan's arrival time (i.e., 53227).

The experimental results demonstrate that the SC-M* can optimize the path cost while preventing each client passenger from collisions, and thus, improving the client passengers' travel experience conditioned on the collision-free constraint.

5.4.2 Evaluations with Respect to Scalability

This thesis attempts to mitigate the scalability problem in real world: The global search space grows exponentially with the number of agents and will become prohibitively large when planning on a city scale; The involvement of multiple preference types and resources produces more collisions, leading to a quick increase in dimensional expansion.

Many factors can help with scalability: We can 1) choose a reasonable time window (e.g., 30 seconds) to limit the number of client passengers that SC-M* must deal with at a time; or 2) reduce the chance of collisions by tuning the build-in parameters of SC-M*.

In this section, we will show that the scalability of SC-M* in real systems is controllable via the tunability of SC-M*. In the experiments, each client passenger attempts to move from the origin to the destination with a low path cost and with bounded collision scores (i.e., the probability of collision [defined in Eq. 4.8] is bounded).

Two-Resource-Two-Passenger-Type Setting

In this experiment, we increase the planning difficulty by considering two types of client passenger requesting two public resources: Wi-Fi and space (i.e., $A = \{A_1 : "WiFi", A_2 : "Space"\}$). The Wi-Fi capacity is the data rate available on the edge and the space capacity is the number of remaining seats. We set the satisfying values to $\varepsilon_1 = 60$ and $\varepsilon_2 = 5$ for Wi-Fi and space, respectively. Type I passengers apply f_1 as the collision CDF for the Wi-Fi resource and the linear CDF f_2 for the space resource, implying that they view WiFi and space as important and trivial, respectively. On the other hand, type II passengers used f_2 for the Wi-Fi resource and f_1 for the space resource. Each client passenger has a 50% chance of being type I. Both CDFs are adjusted using the offset parameter δ, and the collision score is constrained by the collision threshold parameter T. The parameters and symbols used here are defined in Chapter 4.

Each trial randomly generates O-D pairs for client passengers, and we give each trial 1500 seconds to find a solution. For each configuration (including the number of client passengers, the collision threshold T, and the offset parameter δ), we run 20 trials to calculate the average metrics (i.e., the success rate and run time).

Success Rate and Run Time Results

We study the influence of the collision threshold T and offset parameter δ on the SC-M* performance. Figure 5.5.a shows the success rate of the SC-M* in the planning phase under the

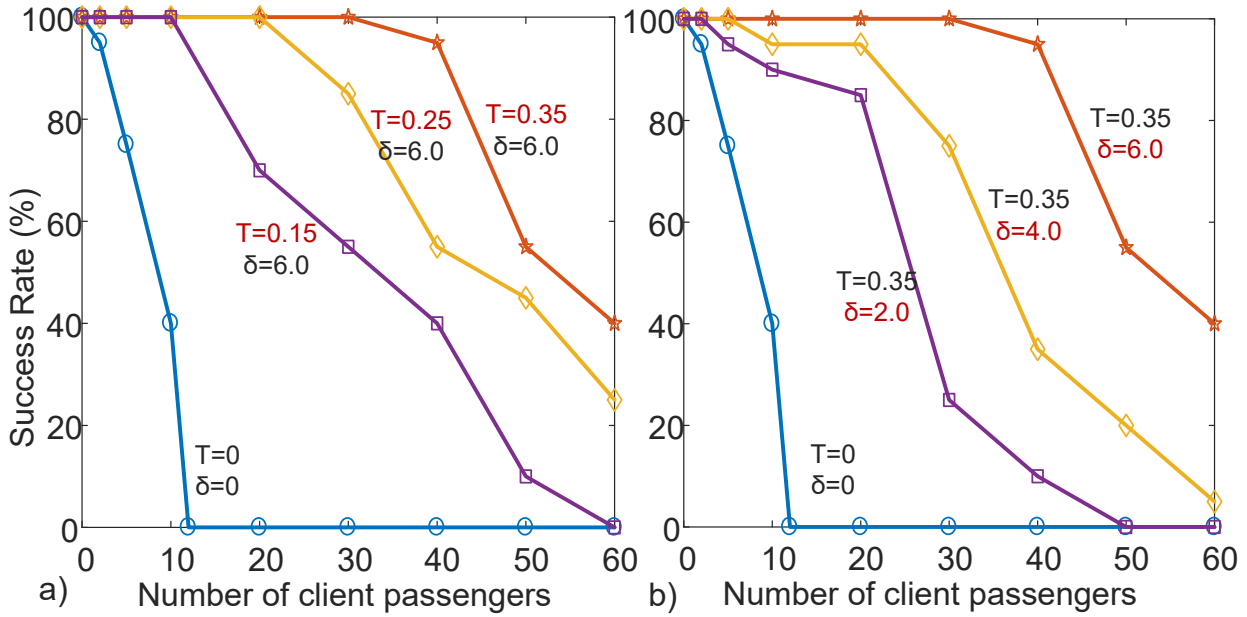

Figure 5.5: a) The impact of the collision threshold T on the success rate of SC-M*, given $\delta = 6.0$; b) The impact of the offset parameter δ on the success rate, given $T = 0.35$.

m	T =0	T =0.15	T =0.25	T =0.35	δ =2.0	δ =4.0	δ =6.0
2	97.55	30.45	32.94	30.93	33.88	30.14	30.93
5	381.26	79.43	83.52	80.70	151.73	77.84	80.70
10	757.95	177.58	182.21	160.93	300.97	222.63	160.93
20	>1500	733.19	340.17	436.08	508.90	369.16	436.08
30	>1500	1023.95	648.30	602.10	1266.13	769.68	602.10
40	>1500	1254.28	1023.05	682.56	1408.25	1255.60	682.56
50	>1500	1460.95	1160.49	1134.24	>1500	1364.94	1134.24
60	>1500	>1500	1336.63	1357.88	>1500	1497.44	1357.88
	Left: $\delta = 6.0$.				Right: $T = 0.35$		

Table 5.1: The impact of T and δ on run time (in seconds).

collision thresholds of $T = 0$ (equivalent to the basic hard-collision-free SC-M*), 0.15, 0.25, and 0.35 while the offset parameter is fixed to $\delta = 6.0$. Table 5.1.Left shows the corresponding run time (in seconds) in the planning phase under various collision thresholds. Figure 5.5.b shows the success rate under the offset parameters of δ = 2.0, 4.0, and 6.0 while the collision threshold is fixed to $T = 0.35$. Table 5.1.Right shows the corresponding run time.

The results clearly show that a large collision threshold T and offset parameter δ improve the scalability (i.e., improving the success rate of finding a solution and reducing the run time under large client passenger sizes [$m > 40$] in the planning phase). The planning under the hard-collision-free constraint, which is equivalent to applying the basic M* to the public transit

network, does not scale well as it can only handle the planning tasks for fewer than 12 client passengers.

The experimental results show that for the SC-M*, larger thresholds and offset parameters render more relaxed constraints; thus, client passengers are expected to tolerate more interference on the resources before announcing a collision. With this property, one can tune the parameters to trade off the scalability against the slackness of the constraints, being quite flexible when planning on a city scale. This experiment also demonstrates that the SC-M* can feasibly handle a complex environment with multiple resources and multiple types of client passengers.

Satisfaction of the Constraints on Collision Scores

The scalability should also consider satisfying the soft-collision-free constraints in the testing phase. We execute all the planned paths in the testing simulation (i.e., the simulation of the unseen Wednesday on May 26, 2010) and extract the planned and the simulated collision scores of each client passenger to check the satisfaction of the soft-collision-free constraints. Figure 5.6 presents the maximal collision score observed for each configuration when executing the planned paths in the planning phase and in the testing phase. Note that each point in the figure corresponds to the maximal collision score in the 20 trials of a certain configuration. If the maximal collision score is less than the collision threshold T, the planning of the SC-M* under the corresponding configuration is successful for all 20 trials. Therefore, each client passenger has a travel experience with a below-the-threshold collision score.

From Figure 5.6, we observe that all the planned collision scores (blue curves with asteroids) are below the collision thresholds (black horizontal lines), indicating that the SC-M* plans strictly satisfy the soft-collision-free constraints in the environment described by the expectation graph. For the simulated collision scores (red curves with circles), we observe that, despite outlier points slightly exceeding the collision thresholds, the executions of the planned paths in the testing simulation perform reasonably well and confine the collision score of each client passenger's on-vehicle experience below the bound, even if the client passenger size is large.

The experimental results support that the SC-M* can scale to assure the quality of the plans, satisfying the soft-collision-free constraints in the expectation graph and extending the soft-collision-free property to the real-world environment, as long as the expectation graph reasonably reflects the real public transit environment.

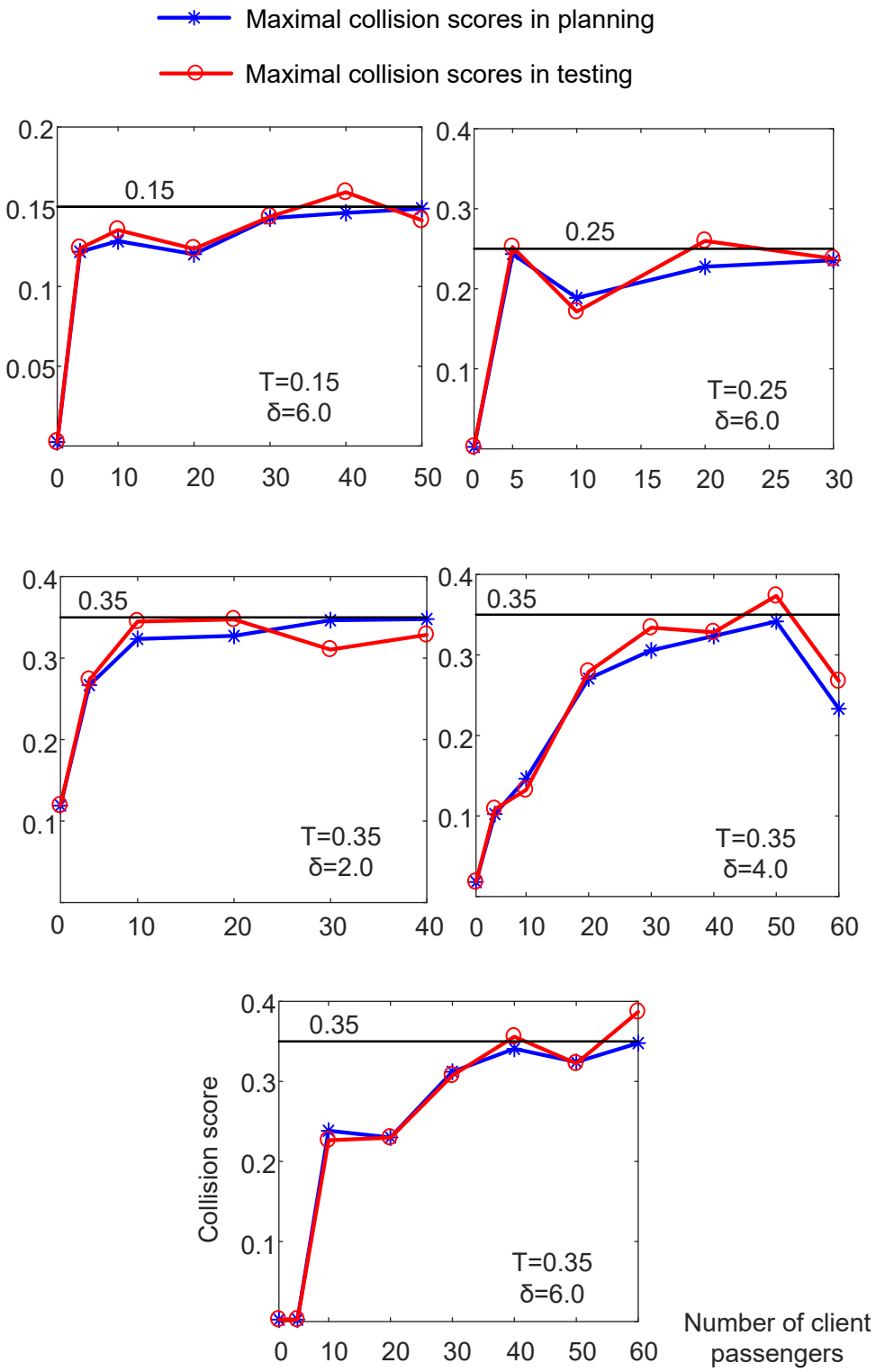

Figure 5.6: The maximal collision score observed in the planned paths in the planning phase and in the testing phase. To satisfy the constraint, the maximal observed collision score is expected to be less than the collision threshold T.

5.5 Summary

This chapter applies the SC-M*, a scalable multi-passenger multi-criteria mobility planner, to improve passengers' on-vehicle experience when travelling through the public transit system in a city. The proposed methodology can optimize client passengers' on-vehicle experience while limiting the probability of collisions to a bound. The parameters control the slackness of the soft-collision-free constraints and brings advanced scalability to the SC-M*, making it a powerful tool in practice. The case study of the Porto bus transit system evaluates the SC-M* in terms of advantages in path cost and scalability in success rate and run time.

The SC-M* is based on the expectation graph, implying that failures may occur at some points in the real-world executions. To solve this problem, we need to develop a fast re-planner at the execution time from the location of failure to the destination given the current state of public networks, which is a promising direction in the future.

Chapter 6

Related Work

We seek to improve the travel experience of passengers in public transit through on-vehicle experience simulation and multi-agent multi-criteria mobility planning, which is of interest to the field of smart city. In this chapter, we review relevant literature that focuses on traffic simulation and, in particular, on passenger mobility planning.

6.1 Intelligent Public Transportation Systems

A city can be defined as "smart" when "investments in human and social capital and traditional (transport) and modern (Information and Communication Technologies [ICT]) communication infrastructure *fuel* sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance" [60]. A smart city is supposed to address the challenges referred above. To achieve this, an intelligent public transportation system (IPTS) is a prerequisite.

Previous solutions to an IPTS focus on developing innovative ICT and using information about the state of the transportation networks for scheduling and dispatching fleets [61]. To name a few, automatic vehicle location systems (AVLSs) monitor the real-time operation of a public transit network and process a very large amount of network information [62]. Traveler information systems (TISs) provide real-time information to passengers about the state and operating conditions of the road network to assist their pre-trip and *en route* decision making [63]. Automatic passenger counters (APCs) are systems that count on-board passengers and those waiting for vehicles at stops and stations [64]. Decision support systems (DSSs) assist decision makers in controlling the transportation network by establishing regular transportation timetables and suggesting strategies when unexpected events occur [65].

Although significant progress has been made to improve the IPTS, people inside of the vehicles, especially their *en route* experience, receive limited attentions. To develop an IPTS, passenger-centered research, that aims to improve passengers' travel experience inside of the public transit vehicles, is necessary. This has seldom been studied in the field of IPTS, which is the main contribution of this thesis.

6.2 Traffic and Passenger Simulation

Simulation has proven to be a useful prerequisites in building an IPTS. In traffic control research, instead of directly applying an approach to control the real traffic flow, testing it through simulation and transferring the knowledge to real road networks can save money and avoid safety problems [66, 67, 68]. For many IPST applications that involve interactions among complex spatial traffic networks, vehicles, and humans, building an effective model requires large-scale real-world data, which can be difficult, time-consuming, and even dangerous to collect. As a workaround approach, simulations are applied to generate synthetic data for city planners and transportation management practitioners to use in their work.

6.2.1 Simulation and Transfer Learning

Simulation-based research and practices are closely related to *transfer learning*, a learning category in the field of machine learning. Transfer learning involves the concepts of *domains* and *tasks* [69]: A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. Given a specific domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ serving as an object predictive function learned from the training data, where $Y \in \mathcal{Y}$. Then transfer learning is usually applied to the following scenario: Given a source domain $\mathcal{D}_s$ with corresponding task $\mathcal{T}_s$, and a target domain $\mathcal{D}_T$, with corresponding task $\mathcal{T}_T$, we attempt to improve the learning of the target predictive function $P(Y_T|X_T)$ in $\mathcal{D}_T$ using the knowledge gained from $\mathcal{D}_s$ and $\mathcal{T}_s$, where $P(X_s) \neq P(X_T)$ or $P(Y_s|X_s) \neq P(Y_T|X_T)$.

Problems related to transfer learning are very common in transportation practices as the training domain-task pair may not concur with the real-world domain-task pair. Simulation plays an important role in relaxing the transfer learning problems by converting X_s to X'_s, such that $P(X'_s) \approx P(X_T)$. In this sense, simulation is a key driving force of machine learning success in the field of transportation research.

6.2.2 Passenger Modeling and Road Traffic Simulation

The topic of traffic and passenger simulation can be divided into passenger modeling and road traffic simulation.

Passenger Modeling

Passenger behavior modeling has been involved in a lot of public transportation system research. To evaluate the performance of vehicle scheduling and platform deploying, the behaviors of passengers must be simulated and analyzed in detail. Although sophisticated enough to consider individual preferences [70], the seat allocation process [71, 72], and even personal activity such as pressure from passengers behind [73], most studies are highly *microscopic*, confining their application domains in environments with a limited amount of public vehicles and not scaling well to provide insights into *macroscopic* passenger flow on a city scale.

Road Traffic Simulation

Compared to passenger modeling, road traffic simulation (including public transit simulation) has gained rapid and significant developments. Many road traffic simulators, such as PTV Vissim [74], AIMSUN [75], Matsim [76], Simulation of Urban MObility (SUMO) [77], etc., are developed with delicate functions suitable to *macroscopic* city-scale simulation. Some of the simulators, such as SUMO, which provides a platform to explicitly simulate vehicles including cars, buses, and urban trains on a city scale, are open source. However, because most traffic simulators operate at the vehicle level, they are unable to provide passenger-level information about passenger-vehicle interactions, which is of great interest to the field of on-vehicle experience studies.

Our work in Chapter 2, which implements the simulator of bus passenger behavior in conjunction with the mature traffic simulator (SUMO), fills the gap between passenger and traffic simulation, capturing the macroscopic interactions between bus passengers and urban traffic environments through the city-scale traffic-passenger joint simulation.

6.3 Destination Estimation for Passenger Data

Research that estimates people mobility patterns requires access to real passenger data, which are important for a high-quality on-vehicle environment simulation. The availability of such data has improved over the past decades, and automated data collection (ADC) in public trans-

portation systems has become increasingly popular in cities worldwide. ADC infrastructures produce a large volume of data about passengers, providing insights into crowd size, passenger journey time, and spatiotemporal distribution of travel demands.

However, when it comes to passenger-centered research, the data available to researchers are usually insufficient, either because the data is incomplete with important features missing, or because the data is not directly related to the research topic of focus. In our study in particular, we are facing the issue about missing destination features in real passenger data. This issue is caused by entry-only transaction systems, which are globally popular. Unfortunately, the entry-only passenger data do not provide passenger destination information, limiting its direct use to construct passenger origin-destination (O-D) matrices which are vital for public transit service planning and adjustment.

Over the past two decades, substantial research interest has been placed in the field of O-D matrix estimation from the entry-only passenger data. Traditional approaches rely on heuristics to estimate the passenger destination information. Barry *et al.* [78] conducted one well-known preliminary work, in which they estimated destinations of entry-only automated fare collection (AFC) data from the New York City subway system using two assumptions:

- The most likely destination of the trip is the origin of the next trip; and
- The most likely final daily destination is the first daily origin.

They validated their methodology and assumptions using travel diary surveys. On top of this work, many researchers have developed variants to improve the robustness of the estimation by taking into account the distance limitation or by integrating additional data resources [79, 80, 81, 82, 83]. These heuristic-based methods can handle some of the entry-only data, but the remaining data usually account for 40–50% of the total.

In general, the ways in which researchers validate their estimations are bifurcated into *exogenous* and *endogenous* validations [15]. The former relies on external datasets, such as ground-truth O-D trips and household surveys, which are independent from the data being processed. Only a few studies have been able to apply exogenous validations [80, 82, 14, 84]. As a workaround, endogenous validations can ensure the consistency within the post-processed data, and most studies have been able to apply the endogenous method to validate their work [79, 83, 15, 34, 85].

In our thesis, we contribute a semi-supervised learning algorithm to conduct destination inference on top of heuristic-based methods. One strong point of our work is that we are able

to conduct exogenous validation that truly evaluates the performance of our method, because the traffic-passenger joint simulation (P-SUMO) can generate complete synthetic passenger data, and in this way, we have the destination information to validate the inference results.

6.4 Passenger Mobility Planning

This thesis proposes passenger mobility planning in the context of a public transit environment. In doing so, it takes into account the involvement of multiple criteria and the influence of multiple passengers. It is rooted in the fundamental shortest path search problem and extends into multi-criteria planning and multi-agent planning. This section contextualizes our work within the multi-criteria planning and multi-agent planning literature.

6.4.1 Shortest Path Algorithms

Most literature concerning mobility planning aims to solve the shortest path finding problem in transportation networks, which is also called *route planning* [58]. Let the transportation networks be formulated as a graph $G = (V, E)$ with a set V of vertexes and a set E of edges. Each edge $e(u, v) \in E$ has a length $l(u, v)$. The cost of a path from source s to destination (target) t is the sum of edge lengths. We denote the cost as distance $dist(s, t)$. The shortest path algorithm aims to find a path from the origin to the destination with minimal cost.

Dijkstra's algorithm [21] is a basic technique for efficiently solving the shortest path search, which lays the foundation for the field of route planning. This algorithm initializes all distances from source to infinity, except for $dist\,(s, s) = 0$, and adds s to a *priority queue* Q of vertexes, which are ordered in terms of the distance from the source. In each iteration, the algorithm expands the first-ranked vertex u in Q and investigates each of its neighbor vertexes v. For each neighbor vertex v, it computes the pending distance $dist\,(s, u) + l(u, v)$, and accepts this value to update $dist\,(s, v)$ to improve the previous distance from s to v. The vertex v along with the new distance $dist\,(s, v)$ is then added into Q. The algorithm continues until the destination t is expanded. A *bidirectional search* [86] is a modification of *Dijkstra's* algorithm that can reduce the search space. This algorithm runs a forward search from s and a backward search from t, and stops as soon as both search spaces intersect.

The *A* search* algorithm [56] generalizes *Dijkstra's* algorithm by cutting down the size of the sub-graph (subset of the search space) that must be explored and by guiding the search toward the destination with a heuristic h: $V \times V \to \mathbb{R}$, estimating the cost from current u to

t. Different from *Dijkstra's* method, the priority queue Q is ordered by the total cost $f(u) = dist(s,u) + h(u,t)$. The heuristic needs to be *admissible* (i.e., always less than or equal to the ground-truth cost) to guarantee the optimal solution. In route planning, the usual choice for the heuristic is the *Euclidean* distance between vertexes. This algorithm can also be combined with the bidirectional search for a speed-up.

There are many extensions to the aforementioned basic techniques to deal with the computing efforts on a *continental* scale. These algorithms require pre-processing, which could be very computationally heavy, to speed up the searching phase. Some examples include the ALT (A* + Landmarks + Triangle inequality) algorithm [87], which picks a small set $L \subseteq V$ of *landmarks* and stores the distance between vertexes in L and all other vertexes in the graph. During the searching phase, it uses triangle inequality pivoted on the landmarks as the heuristic. Specifically, for any landmark l_i, both $dist(u,t) \geq dist(u,l_i) - dist(t,l_i)$ and $dist(u,t) \geq dist(l_i,u) - dist(l_i,t)$ hold, and thus, one can take the maximum bound of all the heuristics if several landmarks are available.

The methods, *geometric containers* [88] and *arc flags* [89], associate the set of *leading-to-vertexes* to each edge and the set of *leading-to-regions* (a region contains a set of vertexes) to each edge, respectively. The former requires the pre-processing of the *all-pairs* shortest path computation, whereas the latter relaxes this requirement by computing *all-vertexes-to-regions* only. The sets associate to each edge can help the planner foresee that certain edge will lead to the destination vertex. In this way, a faster search can be achieved.

The *contraction hierarchies* [90] method introduces the *vertex contraction* operation during the pre-processing phase to assign vertexes into layers according to their "importance." The vertex with many edges (i.e., important) will be contracted first and placed on the low layer. At the same time, the "shortcuts" related to that vertex is inserted in the upper layer to connect neighbor vertexes that are bridged by the contracted vertex. The contraction operation reduces the number of edges on each layer, which can speed up the search. In the searching phase, the algorithm conducts a bidirectional search by looking "upward" and ignoring the lower-layer vertexes. When both directional searches meet at a higher-layer vertex, the shortest path is found.

The path planning in public transit networks is constrained on scheduled networks consisting of a set of stops, a set of routes, and a set of timed trips. By adjusting the graph to be schedule-based, we can directly apply the aforementioned shortest path algorithm to this problem.

6.4.2 Multi-Criteria Route Planning

When planning for passengers' experience, the shortest path algorithms, which only consider distance as the optimization criterion, are not enough. In public transit networks, however, other criteria, such as transit switching time and on-vehicle service, are just as important, leading to a multi-criteria planning problem.

There are mainly two versions of a multi-criteria search. The first one is to find a *Pareto set* (i.e., the maximum set of incomparable paths). In the field of operational research, two paths are considered incomparable if neither one is better than the other in all criteria and they are both members of the *non-dominated Pareto set*. This means that each criterion is equally important, expanding the search space into multiple dimensions. Several algorithms have been developed to find the Pareto set: The multi-criteria label-setting (MLS) algorithm [91] extends *Dijkstra's* algorithm by operating on *label* (i.e., a tuple with one entry per criterion), which indicates whether a path is dominated by another in lexicographically ordered criteria; on the contrary, the multi-criteria label-correcting (MLC) algorithm [92, 93] considers the whole *bag* of non-dominated labels. In this way, individual labels of a vertex can be scanned multiple times during the execution phase of the algorithm. Though some heuristic-based modifications are introduced to speed up the Pareto search, the search algorithms are not feasible due to their prohibitive computational complexity [94].

The other version is *scalarization* [95], which uses a linear combination of criteria as the optimization objective. This approach is to transform the multi-criteria search into a single-criterion search using a weighted sum of all criteria. In practice, scalarization approach dominates multi-criteria planning due to its efficient computation and flexibility because the weights can be tuned using some scale coefficients according to practical use.

6.4.3 Multi-Agent Path Planning

A multi-agent path planning (MAPP) finds a joint path for multiple agents conditioned on certain constraints. As discussed in Chapter 4, approaches for MAPP are folded into three main categories: coupled, decoupled, and intermediate. For coupled approaches, in addition to standard A*, enhanced variants of A* (e.g., operator decomposition [OD], enhanced partial expansion A* [EPEA*], and iterative deepening A* [IDA*]) can mitigate exponential growth in the number of neighbors by improving the admissible heuristics [42, 43, 44]. Recently, M* and its variants [39, 41] has been proposed to conduct coupled search in a computationally efficient

manner. Decoupled methods are used more for a faster run-time purpose. New approaches have been introduced to guarantee completeness but not optimality [96, 97, 98, 99, 38].

As for intermediate approaches, we have discussed CBS/MA-CBS [46, 100, 101] in Chapter 4, which are optimal. The intermediate approaches grow the search space during path planning, so that the search space can initially be very small then grow only where necessary. The work of Al-Wahedi [102] shows an approach in which paths are found separately for each agent, followed by coupled planning in a window around conflicts, but does not return optimal paths. Van den Berg *et al.* [103] presented an approach to avoid sequentially executing single agent paths by identifying the minimal sets of agents that must execute a cooperative path. Melo and Veloso [104] developed a decentralized method, that applies to sparse interaction scenarios where agents act independently for most of the time and plans differently only when entering the known interaction regions.

Another important framework that can be combined with the abovementioned algorithms is the probabilistic planning, such as rapidly-exploring random trees (RRTs) and probabilistic roadmaps (PRMs) [105, 106]. Probabilistic-based approaches are sampling-based approaches. They draw samples from the graph to gradually capture the connectivity of the configuration space until converging to the solutions, which are complete and asymptotically optimal.

Our work is different from traditional MAPPs: We handle urban public transit environments, in which the number of client passengers (i.e., agents) is large; also, agents in traditional work are usually considered as the same, whereas the passengers in a city may have different travel preferences and require different planning, making the planning tasks more complicated.

Chapter 7

Conclusion and Future Work

In this chapter, we first summarize the contributions of the thesis and then discuss some possible avenues for future work.

7.1 Contributions

The key contributions of this thesis are as follows:

- **P-SUMO for simulating the on-vehicle experience** We develop a joint simulator P-SUMO to simulate the passengers' on-vehicle experience through the joint traffic-passenger modeling and simulation on a city scale. This platform captures not only the interactions among road traffic, public vehicles, and passengers but also the dynamics of the on-vehicle services, including Wi-Fi quality and space availability. The simulation logs provide detailed passenger experience information and public vehicles' state information, serving as the public transit data for post-processing. We generate complete passenger data from the simulator, and the synthetic passenger data present significant similarity to the real passenger data in terms of spatiotemporal distribution.
- **A destination inference algorithm** We leverage a semi-supervised learning method, called self-training, to infer the destination information in the entry-only data. To implement self-training, we design a base predictor to predict the missing destinations based on the statistics of a selected similarity-based "training set" as well as a selection strategy to select newly-labeled data with high prediction confidence to update the training set. We also incorporate personal historical information to modify the base predictor for improved ac-

curacy. Evaluations based on the simulation data and a subset of the real bus AFC data show that the proposed methods significantly outperform the baselines in accuracy.

- **SC-M*** The SC-M* is a generalized version of M* with soft-collision-free constraints, which can scale to solving the multi-agent path planning problem under the soft-collision context. The SC-M* tracks the collision score of each agent and place agents, whose collision scores exceed some thresholds into a soft-collision set for sub-dimensional expansion. We show that the SC-M* has advanced flexibility and scalability for efficiently solving MAPP problems under the soft-collision context and can handle complex environments (e.g., with multiple types of agents requesting multiple types of resources). We compare the SC-M* to other SC-based MAPP solvers and show the advantages and trade-offs of the SC-M* against baselines in terms of path cost, success rate, and run time.

- **A real-world demonstration of the SC-M*** We make a case study in which the SC-M* was applied to improve the passengers' on-vehicle experience for the bus transit system in Porto, Portugal. We optimize the mobility of multiple client passengers over multiple public services while limiting the collision probability. To customize the SC-M* to a real-world system, we construct an expectation graph for modeling the dynamics of the on-vehicle service dynamics and propose the forward-the-tail-agent strategy to address the passengers' non-synchronization problem. The planned paths are successfully executed in the Porto bus transit systems. Qualitative evaluations show the advantages of the SC-M* in real-world systems in terms of path cost, collision-free constraint, and scalability in run time and success rate.

7.2 Future Work

In this thesis, we address the challenges to planning for the optimal experience of multiple passengers in terms of their service preference given a complex, urban public transit network. There are multiple possible directions for future work.

- **Adding more sophisticated details to P-SUMO:** We can include more sophisticated behavioral details into our P-SUMO framework such as letting passengers change travel plans when missing a bus or after a long waiting period, considering weather conditions as a factor affecting buses and passenger behaviors, etc. In addition, the P-SUMO implemented in this thesis is highly compatible to many kinds of people mobility data, including cellphone

trajectory data, on-bus Bluetooth sensor data, and even social media data that can provide pieces of people mobility information. With more data sources available in the future, we can gradually improve the quality of the simulation to better capture the real-world passengers' on-vehicle environment.

- **Improving the SC-M*:** We can leverage advanced variants of M*, such as EPErM* and ODrM*, to remove the basic A* component in our planner. We believe that better performance can be obtained this way because these variants improve the coupled planner and policy generator (two important components in the basic M*), which are directly related to the M* bottlenecks that limit the planning scalability.

- **Re-planning for failure executions:** We can consider on-line planning to deal with the failures in executing the plans in the real world. The SC-M* generates plans based on the expectation graph in the planning phase, implying that failures may occur at some points during the real-world executions. Solving this problem requires a fast re-planner at the execution time from the location of failure to the destination given the current state of public networks, which is a promising direction in the future.

- **Making P-SUMO open-source:** P-SUMO is a simulation framework that uses an API (TraCI) to communicate with SUMO without changing such a sophisticated simulator itself. This implies that any researcher can conveniently transplant our experiments and build a P-SUMO for other cities. We will make our P-SUMO open-source and provide a tutorial online or specification website to guide people to establish their own passenger-centered simulator for any cities around the world.

- **Applying to real systems:** The ultimate goal of the algorithm introduced in this thesis is to serve the real-world passengers that use the public transit. There must be many practical issues undetected in our simulation-based experiments when deploying the algorithm to real transportation. We will keep improving the SC-M* by practicing in real systems.

Appendix A

SC-M* for Bottleneck-Featured Environment

We conduct additional one-resource-one-type experiments on another transit system with blocked areas, featuring some bottleneck channels (see Figure A.1). Main experimental results are presented below in tables, where we can observe same phenomena regarding success rate, path cost, and run time of the three SC-based MAPP algorithms. Specifically, SC-A* cannot scale well to handle a large number of agents. SC-CBS has better success rate and run time, but at the expense of larger path cost. SC-M* presents reasonable run time, path cost, and success rate among the three. Also, better scalability is shown in experiments with slacker soft-collision conditions.

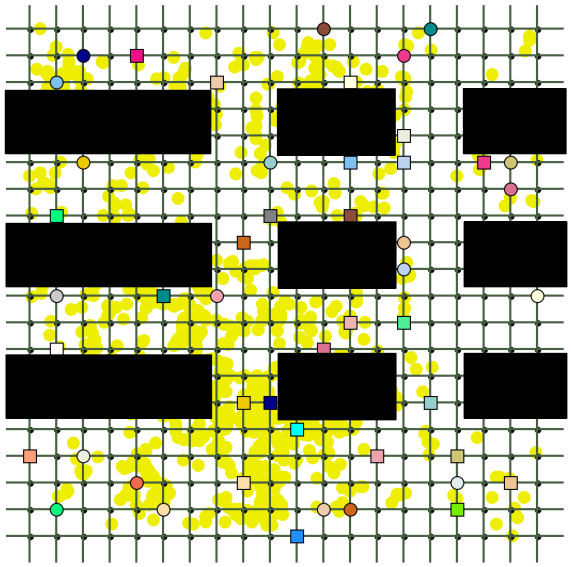

Figure A.1: Transit system with bottlenecks. Blocked areas are shown in black.

	SC-A* (T=0.35, δ=9.0)		
m	Success rate	Run-time	Additional cost
1	100%	0.87383615	0
2	100%	1.45812929	0
5	100%	8.75209618	0
6	100%	41.3597078	0
7	90%	698.139512	0
8	27%	963.238928	0
10	0%	>1000	NA

Table A.1: Records of the SC-A* applied to the bottleneck scenario, given $T = 0.35, \delta = 9.0$.

	SC-M* (T=0.25, δ=6.0)			SC-CBS (T=0.25, δ=6.0)		
m	Success rate	Run-time	Additional cost	Success rate	Run-time	Additional cost
2	100%	0.13148019	0	100%	0.14360401	0
5	100%	0.33542909	0	100%	0.34042161	0
10	100%	0.64117174	0	100%	0.69793972	0
20	100%	1.2541496	0.7145	100%	1.90348556	14.248
40	100%	3.41468452	53.167	95%	58.4426199	52.5578947
50	90%	103.997959	52.9838889	95%	74.8042114	188.086316
60	60%	431.751354	151.040833	90%	208.671149	181.924444
70	35%	657.911671	88.2914286	70%	470.251258	282.013571
80	25%	811.406023	179.05	40%	665.338238	241.6025

Table A.2: Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.25, \delta = 6.0$.

	SC-M* (T=0.25, δ=2.0)			SC-CBS (T=0.25, δ=2.0)		
m	Success rate	Run-time	Additional cost	Success rate	Run-time	Additional cost
2	100%	0.09713184	0	100%	0.14378764	0
5	100%	0.22357324	0	100%	0.33619203	0
10	100%	0.45624378	10.6095	100%	0.78694229	4.0695
20	100%	1.57431868	21.1675	95%	53.2365214	37.2278947
40	70%	303.706158	69.1428571	95%	101.067209	157.702105
50	35%	652.621218	119.564286	55%	591.450676	232.071818
60	15%	758.688175	166.957778	40%	679.158028	271.5525
70	5%	951.916869	204.59	15%	911.038766	215.556667

Table A.3: Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.25, \delta = 2.0$.

	SC-M* (T=0.05, δ=6.0)			SC-CBS (T=0.05, δ=6.0)		
m	Success rate	Run-time	Additional cost	Success rate	Run-time	Additional cost
2	100%	0.13849862	0	100%	0.09303264	0
5	100%	0.33108882	0	100%	0.24391812	0
10	100%	0.91813391	29.4235	100%	1.13260149	19.2485
20	95%	51.4242243	12.7347368	100%	5.61701282	108.716
40	60%	422.514588	123.440833	85%	190.493202	128.957647
50	45%	570.344027	219.467778	70%	394.644647	236.747143
70	0%	>1000	NA	10%	933.408541	315.69

Table A.4: Records of the SC-M* and SC-CBS applied to the bottleneck scenario, given $T = 0.05, \delta = 6.0$.

Bibliography

[1] R. Shi, P. Steenkiste, and M. Veloso, "Generating synthetic passenger data through joint traffic-passenger modeling and simulation," in *Proc. of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, Nov. 2018, pp. 3397–3402.

[2] S. Isaacman, R. Becker, R. Caceres, M. Martonosi, J. Rowland, A. Varshavsky, and W. Willinger, "Human mobility modeling at metropolitan scales," in *Proc. of the 10th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Low Wood Bay, Lake District, UK, June 2012, pp. 239–252.

[3] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti, "Unveiling the complexity of human mobility by querying and mining massive trajectory data," *The VLDB Journal*, vol. 20, no. 5, pp. 695–719, 2011.

[4] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He, "coRide: Carpool service with a win-win fare model for large-scale taxicab networks," in *Proc. of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Roma, Italy, Nov. 2013, pp. 9:1–9:14.

[5] M. P. Pelletier, M. Trepanier, and C. Morency, "Smart card data use in public transit: A literature review," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 2011, pp. 557–568, Aug. 2011.

[6] S. Bhattacharya, S. Phithakkitnukoon, P. Nurmi, A. Klami, M. Veloso, and C. Bento, "Gaussian process-based predictive modeling for bus ridership," in *Proc. of the 2013 ACM International Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp'13 Adjunct)*, Zurich, Switzerland, Sep. 2013, pp. 1189–1198.

[7] N. Lathia and L. Capra, "How smart is your smartcard?: Measuring travel behaviours, perceptions, and incentives," in *Proc. of the 13th ACM International Conference on Ubiquitous Computing (UbiComp)*, Beijing, China, Sep. 2011, pp. 291–300.

[8] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, pp. 779–782, June 2008.

[9] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.

[10] F. Simini, M. C. Gonzalez, A. Maritan, and A.-L. Barabasi, "A universal model for mobility and migration patterns," *Nature*, vol. 484, no. 7392, pp. 96–100, April 2012.

[11] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi-passenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, Sep. 2013.

[12] M. Chen, X. Liu, J. Xia, and S. I. Chien, "A dynamic bus-arrival time prediction model based on APC data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 19, no. 5, pp. 364–376, Sep. 2004.

[13] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Time-evolving O-D matrix estimation using high-speed GPS data streams," *Expert Systems with Applications*, vol. 44, no. 2016, pp. 275–288, Feb. 2016.

[14] L. Moreira-Matias and O. Cats, "Toward a demand estimation model based on automated vehicle location," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2544, no. 1, pp. 141–149, Feb. 2016.

[15] A. A. Nunes, T. Galvao Dias, and J. Falcao e Cunha, "Passenger journey destination estimation from automated fare collection system data using spatial validation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 133–142, Jan. 2016.

[16] Y. Wang, S. Ram, F. Currim, E. Dantas, and L. A. Saboia, "A big data approach for smart transportation management on bus network," in *Proc. of the 2016 IEEE International Smart Cities Conference (ISC2)*, Trento, Italy, Sep. 2016, pp. 1–6.

[17] "Trajectory – Prediction Challenge Dataset, ECML/PKDD 2015," http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html.

[18] "STCP annual report and accounts," STCP, Porto, Portugal, 2016.

[19] B. W. Silverman, *Density estimation for statistics and data analysis*. Boston, MA, USA: Chapman & Hall, 1992.

[20] "Transport Assessment Best Practice: Guidance Document," Transport for London, London, UK, April 2010.

[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 1991.

[23] L. L. Peterson and B. S. Davie, *Computer Networks, 5th Edition: A systems approach*. Burlington, MA, USA: Morgan Kaufmann Publishers, 2011.

[24] M. Antoni, "Comparison of 802.11ac and 802.11n PHY layers," *Kwartalnik Naukowy Uczelni Vistula*, vol. 2, no. 40, pp. 111–123, 2014.

[25] "Cisco wireless mesh access points, design and deployment guide, release 7.3." https://www.cisco.com/c/en/us/td/docs/wireless/technology/mesh/7-3/design/guide/Mesh/Mesh_chapter_0100.html.

[26] R. Shi, P. Steenkiste, and M. Veloso, "Second-order destination inference using semi-supervised self-training for entry-only passenger data," in *Proc. of the 4th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, Austin, TX, USA, Dec. 2017, pp. 255–264.

[27] M. Yin, M. Sheehan, S. Feygin, J.-F. Paiement, and A. Pozdnoukhov, "A generative model of urban activities from cellular data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1682–1696, June 2018.

[28] J. Jung and K. Sohn, "Deep-learning architecture to forecast destinations of bus passengers from entry-only smart-card data," *IET Intelligent Transport Systems*, vol. 11, no. 6, pp. 334–339, Mar. 2017.

[29] X. Chen, Z. Qian, R. Rajagopal, T. Stiers, C. Flores, R. Kavaler, and F. Williams III, "Parking sensing and information system: Sensors, deployment, and evaluation," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2559, no. 1, pp. 81–89, Jan. 2016.

[30] M. Carey, C. Hendrickson, and K. Siddharthan, "A method for direct estimation of origin/destination trip matrices," *Transportation Science*, vol. 15, no. 1, pp. 32–49, Feb. 1981.

[31] X. Zhu, "Semi-supervised learning literature survey," Tech. Rep. 1530, Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, 2005.

[32] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 1, pp. 355–370, Feb. 2017.

[33] Y. Li, C. Guan, H. Li, and Z. Chin, "A self-training semi-supervised SVM algorithm and its application in an EEG-based brain computer interface speller system," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1285–1294, July 2008.

[34] M. Trepanier, N. Tranchant, and R. Chapleau, "Individual trip destination estimation in a transit smart card automated fare collection system," *Journal of Intelligent Transportation Systems*, vol. 11, no. 1, pp. 1–4, April 2007.

[35] R. Shi, P. Steenkiste, and M. Veloso, "SC-M*: A multi-agent path planning algorithm with soft-collision-free constraint," In submission.

[36] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, no. 2008, pp. 591–656, Mar. 2008.

[37] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, Dec. 2007.

[38] D. Silver, "Cooperative pathfinding," in *Proc. of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, Marina del Rey, CA, USA, June 2005, pp. 117–122.

[39] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proc. of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, Sep. 2011, pp. 3260–3267.

[40] D. Ratner and M. K. Warmuth, "Finding a shortest solution for the NxN extension of the 15-PUZZLE is intractable," in *Proc. of the 5th AAAI Conference on Artificial Intelligence (AAAI)*, Philadelphia, PA, USA, Aug. 1986, pp. 168–172.

[41] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, Feb. 2015.

[42] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, Atlanta, GA, USA, July 2010, pp. 173–178.

[43] A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. R. Sturtevant, J. Schaeffer, and R. Holte, "Partial-expansion A* with selective node generation," in *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, Toronto, Ontario, Canada, July 2012, pp. 471–477.

[44] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artificial Intelligence*, vol. 27, no. 1, pp. 97–109, Sep. 1985.

[45] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. of the 2002 IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, USA, May 2002, pp. 2112–2119.

[46] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, Feb. 2015.

[47] C. Ferner, G. Wagner, and H. Choset, "ODrM* optimal multirobot path planning in low dimensional search spaces," in *Proc. of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 3854–3859.

[48] G. Wagner and H. Choset, "Path planning for multiple agents under uncertainty," in *Proc. of the 27th International Conference on Automated Planning and Scheduling (ICAPS)*, Pittsburgh, PA, USA, July 2017, pp. 577–585.

[49] H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, San Francisco, CA, USA, Feb. 2017, pp. 3605–3612.

[50] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proc. of the 2006 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, Pisa, Italy, Sep. 2006, pp. 99–110.

[51] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, and F.-R. Kline, "The MIT – Cornell collision and why it happened," *Journal of Field Robotics*, vol. 25, no. 10, pp. 775–807, Oct. 2008.

[52] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, Oct. 2008.

[53] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion A*," *Journal of Artificial Intelligence Research*, vol. 50, no. 1, pp. 141–187, May 2014.

[54] M. R. Jansen and N. R. Sturtevant, "Direction maps for cooperative pathfinding," in *Proc. of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, Stanford, CA, USA, Oct. 2008, pp. 185–190.

[55] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, June 2011.

[56] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[57] R. Shi, P. Steenkiste, and M. Veloso, "Improving the on-vehicle experience of passengers through SC-M*: A scalable multi-passenger multi-criteria mobility planner," In submission.

[58] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route planning in transportation networks," in *Algorithm Engineering: Selected Results and Surveys*. Cham, Switzerland: Springer, Nov. 2016, pp. 19–80.

[59] M. Müller-Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis, "Timetable information: Models and algorithms," in *Algorithmic Methods for Railway Optimization*. Berlin, Heidelberg: Springer, 2007, pp. 67–90.

[60] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in Europe," *Journal of Urban Technology*, vol. 18, no. 2, pp. 65–82, April 2011.

[61] S. Elkosantini and S. Darmoul, "Intelligent public transportation systems: A review of architectures and enabling technologies," in *Proc. of the 2013 International Conference on Advanced Logistics and Transport (ICALT)*, Sousse, Tunisia, May 2013, pp. 233–238.

[62] Z.-R. Peng, E. A. Beimborn, S. Octania, and R. J. Zygowicz, "Evaluation of the benefits of automated vehicle location systems in small and medium sized transit agencies," Tech. Report, Center For Urban Transportation Studies, University of Wisconsin-Milwaukee, 1999.

[63] L. Zhang, S. D. Gupta, J.-Q. Li, K. Zhou, and W.-B. Zhang, "Path2Go: Context-aware services for mobile real-time multimodal traveler information," in *Proc. of the 14th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, Oct. 2011, pp. 174–179.

[64] X. Chu, "A guidebook for using automatic passenger counter data for national transit database (NTD) reporting," Tech. Report, National Transit Resource Center, 2010.

[65] P. Borne, B. Fayech, S. Hammadi, and S. Maouche, "Decision support system for urban transportation networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 33, no. 1, pp. 67–77, 2003.

[66] C. Wang, J. Ge, , and X. Xu, "Analysis of air traffic flow control through agent-based modeling and simulation," in *Proc. of the 2009 International Conference on Computer Modeling and Simulation (ICCMS)*, Macau, China, Feb. 2009, pp. 286–290.

[67] Y. Yang, Y. Lu, L. Jia, Y. Qin, and H. Dong, "Optimized simulation on the intersection traffic control and organization based on combined application of simulation softwares," in *Proc. of the 24th Chinese Control and Decision Conference (CCDC)*, Taiyuan, China, May 2012, pp. 3787–3792.

[68] X. Zou, Z. Wang, L. Zheng, H. Dong, L. Jia, and Y. Qin, "Traffic impact analysis of urban construction projects based on traffic simulation," in *Proc. of the 24th Chinese Control and Decision Conference (CCDC)*, Taiyuan, China, May 2012, pp. 3923–3927.

[69] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[70] T. Schelenz, A. Suescun, M. Karlsson, and L. Wikstrom, "Decision making algorithm for bus passenger simulation during the vehicle design process," *Transport Policy*, vol. 25, no. 2013, pp. 178–185, 2013.

[71] A. Sumalee, Z. Tan, and W. H. Lam, "Dynamic stochastic transit assignment with explicit seat allocation model," *Transportation Research Part B: Methodological*, vol. 43, no. 8-9, pp. 895–912, 2009.

[72] J.-D. Schmocker, A. Fonzone, H. Shimamoto, F. Kurauchi, and M. G. Bell, "Frequency-based transit assignment considering seat capacities," *Transportation Research Part B: Methodological*, vol. 45, no. 2, pp. 392–408, 2011.

[73] Q. Zhang, B. Han, and D. Li, "Modeling and simulation of passenger alighting and boarding movement in Beijing metro stations," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 5, pp. 635–649, 2008.

[74] M. Fellendorf, "Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority," in the 64th Institute of Transportation Engineers Annual Meeting, pp. 1–9, 1994.

[75] "Aimsun next 8.2 user's manual," Barcelona, Spain, 2017.

[76] H. Ayed, D. Khadraoui, and R. Aggoune, "Using MATSim to simulate carpooling and carsharing trips," in *Proc. of the 2015 World Congress on Information Technology and Computer Applications (WCITCA)*, Hammamet, Tunisia, June 2015, pp. 1–5.

[77] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO – Simulation of Urban MObility," *International Journal on Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, 2012.

[78] J. J. Barry, R. Newhouser, A. Rahbee, and S. Sayeda, "Origin and destination estimation in New York City with automated fare system data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1817, no. 1, pp. 183–187, 2002.

[79] J. Zhao, A. Rahbee, and N. Wilson, "Estimating a rail passenger trip origin-destination matrix using automatic data collection systems," *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 5, pp. 376–387, July 2007.

[80] J. M. Farzin, "Constructing an automated bus origin-destination matrix using farecard and global positioning system data in Sao Paulo, Brazil," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2072, no. 1, pp. 30–37, 2008.

[81] J. J. Barry, R. Freimer, and H. Slavin, "Use of entry-only automatic fare collection data to estimate linked transit trips in New York City," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2112, no. 1, pp. 53–61, 2009.

[82] W. Wang, J. Attanucci, and N. Wilson, "Bus passenger origin-destination estimation and related analyses using automated data collection systems," *Journal of Public Transportation*, vol. 14, no. 4, pp. 131–150, Mar. 2011.

[83] D. Li, Y. Lin, X. Zhao, H. Song, and N. Zou, "Estimating a transit passenger trip origin-destination matrix using automatic fare collection system," in *Proc. of the 16th International Conference on Database Systems for Advanced Applications (DASFAA)*, Hong Kong, China, April 2011, pp. 502–513.

[84] M. Munizaga, F. Devillaine, C. Navarrete, and D. Silva, "Validating travel behavior estimated from smartcard data," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 70–79, June 2014.

[85] L. Zhang, S. Zhao, Y. Zhu, and Z. Zhu, "Study on the method of constructing bus stops OD matrix based on IC card data," in *Proc. of the 3rd International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Shanghai, China, Sep. 2007, pp. 3147–3150.

[86] G. Dantzig, *Linear programming and extensions*. Princeton, NJ, USA: Princeton University Press, 2016.

[87] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A* search meets graph theory," in *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Vancouver, British Columbia, Canada, Jan. 2005, pp. 156–165.

[88] F. Schulz, D. Wagner, and K. Weihe, "Dijkstra's algorithm on-line: An empirical case study from public railroad transport," *Journal of Experimental Algorithmics (JEA)*, vol. 5, no. 12, pp. 1–23, Dec. 2000.

[89] M. Hilger, E. Kohler, R. H. Mohring, and H. Schilling, "Fast point-to-point shortest path computations with arc-flags," in *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*. Providence, RI, USA: American Mathematical Society, 2009, vol. 74, pp. 41–72.

[90] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," *Transportation Science*, vol. 46, no. 3, pp. 388–404, Aug. 2012.

[91] E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, May 1984.

[92] B. C. Dean, "Continuous-time dynamics shortest path algorithms," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1999.

[93] D. Delling and D. Wagner, "Time-dependent route planning," in *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 207–230.

[94] J. Hrncir, P. Zilecky, Q. Song, and M. Jakob, "Speedups for multi-criteria urban bicycle routing," in *Proc. of the 15th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*, Patras, Greece, Sep. 2015, pp. 16–28.

[95] R. Geisberger, M. Kobitzsch, and P. Sanders, "Route planning with flexible objective functions," in *Proc. of the 12th Workshop on Algorithm Engineering and Experiments (ALENEX)*, Austin, TX, USA, Jan. 2010, pp. 124–137.

[96] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, Sep. 1986.

[97] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1, pp. 477–521, Nov. 1987.

[98] S. Leroy, J.-P. Laumond, and T. Simeon, "Multiple path coordination for mobile robots: A geometric algorithm," in *Proc. of the 6th International Joint Conference on Artificial Intelligence (IJCAI)*, San Francisco, CA, USA, July 1999, pp. 1118–1123.

[99] M. Saha and P. Isto, "Multi-robot motion planning by incremental coordination," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, Oct. 2006, pp. 5960–5963.

[100] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, "Conflict-based search for optimal multi-agent path finding," in *Proc. of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, Toronto, Canada, July 2012, pp. 563–596.

[101] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Meta-agent conflict-based search for optimal multi-agent path finding," in *Proc. of the 5th Annual Symposium on Combinatorial Search (SoCS)*, Niagara Falls, Canada, July 2012, pp. 39–40.

[102] K. Al-Wahedi, "A hybrid local-global motion planner for multi-agent coordination," Master's thesis, Case Western Reserve University, Cleveland, OH, USA, 2000.

[103] J. van Den Berg, J. Snoeyink, M. C. Lin, and D. Manocha, "Centralized path planning for multiple robots: Optimal decoupling into sequential plans," in *Proc. of the 5th Robotics: Science and Systems (RSS)*, Seattle, WA, USA, June 2009.

[104] F. S. Melo and M. Veloso, "Decentralized MDPs with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, July 2011.

[105] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[106] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, USA, April 2000, pp. 995–1001.