Understanding social and community dynamics from taxi GPS data

Chao Chen

THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Chao CHEN

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Exploration de la dynamique sociale et collective en utilisant les données GPS de taxi

Soutenue le 4 juillet 2014

Devant le jury composé de :

Fabien Moutarde	**Rapporteur**	**Professeur**	**Mines ParisTech – Paris - France**
Vania Bogorny	**Rapporteur**	**Professeur**	**Universidade Federal de Santa Catarina – Brasil**
Thierry Artières	**Examinateur**	**Professeur**	**UPMC – Paris – France**
Nazim Agoulmine	**Examinateur**	**Professeur**	**Université d'Evry Val d'Essonne – France**
Animesh Pathak	**Invité**	**Chercheur**	**INRIA Paris-Rocquencourt - France**
Tülin Atmaca	**Directrice de thèse**	**Professeur**	**Institut Mines-Télécom – Evry – France**
Daqing Zhang	**Codirecteur de thèse**	**Professeur**	**Institut Mines-Télécom – Evry – France**

Thèse n° **2014TELE0015**

Doctor of Philosophy (PhD) Thesis
Université Pierre & Marie Curie -TELECOM SudParis

Specialization

Informatique

presented by

Chao CHEN

Submitted for the partial requirement of

Doctor of Philosophy
from
Université Pierre & Marie Curie (UPMC) - TELECOM SudParis

Understanding Social and Community Dynamics from Taxi GPS Data

July 4, 2014

Commitee:

Fabien Moutarde	Reviewer	Associate Professor, Mines ParisTech – Paris - France
Vania Bogorny	Reviewer	Professor, Universidade Federal de Santa Catarina – Brasil
Thierry Artières	Examiner	Professor, UPMC – Paris - France
Nazim Agoulmine	Examiner	Professor, University of Evry Val d'Essonne - France
Animesh Pathak	Guest	Researcher, INRIA Paris-Rocquencourt - France
Tülin Atmaca	Thesis Director	Professor, Institut Mines-Télécom – Evry - France
Daqing Zhang	Advisor	Professor, Institut Mines-Télécom – Evry - France

Declaration

I, Chao Chen, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature:

Abstract

Personal mobile devices such as smart phones, portable computers and GPS localizers have become an essential element in people's daily life. They leave digital footprints of their user's daily activities and their surrounding contexts (e.g. the noise, the air quality, the earthquake), which are a reflection of the economical, societal and environmental interactions of a community. We have entered an era where such digital footprints are becoming increasingly big and easily available. Big digital footprints provide us with rich data sources to obtain a better and deeper understanding of the underlying social and community dynamics (dynamics of an individual, community or city). This understanding can further enable many innovative applications and urban services for improving the living quality/safety of citizens, sustainable city development for smart cities.

Taxis equipped with GPS sensors are an important sensory device for examining people's movements and activities. As oppose to public transportation and private vehicles, they serve the transportation needs of a large number of people driven by diverse needs, and are not constrained to a pre-defined schedule/route. Thus, the big taxi GPS data recording the spatio-temporal traces left by taxis provides a richer and more detailed glimpse into the motivations, behaviours, and resulting dynamics of a city's mobile population through the road network.

In this dissertation, motivated by applying pervasive sensing, communication and computing technology to bring citizens closer to the vision of a smart city, we aim to uncover the "hidden facts" regarding social and community dynamics encoded in the taxi GPS data to better understand how urban population behaves and the resulting dynamics in the city. As some "hidden facts" are with regard to similar aspect of social and community dynamics, we further formally define three categories for study (i.e. *social dynamics, traffic dynamics, and operational dynamics*), and explore them to fill the wide gaps between the raw taxi GPS data and innovative applications and smart urban services. Specifically,

- To enable applications of real-time taxi fraud alerts, we propose *iBOAT* algorithm which is capable of detecting anomalous trajectories "on-the-fly" and identifying which parts of the trajectory are responsible for its anomalousness, by comparing them against historically trajectories having the same origin and destination. We verify the superior performance of *iBOAT* over the state-of-art algorithms on a big taxi GPS data set, containing 7.35 million trips which was generated by 7,600 taxis in a month in Hangzhou, China. We further demonstrate the ability of *iBOAT* in detecting road network changes. This work is mainly related to the understanding of *operational dynamics* about the behaviours of taxi drivers when delivering passengers (i.e. honest or not).
- To introduce cost-effective and environment-friendly transport services to citizens, we propose *B-Planner* which is a two-phase approach, to plan bi-directional night bus routes leveraging big taxi GPS data since it can correctly characterize the passenger flows at nighttime. We formulate the problem as the route planning problem with

objective of maximizing the number of passengers expected along the route under a couple of constraints, such as the total travel time, the bus frequency. To validate the effectiveness of the proposed approach, extensive empirical studies are performed on a big real-world taxi GPS data set which contains more than 1.57 million night passenger delivery trips, generated by 7,600 taxis in a month in Hangzhou, China. This work is mainly related to the understanding of *social dynamics* about where are the popular passenger pick-up/drop-off locations and origin-destination (i.e. OD) pairs at nighttime, and the understanding of *traffic dynamics* about how much driving time is needed to travel between popular OD pairs at nighttime.

- To offer a personalized, interactive, and traffic-aware trip route planning system to users, we propose TripPlanner system which contains both offline and online procedures, leveraging a combination of Location-based Social Network (i.e. LBSN) and taxi GPS data sets. In the offline procedure, we construct a dynamic POI network by extracting relevant information from crowdsourced LBSN and taxi GPS trace data. In the online procedure, we propose a two-phase approach for personalized trip planning. We also formulate this problem as the route planning problem with the objective of maximizing the route score and satisfying both the venue visiting time and total travel time constraints. To validate the efficiency and effectiveness of the proposed approach, extensive empirical studies are performed on two big real-world data sets which contain more than 391,930 passenger delivery trips generated by 536 taxis in a month, and more than 110,200 check-ins left by over 15,680 Foursquare users in 6 months in San Francisco, US. This work is mainly related to the understanding of *traffic dynamics* about how much driving time is needed to transit between any two points in the city at different departure time of the day and day of the week.

Finally, some promising research directions for future work are pointed out, which mainly attempt to fuse taxi GPS data with other data sets (e.g. open data released by governments, various types of sensory data recorded by smart phones) to provide smarter and personalized urban services for citizens.

Keywords

digital footprints, taxi GPS data, smart city, social dynamics, traffic dynamics, operational dynamics, anomalous trajectory detection, bus route planning, trip route planning.

Résumé

Les appareils mobiles personnels comme les téléphones intelligents, les ordinateurs portables et les navigateurs GPS sont devenus un élément essentiel dans la vie quotidienne des gens. Ils laissent des empreintes numériques des activités quotidiennes de leur utilisation et de leurs contextes environnants (par exemple, le bruit, la qualité de l'air, le tremblement de terre), qui sont le reflet des interactions économiques, sociales et environnementales d'une communauté. Ces empreintes numériques nous offrent de riches sources de données afin d'obtenir une compréhension meilleure et plus profonde des dynamiques sociales et communautaires sous-jacentes (dynamique d'un individu, de la communauté ou de la ville). Cette compréhension peut en outre permettre de nombreuses applications innovantes et des services urbains pour améliorer la qualité de vie/sécurité des citoyens, et le développement durable de la ville pour les villes intelligentes.

Les taxis équipés de capteurs GPS sont un dispositif sensoriel important pour examiner les mouvements et les activités des gens. Par opposition aux véhicules de transport public et privé, ils répondent aux besoins de transport d'un grand nombre de personnes avec une grande diversité des besoins, et ne sont pas limitées à un horaire/itinéraire pré-défini. Ainsi, les empreintes GPS des taxis un aperçu plus riche et plus détaillée sur les motivations, les comportements et la dynamique résultant de population mobile d'une ville à travers le réseau routier.

Dans cette thèse, motivée par l'application de détection omniprésente, de la communication et de la technologie informatique pour offrir aux urbanistes et aux citoyens de nombreux points de vue et des services, nous cherchons à découvrir les "facts cachées" codées dans les traces GPS des taxis pour combler les grands écarts entre les données de détection brut et les applications, les rapprocher de la vision d'une ville intelligente. On définit trois catégories de dynamiques sociales et communautaires (par exemple, la dynamique sociale, la dynamique de la circulation, et la dynamique de fonctionnement), et d'explorer la diversité intelligence cachée pour permettre à plusieurs applications d'innovation et de services urbains. Plus spécifiquement :

- Pour permettre aux applications d'alertes de fraude de taxi en temps réel et réseau routier change de détection, nous proposons *iBOAT* algorithme qui est capable de détecter "à la volée" des trajectoires anormales et déterminer quelles parties de la trajectoire sont responsable de son anormalité, en les comparant à des trajectoires historiques ayant la même origine et destination. Nous vérifions la performance supérieure de *iBOAT* sur les algorithmes de l'état-de-art sur une grand échelle des traces de taxi GPS, contenant 7,35 millions de voyages qui a été généré par 7,600 taxis dans un mois à Hangzhou, en Chine.
- Pour introduire des services de transport respectueux de l'environnement aux citoyens, nous proposons *B-Planner* qui est une approche en deux phases, pour planifier des itinéraires de bus de nuit bi-directionnelles en exploitant les empreintes GPS de taxi, car elles peuvent bien caractériser les flux de passagers pendant la nuit. Nous

formulons le problème comme un problème de planification d'itinéraire avec pour objectif de maximiser le nombre de passagers attendus le long de la route sous un autre couple de contraintes, telles que le temps voyage au total, la fréquence du bus. Afin de valider l'efficacité de la approche proposée, des études empiriques approfondies sont effectuées sur un ensemble de données GPS réel de taxis qui contient plus de 1,57 million de nuit les trajets de livraison de passagers, générés par 7,600 taxis dans un mois à Hangzhou, en Chine.

- Pour offrir un système de planification d'itinéraire personnalisé, interactif, et le trafic-courant pour les utilisateurs, nous proposons système TRIPPLANNER qui contient à la fois en ligne et hors ligne des procédures, en s'appuyant sur une combinaison de géolocalisation réseau social et des ensembles de données de taxi GPS. Dans la procédure hors ligne, nous construisons un réseau de POI dynamique en extrayant des informations pertinentes de LBSN et des traces GPS de taxis. Dans la procédure en ligne, nous proposons une approche en deux phases pour la planification de voyage personnalisé. nous formulons également ce problème comme le problème de planification d'itinéraire avec l'objectif de maximiser le score de l'itinéraire et de satisfaire à la fois le lieu de visite temps et le total des contraintes de temps de Voyage. Pour valider l'efficacité de l'approche proposée, études empiriques approfondies sont effectuées sur deux ensembles de données du monde réel qui contiennent plus que 391,930 trajets de livraison de passagers générés par 536 taxis pour un mois, et plus de 110,200 check-ins laissés par plus de 15,680 utilisateurs Foursquare pendant 6 mois à San Francisco, États-Unis.

Enfin, nous abordons également quelques directions de recherche prometteuses pour les travaux futurs, qui tentent d'explorer les informations complémentaires fournies par les autres empreintes digitales et les traces de taxi GPS.

Mots-clés

empreintes numériques, traces de taxi GPS, dynamique sociale, dynamique de la circulation, dynamique opérationnelle, détection de trajectoire anormale, planification d'itinéraire de bus, planification d'itinéraire touristique.

To my dearest family.

Acknowledgements

There are many people I would like to thank for all of their support in making this thesis possible.

Firstly, I would like to express my deepest gratitude to my supervisors, Prof. Daqing Zhang and Tülin Atmaca, for their continuous support, guidances and encouragements, which are necessary to survive and thrive the graduate school and the beyond. I also want to thank Prof. Zhi-Hua Zhou from Nanjing University, for his precious guidances, and Prof. Shijian Li from Zhejiang University, for providing the valuable taxi GPS data set, during research collaboration. In particular, I would thank Prof. Daqing Zhang for his generously giving me motivation, support, time, assistance, opportunities and friendship ; for leading me how to identify key and influential problems, present and evaluate the ideas. His selfless and continuous help makes me to develop to be a better researcher, writer, and speaker gradually. I also want to thank all jury members, especially two reviewers, Prof. Fabien Moutarde from Mines ParisTech and Prof. Vania Bogorny from Universidade Federal de Santa Catarina. Their professional suggestions did improve the quality. Thanks also go to the China Scholarship Council (CSC), for the financial support of my Ph.D. study.

Secondly, I also sincerely thank my talent colleagues who work/ed with me in the lab, including Nan Li, Dr. Bin Li, Dr. Pablo Castro, Dr. Zhu Wang, Dr. Zhangbing Zhou, Dr. Lin Sun, Dr. Kejun Du, Dr. Mossaab Hariz, Yang Yuan, Haoyi Xiong, Dingqi Yang, Leye Wang, Dr. Bin Guo, and Dr. Zhiyong Yu. It is a great experience to work with these smart people ! I was provided lots of useful feedback and suggestions during discussions.

Thirdly, I would like to show my gratitude to Madame Françoise Abad and Xayplathi Lyfoung, for their kindness and help during my stay in Institut Mines-Télécom/Télécom SudParis. I also warmly thank all my great and nice friends I have made during my stay in France. Every time when I feel frustrated, they are always ready to listen to my complaints, and their encouragements indeed help me recover immediately. In addition, we share the pain and pressure of the Ph.D. studies or work at Télécom SudParis. They are Mingyue Qi, Guoqin Zhao, Xiao Han and Zhuowei Chen. Special thanks go to Zhuowei Chen for helping proofread the French abstract.

Last but not least, my parents Qijun Chen, Miaojun Xu and elder brother Yao Chen always give me the infinite love and support. Although they will not read this thesis, I cannot complete this journey without their love and support.

Chao @ Paris, France
24th, February, 2014

Publications

The following papers, published, in press or submitted, are the partial outputs of my Ph.D. studies in UPMC and Télécom SudParis.

Journal Papers

— **Chao Chen**, Daqing Zhang, Bin Guo, Xiaojuan Ma, Gang Pan and Zhaohui Wu. TRIPPLANNER : *Personalized Trip Planning Leveraging Heterogeneous Crowdsourced Digital Footprints*. IEEE Transactions on Intelligent Transportation Systems (T-ITS), major revision.

— **Chao Chen**, Daqing Zhang, Nan Li and Zhi-Hua Zhou. *B-Planner : Planning Bidirectional Night Bus Routes Using Large-scale Taxi GPS Traces*. IEEE Transactions on Intelligent Transportation Systems (T-ITS), 2014. (in press).

— **Chao Chen**, Daqing Zhang, Pablo S. Castro, Nan Li, Lin Sun, Shijian Li and Zonghui Wang. *iBOAT : Isolation-based On-line Anomalous Trajectory Detection*. IEEE Transactions on Intelligent Transportation Systems (T-ITS), 14(2) : 806-818, 2013.

— Daqing Zhang, **Chao Chen**, Zhangbing Zhou and Bin Li. *Identifying Logical Location via GPS-Enabled Mobile Phone and Wearable Camera*. International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 26(8) : 1-23, 2012.

— Pablo S. Castro, Daqing Zhang, **Chao Chen**, Shijian Li and Gang Pan. *From Taxi GPS Traces to Social and Community Dynamics : A Survey*. ACM Computing Surveys (CSUR), 46(2) : 17 :1-17 :34, 2013.

— Lin Sun, Daqing Zhang, **Chao Chen**, Pablo S. Castro, Shijian Li and Zonghui Wang. *Real Time Anomalous Trajectory Detection and Analysis*. Mobile Networks and Applications (MONET), 18(3) : 341-356, 2013.

— Daqing Zhang, Lin Sun, Bin Li, **Chao Chen**, Gang Pan, Shijian Li and Zhaohui Wu. *Understanding Taxi Service Strategies from Taxi GPS Traces*. IEEE Transactions on Intelligent Transportation Systems (T- ITS), 2014. (in press).

— Bin Guo, **Chao Chen**, Daqing Zhang, Zhiwen Yu, Alvin Chin and Athanasios Vasilakos. *Mobile Crowd Sensing : When Participatory Sensing Meets Participatory Social Media*. IEEE Computer Magazine, submitted.

— Zhu Wang, Zhiwen Yu, Xingshe Zhou, **Chao Chen** and Bin Guo. *Towards Context-Aware Mobile Web Browsing*. IEEE Systems Journal, submitted.

Conference Papers

— **Chao Chen**, Daqing Zhang and Bin Guo. *D2SC : Data-Driven Smarter Cities*. In Proceedings of IEEE PerCom Workshops, Budapest, Hungary, 2014.

— **Chao Chen**, Daqing Zhang, Zhi-Hua Zhou, Nan Li, Tülin Atmaca and Shijian Li. *B-Planner : Night Bus Route Planning Using Large-scale Taxi GPS Traces*. In Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications (PerCom'13), San Diego, USA, 2013.

— **Chao Chen**, Daqing Zhang, Lin Sun, Mossaab Hariz and Bruno Jean-Bart. *AQUEDUC : Improving Quality and Efficiency of Care for Elders in Real Homes*. In Proceedings of International Conference on Smart Homes and Health Telemetics (ICOST'13), Singapore, 2013.

— **Chao Chen**, Daqing Zhang, Lin Sun, Mossaab Hariz and Yang Yuan. *Does Location Help Daily Activity Recognition ?* In Proceedings of International Conference on Smart Homes and Health Telemetics (ICOST'12), Florence, Italy, 2012.

— **Chao Chen**, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun and Shijian Li. *Real-time Detection of Anomalous Taxi Trajectories from GPS Traces*. In Proceedings of 8th International ICST Conference on Mobile and Ubiquitous Systems (MobiQuitous'11), Copenhagen, Denmark, 2011. [**Best Paper Runner-up**]

— Longbiao Chen, Daqing Zhang, Gang Pan, Leye Wang, Xiaojuan Ma, **Chao Chen** and Shijian Li. *Container Throughput Estimation Leveraging Ship GPS Traces and Open Data*. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'14), Seattle, US, 2014.

— Daqing Zhang, Nan Li, Zhi-Hua Zhou, **Chao Chen**, Lin Sun and Shijian Li. *iBAT : Detecting Anomalous Taxi Trajectories from GPS Traces*. In Proceeding of the 13th ACM International Conference on Ubiquitous Computing (UbiComp'11), Beijing, China, 2011.

— Bin Li, Daqing Zhang, Lin Sun, **Chao Chen**, Shijian Li, Guande Qi and Qiang Yang. *Hunting or Waiting ? Discovering Passenger-Finding Strategies from a Large-scale Real-world Taxi Dataset*. In Proceedings of IEEE PerCom Workshops, Seattle, USA, 2011. [**Featured by IEEE Spectrum**]

Book Chapter

— Lin Sun, **Chao Chen** and Daqing Zhang. *Understanding City Dynamics from Taxi GPS Traces*. Creating Personal, Social and Urban Awareness through Pervasive Computing, IGI Global, 2013.

Table of contents

Chapter 1

Introduction

Contents

1.1 Background

Recent years have been witnessing a rapid development in a variety of technologies, such as sensing, communication, storage and computing. As a result, personal devices including smart phones, portable computers and GPS localizers become ubiquitous. They have revolutionized the way we interact with the cyber-physical worlds. They also leave digital footprints of their user's activities not only in daily life, but also their surrounding contexts (e.g., the noise, the air quality, the earthquake), which are a reflection of the economical, societal and environmental interactions of a community [157]. We have entered an era where such digital footprints are becoming increasingly big and easily available. Big digital footprints provide researchers with rich data sources to obtain a better understanding of the underlying dynamics of an individual, community or city [75]. This understanding further enables many innovative applications in building smart cities, including intelligent transportation, city planning, public safety, environmental sustainability, green computing and so on. However, different digital footprints reveal different aspects of the underlying social and community dynamics, depending on the type of digital footprints, and hence can support diverse applications and urban services. Thus, a very fundamental problem is to analyze the inherent characteristics of each digital footprint. For example, the trace data

left behind by GPS-equipped vehicles offer us an unprecedented window into the dynamics of a city's road network; the trace data left by Foursquare[1] users when checking-in venues can inform the dynamics of Point-of-Interests (POIs) in a city.

GPS-equipped vehicles offer an important kind of footprints since both public and private vehicles are the main transportation means for a city's population. People use vehicles for many purposes: for commuting between home and office, for regular and "irregular" chores, and for leisure activities, etc. By carefully analyzing the observed movement patterns of a population, researchers strive to better understand the demographics of a city (i.e., where do people go frequently at different time periods), the distribution of infrastructures around a city, the effectiveness of the public transportation networks, the dynamics of traffic conditions, and the different driving behaviours.

Public transportation vehicles equipped with GPS sensors provide rather predictable data since the vehicles in question follow fixed routes and stops under a specified schedule, such as public buses. Similarly, GPS-equipped private vehicles are usually restricted to one user for regular usages, e.g., the commuting usage, so they also follow fairly predictable routes. As opposed to public transportation and private vehicles, GPS-equipped taxis serve the transportation needs of a large number of people driven by diverse needs, and are not constrained to a pre-specified schedule/route. Moreover, taxi drivers work continuously in a whole day manner. Therefore, the big taxi GPS data recording the spatio-temporal traces left by taxis provides a much richer and more detailed glimpse into the motivations, behaviours, and resulting dynamics of a city's mobile population through the road network. In more details, many facts regarding social and community dynamics are hidden in the taxi GPS data when drivers are delivering or searching for passengers. As the motivations for passenger-delivery and passenger-finding are quite different, it is necessary to divide the trace data into passenger-delivery trajectories and passenger-finding trajectories respectively. Figure 1.1 illustrates a passenger-delivery trajectory which starts from pick-up point to drop-off point, and a passenger-finding trajectory which starts from drop-off point to the next pick-up point. Diverse facts can be uncovered through analyzing trajectories from different procedures (i.e. passenger-delivery procedure and passenger-finding procedure). For example,

- ♢ Many facts such as *where are high-demand passenger areas in the city at a given time*, *what strategies that good/average/bad taxi divers (measured by their revenues) had taken after dropping off passengers* are hidden in the historical passenger-finding trajectories accumulated by massive taxi drivers.
- ♢ Many facts such as *which road sequences that taxi drivers had taken to deliver passengers*, *how many taxis were there at a given road segment at a previous time period*,

1. http://foursquare.com

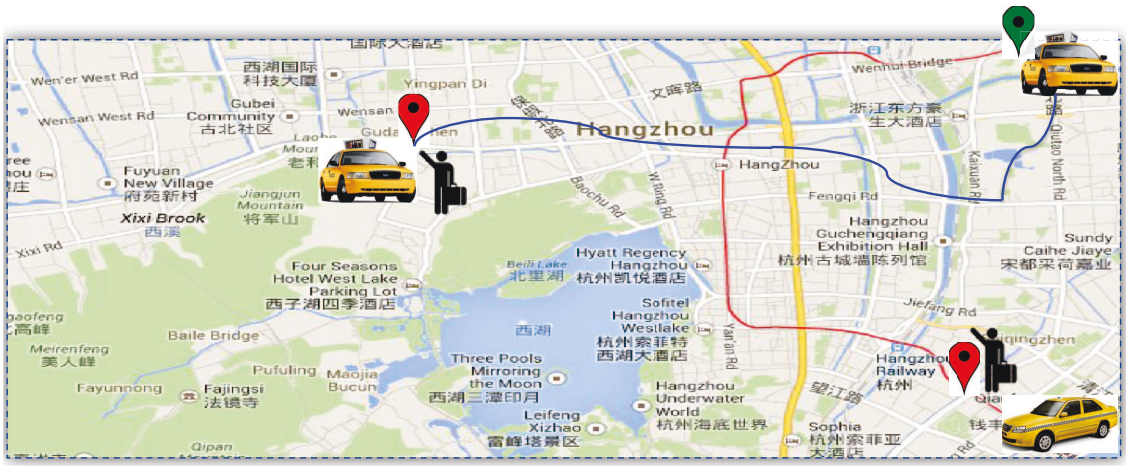


Figure 1.1: The trajectory in blue line is a passenger-delivery one while in the red line is a passenger-finding one. The red pinpoint marker denotes the passenger pick-up point; the green pinpoint marker denotes the passenger drop-off point.

and *how much driving time did it take to travel between two points in city* are hidden in the historical passenger-delivery trajectories accumulated by massive taxi drivers.

Leveraging on facts regarding social and community dynamics hidden in taxi GPS data ("hidden facts"), many innovative applications and smart urban services can be supported. Furthermore, based on the "hidden facts", with some predictive models, we could also foretell future dynamics since people often present some regularities. In this dissertation, the main focus is to uncover "hidden facts" from historical taxi GPS data. Some hidden facts are with regard to similar aspect of social and community dynamics, we thus further formally define three categories of social and community dynamics [28].

Social dynamics is defined as the study of the *collective behaviours* of a city's population, as observed by their movement in the city. It refers to the understanding of people's movement patterns in a city leveraging the *end-points* of the taxi GPS traces (i.e. pick-up and drop-off points, as shown in Figure 1.1), such as where are people going throughout the day, what are the "hottest" spots around a city, what are the "functions" of these hotspots, how strongly connected are different areas of the city, etc. A deep understanding of social dynamics can inform the passenger-demands for taxi drivers at different time and areas in a city, which is useful for recommending potential areas to drivers to find new passengers quickly. Besides, it is also essential for the management, design, maintenance and advancement of a city's infrastructures.

Traffic dynamics is about the resulting *flow* of the population through the *city's road network* since people will move around the city mainly through the road network, governed by their underlying desires or needs. It refers to the understanding of the congestion level

in the road network at different time. These congestion levels have a significant impact on important factors for drivers such as the travel time between two points, the expected speed, potential adverse traffic events such as accidents. The understanding of traffic dynamics is very useful for providing real-time traffic indicators (e.g. travel speed, traffic density) and route navigation for drivers, including both taxi drivers themselves and private vehicle drivers. In addition, they can be used to analyze certain side-effects of vehicle use, such as estimating pollution levels in a city. In the line of traffic dynamics study, we mainly make use of the taxi GPS trajectories in the *passenger-delivery* procedure (i.e., the blue line in Figure 1.1, since taxi drivers may not drive at a normal speed during passenger-finding procedure; they might intentionally drive slowly along roads when hunting new passengers.

Operational dynamics refers to the general study and analysis of taxi driver's *modus operandi*. The aim is to learn from taxi driver's excellent knowledge of the city, as well as to detect their abnormal behaviours. The understanding of operational dynamics is very useful for predicting future trajectories (e.g. next moving directions, destinations), suggesting strategies/routes for finding new passengers quickly, and suggesting navigational routes for reaching a destination efficiently. Additionally, new trajectories of passenger-delivery procedure can be compared against a large collection of historical trajectories to automatically detect abnormal behaviour of taxi drivers. In the study of operational dynamics, we make use of *full trajectories* in both procedures for many different purposes, as the routes taken by drivers are of utmost importance.

The research about taxi GPS data mining can benefit for a number of groups, mainly including taxi drivers, taxi passengers and city administrators.

- ♢ For taxi drivers, their major concern is to make more money while minimizing the cost of fuel. The essence is to increase the passenger occupied/free time ratio. Some taxi drivers can earn more money generally because they are good at finding new passengers after dropping off the last passengers, and simultaneously, they have good knowledge for choosing fast routes with low traffic to deliver passengers to the given destinations efficiently. Therefore, taxi drivers can decrease the passenger-free time by learning passenger-finding strategies from good taxi drivers; on the other hand, they can increase the passenger-occupied time by improving the driving performances through mining the passenger-delivery trajectories of good taxi drivers.
- ♢ For taxi passengers, they are quite interested in questions, such as "which nearby conner is the best and how long is needed to wait for a taxi at that corner", and "how much/long does it cost me to my destination" as well as "am I victim of a taxi fraud". All above-mentioned questions can be solved by leveraging the taxi GPS traces, and also many solutions have been offered. There are a plenty of apps running on smart phones available for daily use; some representative and popular apps are

list in Table 1.1, with a brief introduction of main functions, and the screen shots are also shown in Figure 1.2.

♢ For city administrators, such as taxi company managers and city planners, our research can enable many applications and urban services in building smart cities. To name a few, the location and status of taxis can be monitored in a real time manner, which in turn can be used to facilitate the taxi company managers to dispatch taxis directly. Taxi drivers are continuously driving on the roads around the city almost in the whole day, the collected GPS traces are thus a natural source for detecting city road network changes (e.g. road closure, new roads), and updating the digital map timely at a very low cost. With the help of the taxi GPS traces, city planners can detect the flawed problems in the planning timely, plan better public transportation routes to meet the demands of residents, and evaluate and redefine the current allocation of city infrastructures (e.g. bus stops, taxi stands).

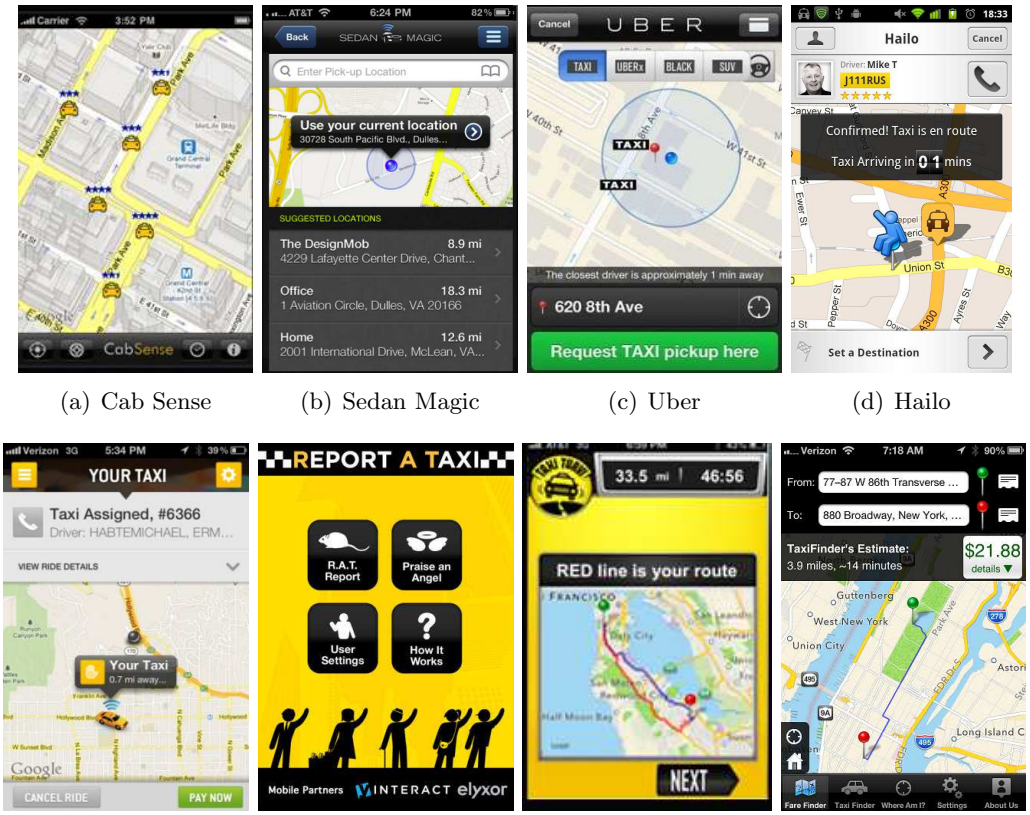
Table 1.1: Popular taxi-related apps running on smart phones.

Name	Functions
Cab Sense[1]	Find the best corner to catch a taxi.
Sedan Magic[2]	Taxi Booking.
Uber[3]	Taxi Booking.
Hailo[4]	Get a taxi wherever you are whenever you want; Pay by credit card.
Taxi Magic[5]	Booking rides via the app and text message; Managing rides in real time.
Report a Taxi[6]	Share positive and negative reviews about drivers.
Taxi Turvy[7]	Check whether drivers are taking the honest route.
TaxiFinder[8]	Taxi company lookup; Taxi fare estimates; Location lookup - where am I ?

[1] http://www.sensenetworks.com/products/macrosense-technology-platform/cabsense/
[2] http://sedanmagic.com/
[3] http://uber.com/
[4] http://hailocab.com/
[5] http://taximagic.com/
[6] http://reportataxi.com/
[7] http://www.newyork.com/articles/travel/new-taxi-turvi-app-44011/
[8] http://taxifinder.com/

1.2 Research Motivations and Contributions

The research work in this thesis is application-driven and motivated by applying pervasive sensing, communication and computing technology for *improving the living quality/safety of citizens, sustainable city development for smart cities*. The research work



(a) Cab Sense (b) Sedan Magic (c) Uber (d) Hailo

(e) Taxi Magic (f) Report a Taxi (g) Taxi Turvy (h) TaxiFinder

Figure 1.2: Screen shots of the popular apps.

presented attempts to "bridge the wide gaps between the raw taxi GPS data and applications and urban services" by leveraging the hidden facts revealed by the taxi GPS data, which include social dynamics, traffic dynamics, and operational dynamics. To better and deeper understand the social and community dynamics, many data mining techniques are exploited, including *clustering, classification, ranking and optimization*.

In the following, we will first present our motivations for concrete studies in the thesis, and then highlight our contributions one by one.

♢ Have you ever experienced a taxi fraud during your visit to an unfamiliar city? Trajectories obtained from GPS-enabled taxis can tell us the truth. Our first main work in this thesis is that we present a novel on-line anomaly detector (i.e. *iBOAT*) which is able to detect anomalous trajectories "on-the-fly" and to identify which parts of the trajectory are responsible for its anomalousness, by comparing them against historically trajectories having the same origin and destination. Trajectories occurring

between the same origin[2] and destination but different time may be not comparable since the traffic conditions are different, resulting in route chosen and driving behavours are also different. To exclude this effect, we simply divide the trajectories into different groups according to their occurring time, and perform *iBOAT* to compare the testing trajectory to those who also have the same occurring time. Furthermore, we conduct an in-depth analysis on around 43,800 anomalous trajectories that are detected out from the trajectories of 7,600 taxis in a month, revealing that most of the anomalous trips are the result of conscious decisions of greedy taxi drivers to commit fraud. Because some cunning taxi drivers may use detour reasons such as traffic accidents on roads as excuses, we also propose a simple mechanism to deny possible excuses for fraud behaviours. We evaluate our proposed method through extensive experiments on a large-scale taxi data set, and it shows that *iBOAT* achieves state-of-the-art performance, with a remarkable performance of the area under a curve (AUC)$\geq$0.99. We further demonstrate the *iBOAT*'s ability in detecting road network changes through various simulated experiments. This work is mainly related to the understanding of *operational dynamics* about the behaviours of taxi drivers when delivering passengers.

♢ In many cities, the daytime bus transportation systems are usually well designed; however, during late nights, most bus systems are out of service, leaving taxis as the only option for intra-city travelling. To provide cost-effective and environment-friendly transport to citizens *for sustainable city development*, many cities start to plan night-through bus routes. Our second main work in this thesis is that we intend to explore the night bus route planning issue by using taxi GPS traces, instead of leveraging the costly and inaccurate human surveys about people's mobility. Specifically, we propose a two-phase approach for bi-directional night-bus route planning (i.e. *B-Planner*). In the first phase, we develop a process to cluster "hot" areas with dense passenger pick-up/drop-off, and then propose effective methods to split big "hot" areas into clusters and identify a location in each cluster as a candidate bus stop. In the second phase, given the bus route origin, destination, candidate bus stops as well as bus operation time constraints, we derive several effective rules to build the bus route graph, and prune invalid stops and edges iteratively. Based on this graph, we further develop a Bi-directional Probability based Spreading (BPS) algorithm to generate candidate bus routes automatically. We finally select the best bi-directional bus route which expects the maximum number of passengers under the given conditions and constraints. To validate the effectiveness of the proposed approach, extensive empirical studies are performed on a real-world taxi GPS data

2. We use *origin* and *source* interchangeably throughout the thesis.

set which contains more than 1.57 million night passenger delivery trips, generated by 7,600 taxis in a month. This work is mainly related to the understanding of *social dynamics* about where are the popular passenger pick-up/drop-off locations and origin-destination pairs at nighttime, and the understanding of *traffic dynamics* about how much driving time is needed to travel between popular OD pairs at nighttime.

♢ Planning an itinerary before travelling to a city is one of the most important travel preparation activities. Motivated by the needs of considering real-world traffic conditions, user preferences, and the travel time budget, we study the problem of personalized trip planning. The third main work in this thesis is that we propose a novel framework called TripPlanner, leveraging a combination of Location-based Social Network (i.e. LBSN) and taxi GPS digital footprints to achieve *personalized, interactive, and traffic-aware* trip planning. First, we construct a dynamic POI network by extracting relevant information from crowdsourced LBSN and taxi GPS trace data. Then, we propose a two-phase approach for personalized trip planning. In the route search phase, TripPlanner works interactively with users to generate candidate routes with specified venues; In the route augmentation phase, TripPlanner applies heuristic algorithms to add user's preferred venues iteratively to the candidate routes, with the objective of maximizing the route score and satisfying both the venue visiting time and total travel time constraints. To validate the efficiency and effectiveness of the proposed approach, extensive empirical studies are performed on two real-world data sets which contain more than 391,930 passenger delivery trips generated by 536 taxis in a month, and more than 110,200 check-ins left by over 15,680 Foursquare users in 6 months in San Francisco. This work is mainly related to the understanding of *traffic dynamics* about how much driving time is needed to transit between any two points in the city at different departure time of the day and day of the week.

1.3 Organization

The remaining chapters of the thesis are organized as follows, with their relationships shown in Figure 1.3. In Chapter 2, we survey the related work from the perspectives of three defined dynamics. Before presenting concrete work, we introduce some necessary preliminaries in Chapter 3, including the data preparation and representation. Then, we introduce our main work leveraging the taxi GPS trace data in details one by one in Chapter 4, 5 and 6 respectively; each concerns certain category of defined *social and community dynamics*, as shown in Figure 1.3. In more detail, in Chapter 4, we present our research on informing anomalous behaviours of taxi drivers in real time through mining their passenger-

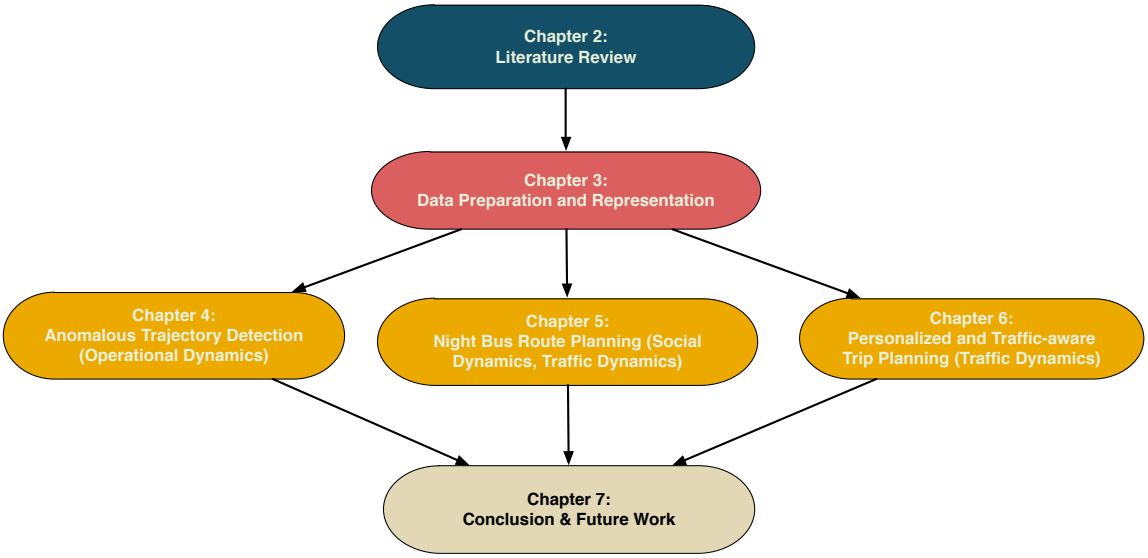


Figure 1.3: Organization of the rest of the thesis.

delivery trajectories, dealing with *operational dynamics* of taxi drivers; in Chapter 5, we introduce a greener and environmental-friendly transport to citizens at night by mining frequent taxi-passenger flows, dealing with *social and traffic dynamics*; in Chapter 6, we offer a personalized and traffic-aware trip planning system to suggest time-sensitive travel routes according to the user's preferences with the help of two heterogeneous crowsourced LBSN and taxi GPS digital footprints, dealing with *traffic dynamics*. Finally, we conclude the thesis and chart the future research directions in Chapter 7.

Chapter 2

Literature Review

Contents

In this chapter, we will review the existing research on mining taxi GPS traces in line with three categories that we have defined in Chapter 1. In each category, we first discuss common research directions, then enumerate some representative work for each direction. Finally, we make a statistical study to show the tendency of emerging research topics during recent years.

2.1 Social Dynamics

Common research directions in this category include: hotspot identification, measuring the linkage strength between areas, and discovering physical laws of human movement from the taxi GPS traces.

2.1.1 Hotspot Identification

The ability to identify the most frequented locations in a city can be useful for urban planning, public transportation route design, tourism agencies, public safety, etc. There are extensive work focusing on detecting significant places from GPS trajectories from personal devices (such as cell phones, the GPS localizers) [6, 26, 110]. Locations where a user has stayed for a minimum amount of time would be identify as his hotspots [166]. For our taxi GPS traces, the places of interests can be detected directly since we know with reasonable accuracy where passengers have been dropped off. We can also compare the importance of different places by simply counting the number of drop offs at different places. Moreover, more meaningful results can be uncovered if we add further contexts such as time of the day, season of the year, etc.

Chang et al. [143] proceed by first filtering trajectories using contextual information (weather, etc.), then clustering GPS points into areas and finally defining a hotness score for each area according to the number of taxi requests divided by the size of the area. Yue et al. [155, 156] use simple nearest-neighbour clustering to group taxi pick-up and drop-off points and discover attractive areas (i.e. hotspots) as well as the attractiveness amongst different areas. With a different definition of hotspots, Liu et al. [96] use vehicular speed information to quantify the "crowdedness" of an area, and define hotspots based on these crowdedness values. Yuan et al. [150, 151] define "landmarks" as the road segments most frequently traversed by taxi drivers, which in some sense, is also a kind of hotspots. In addition to extracting hotspots around the city, some work takes a step further. For example, Li et al. [87] propose a method for predicting the amount of pick-ups at each hotspot by using a variant of the Auto-Regressive Integrated Moving Average (ARIMA), which is a well-known prediction method in time-series analysis [21]. Similarly, Luis et al. [108] ensemble three well-known time-series forecasting techniques to predict the passenger-demands in a 30-min horizon in hotspots. Wang et al. [139] use passenger pick-up and drop-off points to analyze the location and travel patterns to and from hotspots. Based on the observation that the temporal variations in taxi pick-up and drop-off patterns in hotspots correlate well with their "land use" [101], Pan and Qi et al. [112, 119] uncover the different social functions of different regions of a city (i.e. commercial, residential, recreational, etc.). Similarly, Yuan et al. [148] uncover the functionality of different regions combining the taxi GPS traces and points of interest (POIs) of each region (e.g., restaurants, shopping malls.)

2.1.2 Measuring the Linkage Strength between Areas

There are many ways to characterize the "linkage strength" between two areas in a city for different purposes. The "linkage" can be measured by the human flow from original

area to the destination (i.e. OD flow), which can be used to examine the effectiveness of current public transportation networks. Strong "linkage" but with few public transportations between two areas may imply a new bus route is necessary. The "linkage" can be also measured by the driving distance. Two geographically-close areas may not be easily linkable due to the flawed road network, or physical barriers.

Zhang et al. [161] first estimate the OD flows among locations in a city relying on the taxi GPS traces, then mine the semantics of OD flows to understand the activity purposes. More interestingly, Peng et al. [116] decompose a city's OD flow into a linear combination of three types of trips: travel between residential and work areas, travel between work areas, and leisure trips for different purposes; they propose a method for finding these three coefficients, thereby producing a rough estimate of OD flow. Zheng et al. [165] discover inefficient connectivity between two regions by looking at actual versus expected distance required to travel between these two regions, as well as the expected speed and actual volume of traffic, to determine whether their level of connectivity satisfies the demand of travel between them. They evaluate their results using a taxi dataset in Beijing, and demonstrate that the flawed areas uncovered by their algorithm agree with a new subway line added in the same area at a later date.

2.1.3 Discovering Physical Laws of Human Movement

There has also been some work in characterizing the physical laws of human movement, by means of taxi trajectories. This type of work has its roots in biology where the movement of animals is studied. By appropriately and precisely modelling the human movement, we can synthesize large-scale human mobility traces for system scalability test, algorithm performance evaluation, amongst others.

It has been observed that the movement of many animals follow a Lévy flight model, which is a random walk that generalizes Brownian motion. It can be detected by verifying whether the jump length follows a power-law behaviour. However, Jiang et al. [70] previously showed that using taxi data in order to provide evidence of human mobility as a Lévy flight, is mainly due to the underlying street network. Chen et al. [35] study the distribution of travel time and distance of taxi trips and show that they can be approximated by a power law distribution; additionally, they also show that most trips are short in both time and distance. However, Liu et al. [100] study this problem on a large 7-day database of taxi GPS traces in Shanghai, and argue that trip distances do follow the power law distribution, but the direction distribution is not uniform. Liang et al. [88] find that the taxis' travelling displacements in urban areas tend to follow an exponential distribution instead of a power-law. Similarly, the travel time can also be well approximated by an exponential distribution. Veloso et al. [136] also argue that trip distance, duration, and income follow

Gamma and *Exponential* distributions.

2.2 Traffic Dynamics

Common research directions in this category include: trajectory mapping, traffic monitoring and forecasting, and traffic outlier detection.

2.2.1 Trajectory Mapping

As the traffic dynamics refer to the knowledge about important traffic indicators in the road network observed by taxis, a fundamental pre-processing step is to map the taxi trajectories to the road network on a digital map. However, in many cases, an updated digital map of the city is not readily available. Although OpenStreetMap[1] combines GPS traces, satellite images and hand-labelled information to produce a very rich digital map, it is often inaccurate and incomplete. Fortunately, considering the fact that taxis are moving in the road network and can cover the whole road network in a very short time, it provides researchers with rich data source to construct and update the digital map [18, 98, 123]. A good survey, which comprehensively overviews popular methods for inferring maps from large collections of opportunistically collected GPS traces automatically, can be found in [19].

Having had the digital map at hand, researchers are ready to proceed trajectory mapping, whose objective is to align a sequence of sampled taxi GPS positions with the road network on a digital map. The task is very challenging mainly due to the following two reasons: 1) the localization error of GPS sensors can be up to 10 meters, resulting in the GPS points often do not "sit on" the digital map; 2) to save the cost of data transmission, taxis often report their locations at a very low frequency (e.g., one point every 2-5 minutes), increasing the uncertainties (several alternative paths may exist) between two consecutive GPS points. To overcome the above-mentioned challenges, researchers often take certain "contextual information" into consideration, such as distance and orientation [29, 52, 84], spatial context and speed information [102], the influence of neighbouring GPS points [152].

Liu et al. [99] propose first pruning a set of trajectories by using speed and orientation, then cluster the remaining segments using distance and orientation, and finally use B-spline fitting [125] to fit the clustered traces onto road segments. Chawathe [32] proposes assigning a confidence score to different segments of the trajectory, and then proceeds to sequentially match the different segments, beginning with those with the highest confidence score. Rahmani et al. [121, 122] propose a two-step approach to tackle this problem: map-matching and path inference. The first step is to identify a set of candidate links in the

1. http://www.openstreetmap.org/

vicinity of each GPS point and find the (perpendicular) projection of the point on each link. The second step is to identify the most probable trajectory among all possible trajectories that pass through candidate links of a sequence of GPS points, such as the shortest path which connects a pair of two projected points on the digital map [29]. To deal with the data stream, Hunter et al. [66] introduce a *path inference filter* to map steaming GPS data in real-time. The filter is trained based on the new data without ground truth observations. The evaluation results on taxi data collected from different cities validate high performance and throughput of the proposed method.

2.2.2 Traffic Monitoring and Forecasting

Real-time and near-future traffic conditions monitoring/forecasting in the road network are vital in most of route planning problems. Although Google maps[2] have provided traffic services in many cities, the accuracy is far from promising [12,150,151] and only very limited indicators are offered (e.g., the driving speed bands from "slow" to "far"). Given that taxi drivers are continuously driving around the city, the real-time collected GPS traces are a natural source for monitoring the traffic conditions (e.g., the travel speed, the taxi density) in the road network. Furthermore, it has been observed that traffic generally follows a regular pattern throughout the day, and hence many researchers have used a vast array of different methods to forecast the traffic conditions in the near future, by leveraging the taxi GPS traces collected in history. We will list some representative work for traffic monitoring and forecasting, respectively.

2.2.2.1 Traffic Conditions Monitoring

The essence of traffic conditions monitoring is to estimate the traffic indicators in the *current* road networks. Gühnemann et al. [53] use GPS data to construct travel time and speed estimates for each road segment, which are in turn used to estimate emission levels in different parts of the city. Their estimates are obtained by simply averaging over the most recent GPS entries. Herring et al. [59] use *Coupled Hidden Markov Models* for estimating traffic conditions on arterial roads. They propose a sophisticated model based on traffic theory which yields good results. Based on the fact that traffic conditions tend to follow distinct patterns over the course of a week, Hofleitner et al. [61] from the same research team (i.e. the *Mobile Millennium* team at UC-Berkeley) learn historic traffic patterns from previous data which are used as prior information to estimate traffic conditions via a *Bayesian* update. Both of the work use sparsely sampled GPS probe vehicle data provided by a small percentage of vehicles. In modern cities, only few main roads have installed loop

2. http://maps.google.com

detectors. Aslam et al. [11] find that taxi volumes on the roads has a strong correlation with all traffic volumes, based on the data collected from loop detectors and taxis. Then, they build a model to infer the traffic volumes in road segments where do not have loop detectors installed, from the taxi GPS traces only. By monitoring the real-time traffic conditions, traffic jams can be detected. Schäfer et al. [124] demonstrate a visualization of traffic conditions around the city, which can be used to detect congested and blocked road segments by considering congested roads as those where the velocity is below 10 km/hr. More recently, Wang et al. [141] develop an interactive system for visual analysis of traffic congestion, and they detect the traffic jams based on the traffic speed on individual road segment automatically. They further build traffic jam propagation graphs to understand how traffic jams on each road segment influences the neighbouring road segments. To decrease possible false alarms for traffic jams, Giannotti et al. [51] detect traffic jams by searching for groups of cars close together that are all moving slowly. An interesting work is the analysis of traffic congestion changes around the Olympic games in Beijing based on location data collected by GPS-equipped taxis [142], although it is an *ex post facto* analysis of traffic conditions.

2.2.2.2 Traffic Conditions Forecasting

Balan et al. [12] predict the travel time and fee between two locations in Singapore by averaging the travel time of similar taxi trajectories in history (e.g. similar stating time, similar starting and ending locations). Lippi et al. [91] use Markov logic networks to perform relational learning for traffic forecasting on multiple simultaneous locations, and at different steps in the future. This work is also designed for dealing with a set of traffic sensors around the city. Su & Yu [129] use a Genetic Algorithm to select the parameters of a SVM, trained to predict short-term traffic conditions. Furtlehner et al. [45] from *INRIA* propose a traffic inference method based on the *Belief Propagation* algorithm. Based on the fact that traffic conditions on different links are highly correlated (both spatially and temporally), Han & Moutarde [55,56] demonstrate specific traffic patterns or traffic configurations over the *entire network* can be very informative and useful for long-term traffic modelling and forecasting. Yuan et al. [149] use both historical patterns and real-time sensory information to predict traffic conditions. However, the prediction they provide are between a set of "landmarks", and only the travel time between "landmarks" can be predicted. They define the "landmarks" as road segments which are traversed by taxi drivers frequently, so they are only a subset of the whole road network. Castro et al. [29] propose a method to construct a model of traffic density and automatically determine the capacity of each road segment using a large database of taxi GPS traces; by pairing these two pieces of information one can obtain accurate predictions of future

traffic conditions and potential traffic jams. Besides predicting the expected travel time between two points, Hunter et al. [67] propose an expectation maximization algorithm that simultaneously learns the likely paths taken by taxis as well as the travel time distributions through the network, mainly addressing the secondary road network. Later on, the same authors present a scalable algorithm for learning path travel time distributions on the entire road network [68]. Both algorithms are validated using a small sample of taxi GPS traces data collected over the Bay Area of San Francisco, CA.

2.2.3 Traffic Outlier Detection

Researchers have defined different traffic outliers for many objectives, such as event detection, traffic diagnose. A common outlier is defined as the abnormal traffic patterns between regions, by comparing the differences (using distance to measure, such as the Euclidean and Mahalanobis distance) to their spatial and/or temporal neighbours. For example, Pang et al. [113] propose to use an adaptation of likelihood ratio tests (a technique which has previously been mostly used in epidemiological studies) to describe traffic patterns and uncover unexpected traffic outliers. Liu et al. [97] also consider the traffic outliers as the unusual traffic patterns between regions. Intending to discover the relationships, especially causal interactions among detected traffic outliers, they further construct outlier causality trees based on temporal and spatial properties of outliers. They propose an algorithm to generate the frequent subtree from the outlier trees, which can potentially reveal underlying flows in the design of the existing road network. Taking a step further, Chawla et al. [33] infer the root cause of the outliers (i.e., the OD link which contributes the anomalies most) by solving an L_1 inverse problem.

Traffic outlier can be also defined in the temporal dimension only. As an example, Li et al. [86] utilizes agglomerated temporal information of the entire dataset as the basis for outlier detection. Some traffic outliers are also studied in the granularity of road segments for traffic diagnose. For instance, Pan et al. [111] identify road segment outliers according to drivers' routing behaviours on an urban road network. Traffic outliers are then described by mining representative terms from the user generated data in twitter mobile social network. Similarly, road outliers are detected and possible causes are diagnosed by integrating a number of data sources [40], such as taxi/bus GPS data, eventful data.

2.3 Operational Dynamics

Common research directions in this category include: Passenger/taxi finding, driving route planning and anomalous driving behaviours detection.

2.3.1 Passenger/Taxi Finding

Study of the taxi drivers' behaviours in finding new passengers is an intensive direction for a number of research groups. Most papers have focused on finding passenger-demand hotspots to direct the navigation for unoccupied drivers (or waiting passengers), thus work about identifying hotspots often serves as the preliminary procedure. Some papers discover the efficient/inefficient passenger-finding strategies to provide drivers with guidances to find new passengers after dropping off the last passengers for a given region and time slot. Additionally, a number of studies pay attention to aiding passengers to find vacant taxis, and the estimation of the waiting time for the vacant taxis at the waiting roads/corners.

2.3.1.1 Passenger-demand Hotspot Recommendation

Chang et al. [143] find demand hotspots by extracting the time and environmental contexts of a set of taxi requests, clustering these requests using k-means and agglomerative hierarchical clustering, and ranking these clusters for drivers to search new passengers. Palma et al. [110] use the speed of vehicles in a data set of trajectories to find "interesting places" by means of a density-based clustering algorithm. Considering the potential fuel cost by driving to a distant area, some studies focus on finding passengers *locally*. For example, Powell et al. [118] construct a spatio-temporal profitability map based on historical data to guide taxis to find new passengers on a *local basis*. Lee et al. [77] first use k-means clustering to split a road network into different areas, and then perform a temporal analysis to create a time-dependent pick-up pattern within each area. Their analysis suggests taxis should go to the nearest area with demand to pick up new customers. The simple approach is able to find clusters with highest demand. However, as Liu et al. [95] demonstrated, in order to maximize profit, a taxi driver may not necessarily want to base his choice solely on demand [95]. A balance between profit maximization and demand coverage is necessary. Considering the fact that taxi drivers may fail to pick up new passengers at the first suggested locations, some papers intend to recommend a sequence of locations (i.e., passenger-finding routes) to follow to find new passengers successfully. For instance, Ge et al. [48] develop a mobile recommendation system, which first clusters the pick-up points of the top drivers, then recommends a sequence of these pick-up points for other drivers to find new passengers. Hu et al. [63] extend this idea by creating a pick-up tree with the pick-up points with highest probability; the authors argue that this method is more suitable for situations where you have a set of vacant taxis (as opposed to a single one) in the same area (i.e. the competition from other drivers). Yuan et al. [153, 154] present *T-Finder*, which automatically extracts "waiting areas" for taxis based on the distance between consecutive GPS points. The authors then compute the probability of picking

up a passenger based on the current time and the road segment or waiting area. This information is used to provide a recommendation system for drivers and passengers. More recently, Ding et al. [43] present a system, called HUNTS to find a connected trajectory of high profit and high probability to pick up a passenger within a given time period in real-time, by exploiting heuristic algorithms. Different from all mentioned work using taxi GPS traces in history or real time, Takayama et al. [132] perform an empirical study which solely rely on survey results from the drivers to propose promising "waiting/cruising" locations to taxi drivers. However, their method is based on surveys given to drivers, which is inefficient to obtain data, is sensitive to human error, and is also difficult to continue indefinitely.

2.3.1.2 Uncovering Passenger-finding Strategies

Through extensive statistical analysis, Liu et al. [95] uncover the good driving behaviours in both choosing effective passenger-finding and passenger-delivery strategies. Experimental results reveal that most top taxi drivers (ranked according to their revenues generated) choose similar spatio-temporal areas. The authors discovered the somewhat surprising facts that top drivers strived to drop off passengers as quickly as possible in order to serve as many passengers as possible; additionally, they choose to operate in areas *other* than the Central Business District. Similarly, Li et al. [80] propose an analytical model, intending to discover the efficient/inefficient passenger-finding strategies, and the efficient strategies are used to be the guidances for drivers to increase income. Specifically, they categorize the observed passenger-finding strategies based on time, location, whether they are "hunting" or "waiting", and whether the driver remains in a local area or travels a longer distance to find a new passenger. The authors then use a form of Support Vector Machine (SVM), L1-Norm SVM [17] to determine, based on the current time and location, whether the driver should *hunt locally, waiting locally, or going distant (i.e. travelling to a distant location)*. Yamamoto et al. [144] provide routing strategies for multiple taxis using fuzzy clustering mechanisms. Hu et al. [64] formulate taxi driver's task of hunting for new passengers as a decision problem at each intersection and propose solving it using probabilistic dynamic programming. Nevertheless, it is unclear whether a concrete "micro-strategy" for finding passengers can be extracted by mining past taxi trajectories: the strategies employed by top drivers would have to be fairly *consistent* or *predictable*. As shown by the study in [135], Veloso et al. perform a predictability analysis of the next pick-up area given drop-off features. Their results show there is only a 54% predictability rate, suggesting hunting/cruising trips are largely random.

2.3.1.3 Vacant Taxi Finding

In addition to aiding taxis finding new passengers, some studies develop algorithms to help passengers find vacant taxis quickly. For example, Phithakkitnukoon et al. [117] use a grid decomposition and a naive Bayesian classifier to predict vacant taxis in different areas. Zheng et al. [164] model the probability of taxis leaving their current road segment as a Non-homogeneous Poisson Process, and use this model to estimate the waiting time for taxis at different locations and at different times; these estimates are then used to provide a recommender system for people searching for taxis. More recently, Qi et al. [120] present a method to predict the waiting time for a passenger at a given time and spot, where the arrival model of passengers and vacant taxis are built from the events that taxis arrive at and leave a spot. The passenger waiting queue in a spot can be simulated and the waiting time can be inferred with the models.

2.3.2 Route Planning

Users are often experiencing route planning problems when visiting cities, especially when visiting unfamiliar cities. The generalized routing problems in transportation networks has been studied extensively for (at least) four decades, dealing with different objectives and constraints. Popular objectives include shortest route, shortest travel time, lowest operation cost, maximum passenger flow, maximum area coverage and maximum service quality while the constraints include time, capacity and resources. Popular techniques that have often been used are dynamic programming [38], variations of Djikstra's algorithm [42], and variations of the A^* algorithm [71]. A recent released technical report which surveys recent advances in algorithms for route planning in transportation networks can be found in [14]. Note that the research of route planning is also a common topic in other networks, such as wireless sensor networks [5], mobile social networks and delay tolerance networks [39, 65].

Some work have explored the excellent knowledge of taxis about the city's road network to suggest driving directions. For example, by observing taxi drivers' behaviours, Yuan et al. [149, 151] combine historical traffic patterns to compute shortest-time driving routes. They first identify "landmarks" which are traversed by taxis frequently, then construct a time-dependent landmark graph based on a large set of taxi trajectories. The routing algorithm first finds a rough route on the landmark graph, and then this is refined to a route on the underlying road network. By estimating travel time distributions, the authors allow travel times to behave stochastically, which may yield more accurate representations. Their results are validated by the in-the-field testing of real drivers. Similarly, Li et al. [82, 83] construct a hierarchy of roads based on frequency of use, and perform planning from a

source to a target by trying to travel through the highest hierarchy roads.

With a quite different objective, Bastani et al. [15] propose defining new transportation routes by mining through and combining multiple taxi trajectories. The authors suggest these new routes could be used by a mini-shuttle transportation system that lies somewhere between taxis and buses. More recently, Ma et al. [106] propose a scheduling algorithm to plan ride-sharing routes for taxi drivers, and their optimal objective is to minimize the additional incurred travel distance. With a similar objective, Zhang et al. [159] present *coRide* which has three components, a dispatching cloud server, passenger client, and an on-board customized device to plan cost-efficient carpool routes for taxi drivers and thus lower fares for the individual passengers.

2.3.3 Anomalous Driving Behaviours Detection

The objective is to detect the anomalous driving behaviours through mining passenger-delivery trajectories. In another word, this "abnormality" is defined in the "individual" level, which is different from the *traffic outlier* we have discussed, which is often a result of the collective behaviours (e.g. traffic jams, big sport events). By collecting the trajectories from many taxis, we may be able to automatically identify not only these "normal" trajectories, but also "anomalous" trajectories. An anomalous trajectory can be caused by external factors such as accidents or the closure of a main road, and may also be caused by fraudulent drivers trying to charge more money from passengers. The ability to automatically detect anomalous trajectories can thus enable to prevent drivers to take advantage of passengers unfamiliar with the city.

Liao et al. [89] use conditional random fields to label anomalous taxis, coupled with an active learning scenario, where human interaction can help guide the learning. Balan et al. [12] reported trajectories with extremely long travelling distances as anomalous: any trajectory with a distance twice as long as the straight line distance between the start and end positions, or any trajectory with an average speed lower than 20 km/hr or higher than 100 km/hr. Zhang et al. [158] propose *iBAT*, a method based on isolation trees and a grid decomposition, to solve this problem. The authors maintain a set of historical trajectories and determine whether new trajectories are *isolated* from this set by randomly selecting grid cells from the new trajectory and determining how many of the historical trajectories also contain this grid cell. Since the method is based on sampling, the process must be repeated a number of times for each trajectory in order to obtain an *anomaly score* that indicates the degree of anomalousness of the new trajectory. Through the use of a testing set of manually labelled trajectories, the authors verified the accuracy of their proposed method. More recently, Ge et al. [46, 47] proposed a similar method for detecting taxi fraud. Their method uses a grid decomposition and complete trajectories

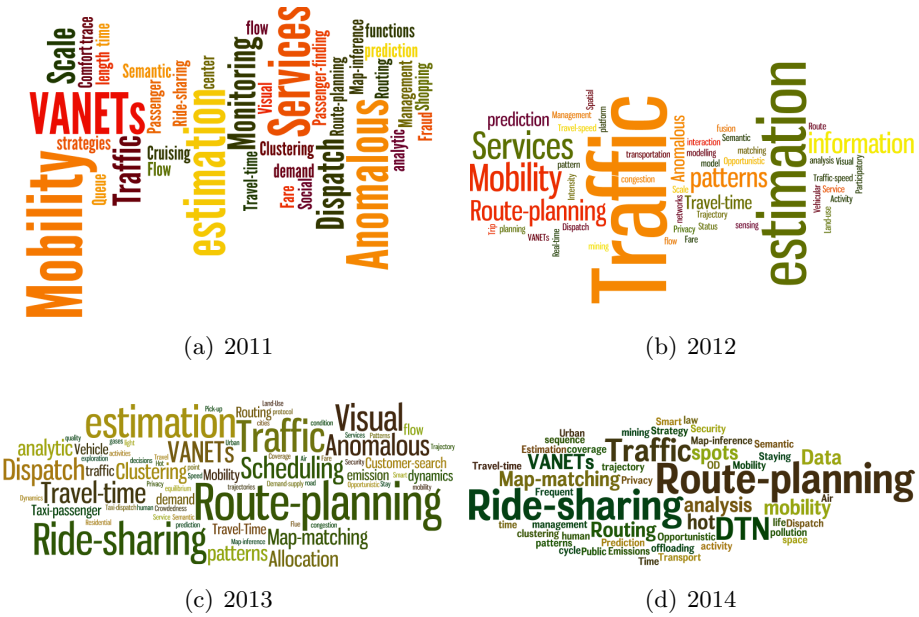
in a similar way as done in [158]. They compute two pieces of evidence for detecting anomalous trajectories. The first involves computing the independent components (using Independent Component Analysis) of a set of trajectories, and compute the *coding cost* (which is essentially the entropy) of a trajectory's independent components. The second is a method for determining the expected distance for the most common routes, and compute how much a trajectory's distance differs from the norm. These two pieces of evidence are combined using Dempster-Schafer theory. Although their experimental results fail to convince the reader that their method provides an advantage over standard density-based methods, they provide mechanisms for differentiating between malicious detours and detours due to traffic interruptions or poor knowledge of the area. Despite the high accuracy and solid evaluations, both methods presented in [46, 47, 158] suffer from a number of shortcomings. The most important one is that they only work with completed trajectories, disqualifying it from being used for real-time fraud detection.

2.4 A Statistical Study

We make a statistical study to understand the tendency of research topics during recent years. Specifically, we first group the related papers according to its published year, and then extract the research topics from keywords. Note that we only keep keywords that the studied paper focuses on most, since we are often experienced that one paper may deal with several subjects. As an example, consider Castro's paper on the traffic conditions prediction [29]. The extracted topic should be "traffic conditions prediction", though their method requires the trajectory matching techniques. Moreover, we also unify different descriptions of the same topic manually. For instance, some papers may use "path planning" as the keywords while others may use "route planning". We intentionally use "anomalous" to describe *anomalous driving behaviours* and *traffic outliers* since both of them are dealing with certain aspects of abnormality. Finally, the extracted topics are visualized by "word clouds" shown in Figure 2.1, which is generated by Wordle[3].

As can be seen from Figure 2.1(a), in 2011, the hottest research topic is about "mobility", which aims to understand the spatial-temporal mobility patterns of the whole population in the city underlying by the taxi GPS traces. Also the research about "anomalous" has received wide attention during that year, attempting to detect the anomalous driving behaviours, and abnormal traffic patterns in the road network (a.k.a. traffic outliers). During 2012, most of attention has been shifted to the research on "traffic estimation", as shown in Figure 2.1(b); most papers focus on discussing and developing algorithms to accurately estimate the congestion level in each road segment as well as the travel time be-

3. http://www.wordle.net



(a) 2011 (b) 2012

(c) 2013 (d) 2014

Figure 2.1: Word clouds generated by keywords from literatures during the recent 4 years. Keywords with bigger size refer to more popular studied topics.

tween two points in a city. Research on "traffic estimation" continues to be a popular topic in the following two years (i.e. 2013 and 2014). However, topics about "route planning" and "rider-sharing" are growing significantly in 2013 and 2014[4], as shown in Figure 2.1(c) and (d). Route-planning can refer to the work leveraging the excellent knowledge about the city of taxi drivers to recommend driving directions [82, 149], and can also refer to the work related to public transportation routes planning based on the traffic flow reflected by the taxi GPS traces [15]. Taxi ride-sharing is a promising new research direction that can have significant social and environmental impact. Given the rapid growth of cities and use of motorized vehicles, taxi ride-sharing provides a promising mechanism to mitigate increasing road congestion.

4. Results in 2014 are obtained by the statistics of papers appeared in the first five months, just before the accomplishment of this manuscript.

Chapter 3

Data Preparation and Representation

Contents

In this chapter, we will discuss some issues related to the data preparation and representation.

3.1 Data Preparation

Data preparation is the pre-procedure and reliable results can be guaranteed only if data cleaning is provided for many taxi data mining tasks [162]. Here, we will first introduce the data format, followed by the overview of possible data problems for the taxi GPS data.

3.1.1 Data Format

We get a large-scale real-world taxi GPS data set of more than 7,000 taxis served in a large city in China (Hangzhou) for one year (April, 2009∼March, 2010). Hangzhou has a population of more than 6 million people and it is also a famous tourism city. The large population and massive passenger flows raised great challenges and opportunities to taxi drivers. The sampling frequency for this data set is 1∼7 times per minute.

Another taxi GPS data set used in our research is freely available online[1]. This taxi data was generated by 536 taxis in June 2008 in San Francisco, CA. Note that the taxi GPS traces in this city can be also gathered in real time, with the provision of the public API[2]. The sampling frequency for this data set is around once per minute.

Table 3.1 lists the fields for each GPS record for the taxi GPS data set in Hangzhou city, along with a sample of the GPS entry. The "bearing" information (measured by the angle between the taxi heading direction and the north direction) refers to the heading orientations of taxis at the sampling time. Note that the "bearing" information is not provided for the taxi GPS data set in San Francisco city.

Table 3.1: Fields for a GPS entry with a sample

Taxi ID	Longitude	Latitude	Speed (km/h)	Bearing (°)	Occupied flag	Year	Month	Day	Hour	Minute	Second
10429	120.214134	30.212818	70.38	240.00	1	2010	2	7	17	40	46

Data provided do not contain the pick-up/drop-off information directly. But we can easily extract the pick-up/drop-off points based on the taxi status (i.e. the occupied flag in Table 3.1) and the driving speed. Specifically, when the taxi status is '1', it means that the taxi is occupied; otherwise, the taxi is empty. The pick-up point is then identified when the taxi status changes from '0' to '1', and the speed increases from zero. Similarly, the drop-off point is inferred when the taxi status changes from '1' to '0', and the speed decreases to zero. Consequently, trace data can be separated into passenger-delivery and passenger-finding trajectories: trajectory starting from the pick-up point to drop-off point is corresponding to the passenger-delivery trajectory; while the one starting from the last drop-off point to the new pick-up point is corresponding to the passenger-finding trajectory.

3.1.2 Data Problems

Identification of possible data problems is essential for the data cleaning process. Here, we overview the possible data problems with the hope of providing an insight into the type of problems that researchers should be aware of in order to reduce "tainted" results, though it is far from being a complete list.

- *Missing data*

Due to the occlusions of buildings, poor GPS single in some locations (e.g., the tunnel, the underground parking) or GPS device errors, the GPS data cannot be received occasionally, resulting in a lapse of several minutes or even hours between two nearby entries. In other words, a big physical jump with no information about the taxi's movement throughout this time duration. Depending on the problem being addressed, this data can either

1. http://cabspotting.org/
2. http://cabspotting.org/api

be left as it is, the trajectory can be split into two (or more) parts, or the trajectory can be truncated at the point before the jump occurred.

■ *Erroneous data*

Due to the GPS device errors, certain GPS entries may contain erroneous data, such as erroneous latitude/longitude or time entries. Most of the time, these are isolated (i.e. with abnormal far distance from nearby entries) and can be easily identified by using contextual information from the surrounding entries. A simple and common way to overcome these erroneous entries is to extrapolate from the surrounding entries.

■ *Occupied flag improperly set/detected*

The state of the occupied flag may not be properly set, partly because the taxi driver does not set his indicator properly, partly because there is a fault in the device. This may result in certain taxis being continuously occupied or vacant (e.g. last for an extremely unreasonable long time). When attempting to extract information from occupied/vacant trips, this type of problem can play negative impact at the obtained statistics. One possible solution is to compute the proportion of time the taxi is occupied/vacant, and discard any taxis that have extreme values, such as being occupied over a certain threshold ratio of time (e.g., 90%).

A relevant but more difficult problem is due to the low sampling rate of the GPS device (i.e., low time resolution). Specifically, we may not be able to determine when one trip ended and when the other began because of the rate at which GPS entries are received. This can be observed in popular transport areas, such as the airport: a taxi dropping off a passenger at the airport may find a new passenger immediately.

■ *Multiple drivers*

GPS data provided does contain the ID information of drivers for each taxi. However, we find that a taxi is commonly operated by more than one driver, through our interview with some taxi drivers in Hangzhou, China. The problem of the determination of which driver is currently active is very challenging and cannot be inferred from the taxi GPS data directly. So far, no solutions are reported [28]. A possible approach might be to first detect the areas where the drivers always visited at the same time, and then identify the true place where the drivers take shift handover, given the fact that the taxi drivers roughly take handover at the same time and places (e.g., gas re-fuelling stations are preferable for most of drivers) based on their agreements. However, it is often difficult to verify since we do not have the ground truth of which driver is active.

■ *"Sleeping" taxis*

Although having multiple drivers allows taxis to operate at all hours of the day, some night-duty drivers may stop to sleep at certain points for some time. The case is even common for single-driver taxis. Thus, it is important to differentiate between a sleeping

taxi and a taxi that is waiting for a passenger. In a similar manner to what was proposed above, one could begin by detecting areas where the taxi is always parked at the same time during the night.

3.2 Data Representation

Definition 3.1. A GPS point is formally defined as a location where the taxi is at the sampling time. It can be represented by a triplet $p_i = \langle x_i, y_i, t_i \rangle$, where $x_i, y_i \in \mathbb{R}$ refer to the physical location (i.e., longitude and latitude). A pick-up point is a special GPS point indicating when and where pick-up event occurs; a drop-off point is a special GPS point indicating when and where drop-off event occurs.

Having defined the GPS point, we are ready to define the passenger-finding and passenger-delivery trajectories, respectively.

Definition 3.2. A passenger-delivery trajectory is composed of a sequence of GPS points, in which the first point is the pick-up point and the last point is the drop-off point. On the contrary, a passenger-finding trajectory is composed of a sequence of GPS points, starting from the last drop-off point and ending at the next pick-up point.

How to represent a taxi trajectory (i.e., the passenger-finding or the passenger-delivery trajectory) is the preliminary, and often depends on the concrete problems and applications. Here, we will list some popular taxi trajectory representation methods, as follows.

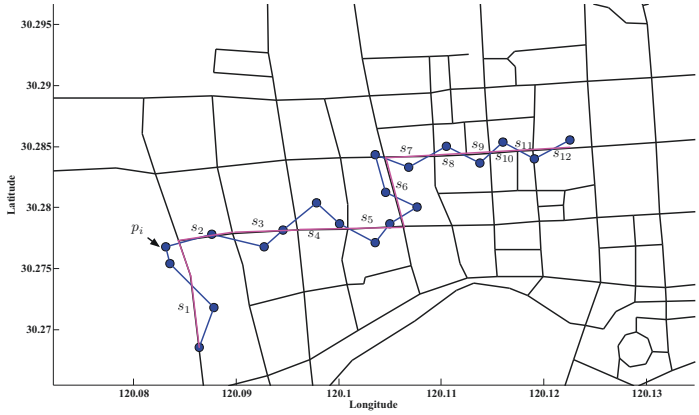
■ *A sequence of GPS points*

The most intuitive way is to represent the trajectory as a sequence of GPS points (i.e. $t = \langle p_1, p_2, \cdots, p_n \rangle$), according to the definition directly. However, it suffers from several drawbacks.

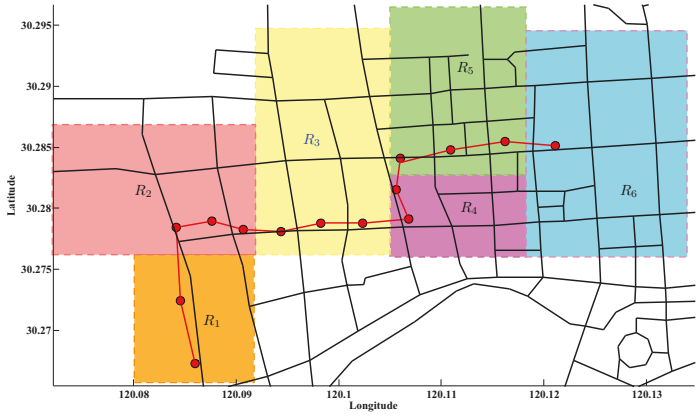
♢ A unique taxi trajectory may have different representations, due to the non-uniform sampling rate. Consider the two illustrative trajectories in Figure 3.1: trajectories t_A and t_B are unique, which are generated by taxis at the same time, from the same origin to the same destination, also following the same roads as well. The only difference is the GPS sampling rate; there are more sampling points for t_A. Obviously, the similarity of these two trajectories is small if comparing them based on the representation, which is not true.

♢ The trajectory is very difficult to understand, since its representation has very little semantic meaning.

To overcome the drawbacks, researches have developed many methods to map the raw trajectory (i.e. a sequence of GPS points) to road networks, regions in order to get a better representation, detailed as follows.



(a) t_A. The trajectory in purple is the mapped trajectory on the digital map.



(b) t_B. Different regions are denoted with different colors.

Figure 3.1: Illustration of two trajectories. The marks denote the sampling points.

■ *A sequence of road segment IDs*

The trajectory can be represented by a sequence of road segment IDs. Additionally, an entering and a departure time stamp indicating the time when the taxi enters and departures for each road segment can also be integrated if necessary. By simply differentiating the departure time and the entering time, the stay time in each road segment can also be inferred. For instance, the trajectory shown in Figure 3.1(a) can be represented as $t_A = \langle s_1, s_2, s_3, \cdots, s_{12} \rangle$, where s_i refers to the road segment IDs. The main challenge for this representation is to develop robust trajectory mapping algorithms to map the GPS point to the road segment correctly, which has been discussed extensively in Section 2.2.1 of Chapter 2.

■ *A sequence of region IDs*

A taxi is moving inside a region or across different regions in the city, thus it is also a natural way to represent the taxi trajectory as a sequence of city regions. Similarly, region ID can also be associated with time information, showing when the taxi enters, departures for that region. For instance, the trajectory shown in Figure 3.1(b) can be represented as $t_B = \langle R_1, R_2, R_3, R_4, R_5, R_6 \rangle$, where R_i refers to the region IDs. The city is usually divided into different regions in advance, usually independent of the taxi GPS trajectories. Regions can either be obtained by dividing the city based on ZIP codes, the main city roads, and dis-jointed grid cells, or obtained by clustering other spatial data sources (e.g., Foursquare data, POI data). In a general sense, the union of divided regions can be a subset of the whole city (as illustrated in Figure 3.1(b)). Mathematically, $C - \cup_{i=1}^{n} R_i \neq \emptyset$, where C denotes the whole city.

For the last two representations, road segment ID and region ID can be enriched with semantic meanings to understand the trajectory deeper from the third party (a.k.a semantic annotation/labelling), such as Google maps, POI data [145]. A good survey, which overviews the state-of-art of semantic trajectory modelling and analysis, can be found in [114]. In our research, the taxi trajectory is represented by a sequence of region IDs: the whole city is divided into small-equal sized grid cells (regions) in Chapter 4; the regions are clustered by the spatial data provided by the Foursquare check-in data in Chapter 6. We will introduce the technical details about the taxi trajectory representation separately in the corresponding chapters.

Chapter 4

iBOAT: On-line Anomalous Trajectory Detection

Contents

4.1 Introduction

Recent years have witnessed an increasing interest in automatically detecting anomalous trajectories [24, 50, 78]. Although several aspects of abnormality have been used for

automatic detection by previous works, few of them have analyzed them with respect to the practical applications which they may serve. In this chapter, we mainly concern the *operational dynamics* when taxi drivers are delivering passengers, and we would like to use two potential applications to motivate.

Application I. Many passengers are victims of fraud caused by greedy taxi drivers who overcharge passengers by deliberately taking unnecessary detours [1]. The detection of these fraudulent behaviours is essential to ensure a high quality taxi service. Currently, these frauds are detected by manual inspection from experienced staff, based on complaints from passengers. This is rather costly and not effective enough. More seriously, most frauds are not even noticed by passengers if they are unfamiliar with the city. Given that anomalous traces usually deviate significantly from "normal" ones, it is possible to automatically detect them by comparing against a large collection of historical trajectories.

Another anomalous situation could occur when there are abnormal traffic conditions such as traffic accidents, resulting in certain road segments being blocked, forcing taxi drivers to find alternative routes.

Application II. Urban road networks undergo changes regularly, and these changes must be reflected in digital maps. These changes not only refer to newly installed roads, but roads permanently or temporarily closed. Performing these updates manually can be expensive, time-consuming, and would be "lagging" behind the occurrence of the actual changes. Taxis equipped with GPS devices can be viewed as moving sensors probing the real-time information about urban road networks, and can thus provide us with accurate and up-to-date information about changes in the road network.

In order to support the applications effectively, a successful anomaly detection method should posses the following characteristics.

1. **Accurate classifications**: This implies that the method should have a high detection accuracy while with low false-alarm rate.
2. **Sub-trajectory specificity**: In addition to labelling trajectories as anomalous, it can inform which parts, or *sub-trajectories* are responsible for the trajectory's anomalousness.
3. **Real-time response**: Detect anomalous trajectories in real-time. Alerts can be provided once anomaly is detected while the trip is still on-going.
4. **Characterizing the anomaly degree**: Provide a score quantifying the degree of anomalousness for each trajectory. This score can be used to rank a collection of trajectories.

A set of trajectories are considered "normal" with respect to a particular travel itinerary (i.e. from a specified point to another). We must then specify source (S) and destination

(D) areas, and consider only those trajectories travelling from S to D.

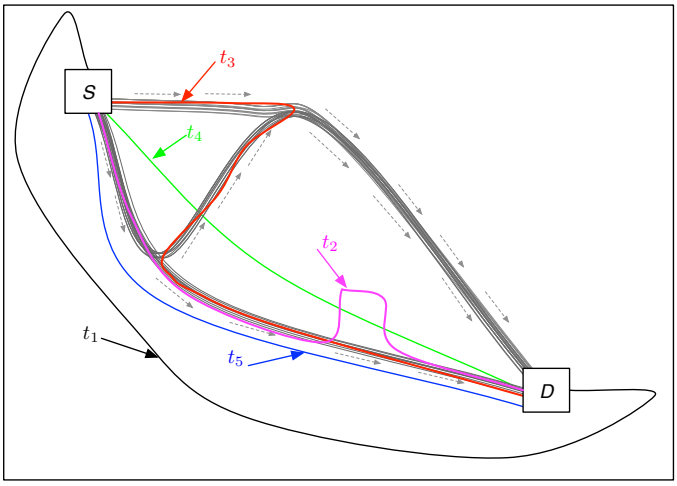


Figure 4.1: Example taxi trajectories between S and D.

Consider the three groups of "normal" trajectories between S and D, along with five anomalous trajectories (t_1 through t_5), displayed in Figure 4.1. The anomalous trajectories are labelled so because they are *infrequent and different* from the majority of other trajectories. Not only should trajectories that follow a completely different route (t_1, t_4, t_5) be considered anomalous, but also those that detour for part of the trajectory (t_2, t_3). The anomalous trajectories can be long detours made by greedy taxi drivers (t_1 and t_2 in Figure 4.1), or they can be short-cuts or new routes taken by experienced drivers (t_4 and t_5 in Figure 4.1). Detecting these anomalous trajectories is no trivial task due to the following challenging issues.

- ♢ First, as can be seen in Figure 4.1, there may be many different normal routes between S and D, and these clusters are usually with different densities and separated from each other. Traditional anomaly detection techniques [24, 50, 78], which are based on differences in distance or density, may be difficult to identify all the anomalies.
- ♢ Second, multiple normal routes also mean different driving distances. If we model driving distance, it is not able to discover those anomalies whose driving distance is close to that of the normal trajectories (like t_3 and t_5).
- ♢ Third, anomalous trajectories can be diverse. Like t_1, t_2, t_3, t_4 and t_5 in Figure 4.1, they are regarded to be anomalous due to quite different reasons. Then, it is not straightforward to characterize them with a single method.
- ♢ Finally, the concept of anomalous trajectory might drift over time, because the road network may change (i.e., newly-built or blocked roads). Hence, it is important to be

able to capture these changes and incorporate them into the model. Moreover, GPS traces often suffer from the low-sampling-rate problem since GPS devices usually send data at a *low and changing* frequency.

In this chapter, we aim to propose a novel anomalous trajectory detection method which addresses the four challenges above. Firstly, we extract valid taxi rides from all the taxi GPS traces, divide the city map into grid-cells of equal size, group all the taxi rides crossing the same source destination cell-pair, and augment and represent each taxi trajectory in each source-destination pair as an ordered sequence of traversed cell symbols (i.e., the taxi trajectory is represented as the sequence of grid cell IDs as discussed in Chapter 3.2. The technical details can be found in Section 4.3.). In such a way, the problem of anomalous trajectory detection is converted to that of finding anomalous trajectories from all the trajectories with the same source-destination cell pair. Secondly, for all the taxi trajectories between a certain source-destination cell-pair, we define those trajectories that are "few" and "different" from the normal trajectory clusters as anomalies. We then propose an Isolation-Based On-line Anomalous Trajectory (*iBOAT*) detection method which exploits the property that anomalies are susceptible to a mechanism called isolation [94]. Thirdly, we perform an empirical evaluation comparing *iBOAT* and other state-of-the-art methods with real-world taxi GPS data. Finally, we show how *iBOAT* can be used to effectively support real-world applications. In summary, the main contributions of this work include:

1. We present an Isolation-Based On-line Anomalous Trajectory (*iBOAT*) detection method that successfully addresses all the challenges mentioned above while still possessing the four characteristics mentioned above. (See Section 4.4 for details.)
2. We evaluate *iBOAT* with real-world GPS traces collected from 7,600 taxis for one month. We demonstrate the remarkable accuracy of our method, its ability to identify which *sub*-trajectories are anomalous, and its low computational cost. We also show that *iBOAT* outperforms the state-of-the-art anomalous trajectory detection methods. (Refer to Section 4.5 for details.)
3. After detecting the anomalous trajectories, we perform an analysis revealing that most of the anomalous trips are the result of conscious decisions of greedy taxi drivers to commit fraud. Also, we further provide evidence to deny possible excuses that some cunning drivers may use. We further discuss the ability of *iBOAT* in detecting road network changes. (See details in Section 4.6)

4.2 Related Work

In Chapter 2, we have extensively surveyed the work about mining taxi GPS traces mainly from the perspective of research topics. The research topics about "traffic outliers"

and "anomalous driving behaviours" are most relevant, each addressing certain aspects of abnormality. Here, we would like to review the related work from the perspective of the proposed algorithms.

In the literature, some solutions of anomalous trajectory detection have already been reported, For instance, Lee et al. [78] split a trajectory into various partitions (at equal intervals) and a hybrid of distance and density based approaches was used to classify each partition as anomalous or not; however, as we previously mentioned, solely using distance and density can fail to correctly classify some trajectories as anomalous. Bu et al. [24] presented an outlier detection framework for monitoring outliers over continuous trajectory streams, whose key idea was to build local clusters upon trajectory streams and detect outliers by a cluster join mechanism; Ge et al. [50] studied a similar problem of detecting evolving trajectory outliers, and they computed the anomaly score based on evolving direction and density of trajectories. Somewhat related, but addressing a different problem, Li et al. [86] identified outlier road segments by detecting drastic changes between current data and historical trends. Their approach detected what can be labeled as *global* anomalous events: they were events that affect many taxis; thus, their method would not be able to detect anomalous behaviours on an *individual* level. Balan et al. [12] reported trajectories with extremely long travelling distances as anomalous; as we previously mentioned, this rather simplistic approach may fail to detect other types of anomalous behaviours. Ge et al. [47] identified fraudulent taxi trajectories by using a model combining two forms of evidence: distance and density characteristics. Specifically, they first computed the independent components (using Independent Component Analysis) of a set of trajectories, and computed the *coding cost* (which is essentially the entropy) of a trajectory's independent components; once this was done, they determined the expected distance for the most common routes, and computed how much a trajectory's distance differs from the norm. These two pieces of evidence were combined using Dempster-Schafer theory. Finally, some recent work has used learning methods to identify anomalous trajectories [4, 85, 89, 127]. However, these last methods usually required training data which is expensive to label. After reviewing existing works on anomalous trajectory detection, it is not difficult to find that we are investigating a different problem from previous ones. That is, given all the taxi trajectories between a certain source and destination pair, our objective is to discover those few which take very *different* routes from the majority.

Most of these methods identify anomalous trajectories based on their physical distance to "normal" clusters or their orientations [22, 57]. Based on the idea of isolating anomalies [94], our previous work [34, 158] proposed a method which identifies trajectories as anomalous when they follow paths that are rare with respect to historical trajectories. The work in this chapter builds on them, but differs from them in the following respects: 1)

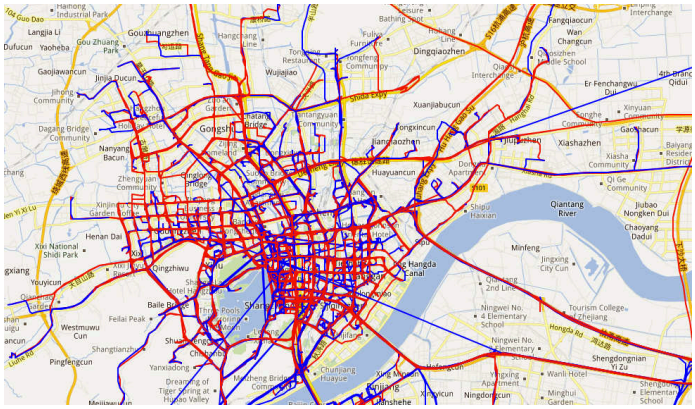


Figure 4.2: Traces of a taxi in Hangzhou city during a month, where red or blue indicates the taxi is occupied or vacant.

We introduce a novel anomaly scoring method which considers both the anomalous sub-trajectory and the number of trajectories "supporting" it. Anomalous trajectories with longer anomalous sub-trajectory and less support would be ranked higher; 2) The effect of the anomaly threshold and the size of the set of historical trajectories on the detection performance are investigated. This study allows developers to trade-off between the detection accuracy and the cost (i.e. computation time, memory); 3) Motivations behind the anomalous behaviours are analysed. Different applications corresponding to this motivation could be developed leveraging the proposed *iBOAT* method.

4.3 Preliminaries and Problem Statement

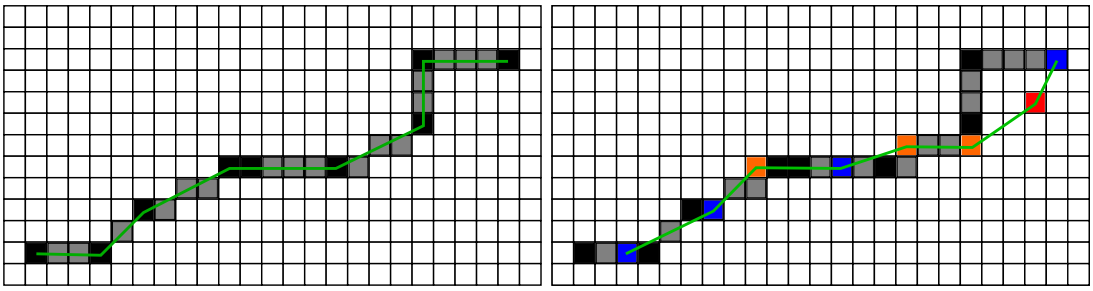
A taxi's GPS trace consists of a sequence of time-stamped GPS points (i.e., latitude/longitude, the estimated speed, vacant/occupied state) generated by a GPS device. Our dataset for this study consists of the GPS trajectories for 7,600 taxis in Hangzhou, China, where each GPS record is received at a rate of around once per minute. Figure 4.2 shows the trajectories for one taxi during a month; the red lines indicate when the taxi is occupied, while the blue lines indicate when it is vacant. In this work, we will only use occupied trajectories, since fraud detection is one of the motivations for this study, and fraud can only be committed with a passenger.

Definition 4.1. A **trajectory** t consists of a sequence of points $\langle p_1, p_2 \ldots, p_n \rangle$, which has been defined in Chapter 3.2. We will use t_i to reference position i in t, and for any $1 \leq i < j \leq n$, $t_{i \to j}$ denotes the **sub-trajectory** $\langle p_i, \ldots, p_j \rangle$.

The points p_i exist in a continuous domain, so dealing with them directly is difficult. In order to mitigate this problem, we assume we have access to a finite decomposition of

the area of interest. Specifically, we decompose the city area into a matrix G of grid cells, and we define $\rho : \mathbb{R}^2 \to G$ as a function that maps locations to grid cells. The criteria for choosing the grid cell size is to ensure the accuracy of the anomalous trajectory detection while maximizing the grid cell size. We experimented with different grid cell sizes and found that $250m \times 250m$ is the biggest grid size with the set detection accuracy.

Definition 4.2. A **mapped trajectory** $\bar{t}$, obtained from a trajectory t, consists of a sequence of cells $\langle g_1, g_2 \ldots, g_n \rangle$, where for all $1 \leq i \leq n$, $g_i \in G$ and $\bar{t}_i = \rho(t_i)$. We will write $g \in \bar{t}$ when $\bar{t}_i = g$ for some $1 \leq i \leq n$.



(a) (b)

Figure 4.3: An example of a trajectory with augmented cells (a); Comparing existing trajectory with a new trajectory (b).

Henceforth we will only deal with mapped trajectories, so we will drop the *mapped* qualifier. Because of the rate at which GPS entries are received and the small size of our grid cells, the mapped points (black squares in Figure 4.3) may not be adjacent, thereby leaving gaps. We **augment** all the trajectories to ensure that there are no gaps in the trajectories by (roughly) following the line segment (green line in left panel) between the two cells in question and "coloring" the cells underneath (gray cells in figure). Whereas the original trajectory consisted only of the black grids in Figure 4.3, the augmented trajectory consists of both the black and gray grids.

Let $\mathbb{T}$ denote the set of all *mapped and augmented* trajectories. Define the function $pos : \mathbb{T} \times G \to \mathbb{N}^+$, given a trajectory t and element g returns the first index in t that is equal to g:

$$pos(t, g) = \begin{cases} \arg\min_{i \in \mathbb{N}^+} \{t_i = g\} & \text{if } g \in t \\ \infty & \text{otherwise} \end{cases} \tag{4.1}$$

For example, if $t = \langle g_1, g_2, g_3, g_5, g_3, g_8 \rangle$, then $pos(t, g_3) = 3$ and $pos(t, g_7) = \infty$.

We will be comparing an ongoing trajectory against a set of trajectories T. Because of the low sampling rates, two taxis following the same path may have points mapping to

disjoint cells. In the right panel of Figure 4.3 we display the augmented trajectory from the left panel, along with a new trajectory (colored squares and green line). Some of the grid cells of the new trajectory fall on the augmented path (blue squares), while others fall in "empty" grid cells (orange and red cells). Because of the simplicity of the augmentation method, there is the possibility that the augmented path was not completely accurate, so we must account for this type of error: If a grid cell of the new trajectory is adjacent to one of the augmented cells, we consider it as if it were along the same path (orange cells), while if it is not adjacent to any augmented cells, we consider it as following a different path (red
adjacent
including
at least
is equal
a sample
$g_9 \in N(g$

g1	g2	g3	g4	g5	g6
g7	g8	g9	g10	g11	g12
g13	g14	g15	g16	g17	g18
g19	g20	g21	g22	g23	g24
g25	g26	g27	g28	g29	g30
g31	g32	g33	g34	g35	g36

Figure 4.4: Sample trajectory used to illustrate a cell's neighbours.

Problem Statement: We say a sub-trajectory t is anomalous with respect to T (and the fixed source-destination pair) if the path it follows rarely occurs in T. Given a fixed source-destination pair (S, D) with a set of trajectories T between them and an ongoing trajectory $t = \langle g_1, g_2, \ldots, g_n \rangle$ going from S to D, we would like to verify whether t is **anomalous** with respect to T. Furthermore, we would like to identify which parts of the trajectory are anomalous.

Definition 4.3.1. *We define a function* $hasPath : \mathcal{P}(\mathbb{T}) \times \mathbb{T} \to \mathcal{P}(\mathbb{T})$ *(where* $\mathcal{P}(X)$ *is the power set of* X*) that returns the set of trajectories that contain all of the points in* t *in the correct order. Note, however, that the points need not be* sequential*, it suffices that they appear in the same order.*

$$hasPath(T,t) = \left\{ t' \in T \;\middle|\; \begin{array}{l} \text{(i) } \forall 1 \leq i \leq n.\ N(g_i) \in t' \\ \text{(ii) } \forall 1 \leq i < j \leq n. \\ pos(t', N(g_i)) < pos(t', N(g_j)) \end{array} \right\} \tag{4.2}$$

For instance, if $T = \{t1, t2, t3\}$, in which $t1 = \langle g_1, g_2, g_3, g_4, g_5, g_8, g_9, g_{10} \rangle$, $t2 = \langle g_1, g_2, g_4, g_5, g_6, g_8, g_{10} \rangle$, and $t3 = \langle g_1, g_3, g_4, g_3, g_6, g_8, g_{10} \rangle$, and an ongoing trajectory $t = \langle g_1, g_2, g_5, g_8 \rangle$, then $hasPath(T, t) = \{t1, t2\}$. Given these definitions, we can specify when two trajectories are identical, given our augmentation method.

Definition 4.3.2. *Given a threshold* $0 \leq \theta \leq 1$*, a trajectory* t *is* θ**-anomalous** *with respect to a set of trajectories* T *if*

$$support(T,t) = \frac{|hasPath(T,t)|}{|T|} < \theta \tag{4.3}$$

4.4 iBOAT: Isolation-based On-line Anomalous Trajectory Detection

Having defined the necessary preliminaries, we are ready to present our method for anomalous trajectory detection. The process is split into an offline pre-processing phase and an online detection phase (see Figure 4.5). In the offline phase, we receive a set of historical trajectories which we classify and index using a sophisticated, but highly efficient, method. This allows us to respond to the on-line algorithm's queries in real-time. In the on-line phase, we process a series of incoming GPS points from each occupied taxi and provide an indication as to whether each point is anomalous or not. Once this on-going trajectory is completed, we add it to our historical database.

4.4.1 Offline Pre-processing

The offline phase is in charge of collecting and classifying a set of historical trajectories which will be used to determine "normal" routes between a source and destination pair. These historical trajectories must be accessible in an efficient manner in order to provide a real-time response.

We begin by grouping the trajectories according to source-destination pairs and the time of occurrence. It is important to separate trajectories according to the time of occurrence, since the "normalcy" of routes may depend on traffic patterns. We index the set of historical trajectories using a triplet $\langle sg, eg, time \rangle$, where sg is the starting grid cell, eg is the end grid cell and time is the time when the trajectory occurred. Note that in order to avoid the unnecessarily fine granularity of time, we divide time into coarser bins. Each set indexed

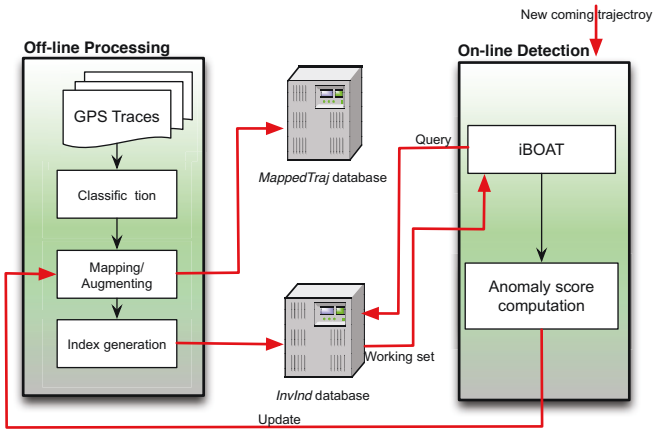


Figure 4.5: Overview of our approach.

by a triplet may contain both normal and anomalous trajectories. Once the trajectories have been classified, we map and augment them. For each trajectory, we store the resulting mapped grid cells (in correct order) in a record in the $MappedTraj$ database, and we index the records by their (unique) trajectory number as well as the time of occurrence.

To determine the anomalousness of a new mapped GPS point, we must be able to access *all* trajectories that contain this mapped point (or some point in its neighbourhood) in the same time bin. Using $MappedTraj$ for this purpose would be terribly inefficient, as it would imply searching through all trajectories for each new point. Instead, we make use of the Inverted Index Mechanism [169] for fast retrieval of relevant trajectories. For this mechanism, we maintain a second database where we maintain a record for each possible grid cell; the elements of each record are trajectory-position pairs, indicating the trajectories where the indexing grid cell appears, along with its position in that trajectory. For instance, consider the following trajectories:

$$t_1 : g_1 \rightarrow g_5 \rightarrow g_8 \rightarrow g_{10}$$
$$t_2 : g_1 \rightarrow g_2 \rightarrow g_5 \rightarrow g_8 \rightarrow g_5 \rightarrow g_9$$
$$t_3 : g_2 \rightarrow g_8 \rightarrow g_9$$

In the inverted index database $InvInd$, the record indexed by grid cell g_1 will be $InvInd(g_1) = \{(t_1, 1), (t_2, 1)\}$, the record indexed by g_5 will be $InvInd(g_5) = \{(t_1, 2), (t_2, 3), (t_2, 5)\}$, and the record indexed by g_9 will be $InvInd(g_9)\{(t_2, 6), (t_3, 3)\}$. Thus, if a new GPS point maps to g_9, by accessing $InvInd(g_9)$ we will immediately know that this grid cells occurs in trajectories t_2 and t_3.

We now have an efficient mechanism for accessing the trajectories that contain a particular grid cell. In the next section, we will use this incrementally (as new GPS points

arrive) to determine the anomalousness of an on-going trajectory.

4.4.2 *iBOAT* Algorithm

Our Isolation-Based On-line Anomaly Trajectory (*iBOAT*) detection method is based on the idea of *isolating* trajectories: anomalous (sub-)trajectories will be isolated from the majority of routes, while normal trajectories will be *supported* by a large number of trajectories. The less support a trajectory has, the higher its degree of anomalousness.

In Zhang et al. [158] the authors determine the anomalousness of a trajectory once the trajectory is completed. This is unfortunate, since it prevents one from providing alerts to the passenger while a trajectory is on-going. On the other hand, using purely density-based methods as described above will most likely result in inaccurate classifications. We aim to overcome this problem by using an *adaptive working window* that provides us with historical contexts to better determine the anomalousness of the incoming trajectory. We will use the definition of θ-anomalousness presented in Section 4.3 to describe our proposed algorithm.

Basic idea: The basic idea of *iBOAT* is to maintain an *adaptive working window* of the latest incoming GPS points to compare against the set of historical trajectories. As a new incoming point is added to the *adaptive working window*, the set of historical trajectories is pruned by removing any trajectories that are *inconsistent* with the sub-trajectory in the *adaptive working window*. New points continue to be added to the *working window* as long as the support of the sub-trajectory in the *adaptive working window* is above θ. If the support drops below θ, then the *adaptive working window* is reduced to contain only the latest GPS point. We outline this approach in Algorithm 4.1.

We maintain a *working set* of trajectories (initially equal to T), and an *adaptive working window* w. After $i-1$ entries received, our partial trajectory t (*adaptive working window*) consists of $\langle p_1, p_2, \ldots, p_{i-1} \rangle$ and we have a *working set* T_{i-1}. Upon arrival of point p_i, we map it to grid cell g_i (Line 8), and concatenate g_i to the *adaptive working window* w (Line 9). We then compute $support(T_{i-1}, w)$ (Line 10); If its value is less than θ, the trajectory points contained in the *adaptive working window* are said to have anomaly, then point p_i is considered anomalous so it is added to the set of anomalous points χ (Line 13), and we reset the *working set* (Line 14) as well as the *adaptive working window* (Line 15); otherwise, we set $T_i = hasPath(T_{i-1}, w)$ (Line 11). This procedure is repeated as long as the trip doesn't reach the destination. Note that $T_0 = T$, and that every time an anomalous point is encountered, the *working set* is reset to the original trajectory set T. This resetting is what enables our adaptive algorithm to accurately detect anomalous sub-trajectories in real-time with a finer granularity than the fixed-window approach (with $k > 1$). Additionally, by reducing the *working set* with each incoming point, the adaptive

Algorithm 4.1 *iBOAT* with adaptive window

Input: *incoming trajectory* - $t = \langle p_1, p_2, \cdots \rangle$; T - set of mapped and augmented historical trajectories; θ - anomaly threshold;
Output: *score*; χ- set of anomalous points

$\chi \leftarrow \emptyset$//initialization
$T_0 \leftarrow T$
$i \leftarrow 0$ //Position in incoming trajectory
$w \leftarrow \emptyset$ //Adaptive window from t
$score(0) \leftarrow 0$
while the testing trajectory is not completed **do**
    $i \leftarrow i + 1$
    $g_i = \rho(p_i)$
    $w \leftarrow w \cdot g_i$
    $support(i) = |hasPath(T_{i-1}, w)|/|T_{i-1}|$
    $T_i \leftarrow hasPath(T_{i-1}, w)$ //working set reduced
    if $support(i) < \theta$ **then**
        $\chi \leftarrow \chi \cup p_i$
        $T_i \leftarrow T$ //reset the working set
        $w \leftarrow g_i$
    end if
    $score(i) = score(i-1) + \sigma(support(i)) * dist(p_{i-1}, p_i)$
end while

approach has a computational advantage over the fixed-window approach.

To illustrate the process, we will use a running example, as shown in Figure 4.6. We assume there are three common routes that drivers take when delivering passengers from S to D: There are 100 taxi drivers who have taken *Route 1* (in black), 200 taxi drivers have taken *Route 2* (in red), and 150 drivers have taken *Route 3* (in blue). The test trajectory is depicted using the numbered yellow circles and the purple line (indicating the order of arrival of the points). We can immediately see that although the test trajectory visits only "common" cells in the initial part, it does so in reverse order between points 4 and 7. In the beginning, the *adaptive working window* will grow to contain points $\langle g_1, g_2, g_3, g_4 \rangle$, since this sub-trajectory has enough *support*. However, when g_5 is added to the *adaptive working window*, the support of this sub-trajectory drops below the threshold, thus g_5 is considered as an anomalous point and the new *adaptive working window* contains only g_5. The size of *adaptive working window* would not increase (only containing the single latest GPS point) until receiving g_7, and now the *adaptive working window* will be $\langle g_6, g_7 \rangle$. Again, the *working window* will shrink to contain only a single point throughout the anomalous section ($\langle g_8, g_9, g_{10} \rangle$). When the trajectory is completed, *iBOAT* will return $\chi = \{g_5, g_6, g_8, g_9, g_{10}\}$ as the set of anomalous points.

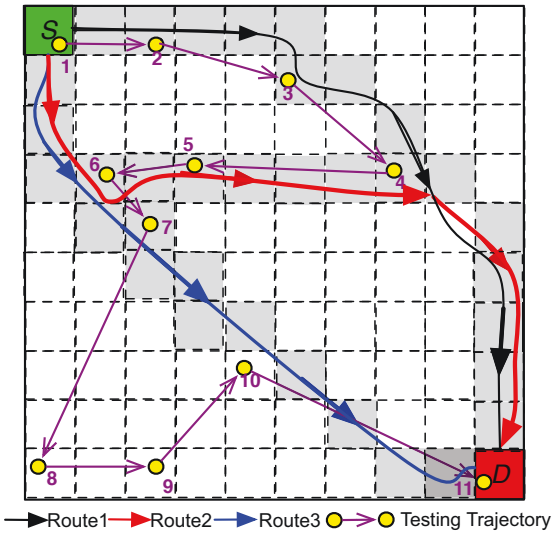


Figure 4.6: Running example for *iBOAT*.

We can also consider a simple variant of *iBOAT*: maintaining a fixed-sized window. In this approach, the *sliding window* consists only of the most recent k points. Specifically, given a set of trajectories T and an ongoing trajectory $t = \langle p_1, p_2, \ldots, p_n \rangle$, we verify whether the last k-sized sub-trajectory from t occurs with enough frequency in T to determine if it is anomalous. Note that when $k = 1$, we have the density method used for comparison in [25]. Following the example in Figure 4.6, we have the following results for different values of k:

$$\chi = \begin{cases} \{g_8, g_9\} & \text{If } k = 1 \\ \{g_5, g_6, g_8, g_9, g_{10}\} & \text{If } k = 2 \\ \{g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11}\} & \text{If } k = 3 \end{cases}$$

Note that the size of χ depends on the value of k: for the same anomalous trajectory, the larger k is, the larger χ would be. While the size of χ for $k = 2$ and *iBOAT* is closer to that of the real anomaly segments, it produces larger size of χ when k increases, leading to excessive counting of the anomaly segments. But in the case of $k = 1$, the size of χ is much smaller than that of the real anomaly segments. This explains why the anomaly detection algorithm with $k = 2$ and *adaptive window* outperforms that with $k = 1$ or $k \geq 3$. However, in some specific cases, as can be seen in Section 4.5, the proposed *iBOAT* method with adaptive window can detect certain anomalous trajectories that the fixed sliding window methods are not able to detect, making *iBOAT* the most effective anomaly detection approach.

Anomaly score: As the trajectory is on-going, we maintain an anomalous score, which

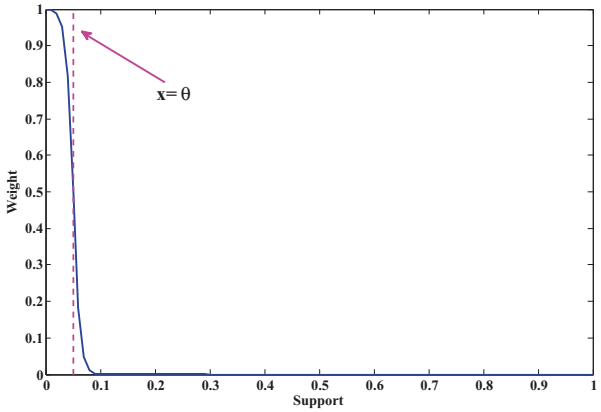


Figure 4.7: Weighting function σ.

will be used to provide alerts and rank the trajectories once they are completed. Intuitively, a trajectory with smaller *support* and longer anomalous distance should be ranked higher, so we compute this score based on the length of the anomalous sub-section, as well as the density in each anomalous sub-section, rather than only summing the length of each anomalous part [34]. We weigh the support according to the function σ, which is a logistic function (shown in Figure 4.7).

$$\sigma(x) = \frac{1}{1 + e^{\lambda(x-\theta)}}$$

Here, λ is a temperature parameter and θ is the aforementioned threshold. For our experiments, we choose $\lambda = 150$. This function will assign a larger weight to very low supports, and the weight will drop to zero for values above θ. The advantage of using this weighting function is that it *smoothes* the cutoff point imposed by the chosen threshold θ; in a sense, it plays a similar role as sigmoid functions in neural networks. For each incoming point p_i, we compute its score as shown in line 17 of the Algorithm 4.1: we add the score for the previous point to the distance just travelled multiplied by the weighted support (note that we also do for the fixed-window approach).

Given the way the on-going score is computed, once the trajectory is completed after n steps, we have the final score as given by the following equation, which is a weighted sum of the distance between points.

$$score = score(n) = \sum_{i=2}^{n} \frac{1}{1 + e^{\lambda(support(i)-\theta)}} dist(p_i, p_{i-1}) \tag{4.4}$$

where $dist : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is the standard sphere distance between two points.

4.5 Empirical Evaluation

In this section, we provide an empirical evaluation and analysis of our proposed approaches. All the experiments are run in MATLAB on an Intel Xeon W3500 PC with 12GB RAM running Windows 7.

4.5.1 Datasets

Out of the 7.35 million of trajectories extracted from the one month GPS records of 7,600 taxis, we picked nine source-destination pairs[1]. (T-1 through T-9) with sufficient trajectories between them (at least 450, but on average over 1000), and asked volunteers to *manually* label whether the trajectories are anomalous or not. On average, about 5.1% of the trajectories are labelled as anomalous. We summarize the information for each dataset in Table 4.1.

Table 4.1: Datasets used in our experiments.

	#Trajectories	#Anomalousness(%)
T-1	453	15(3.3%)
T-2	1494	57(3.8%)
T-3	528	43(8.1%)
T-4	946	58(6.1%)
T-5	1018	68(6.7%)
T-6	1369	72(5.3%)
T-7	1310	67(5.1%)
T-8	1216	71(5.8%)
T-9	1254	24(1.9%)

4.5.2 Evaluation Criteria

A classified trajectory will fall into one of four scenarios: True Positive (TP), when an anomalous trajectory is correctly classified as anomalous; False Positive (FP), when a normal trajectory is incorrectly classified as anomalous; False Negative (FN), when an anomalous trajectory is incorrectly classified as normal; True Negative (TN), when a normal trajectory is correctly classified as normal. The True Positive Rate (TPR) measures the proportion of correctly labelled anomalous trajectories, and is defined as:

$$TPR = \frac{TP}{TP + FN} \tag{4.5}$$

1. The source and destination areas are twice as big as the regular grid cells

The False Positive Rate (FPR) measures the proportion of *false alarms* (*i.e.* normal trajectories that are labelled as anomalous), and is defined as:

$$FPR = \frac{FP}{FP + TN} \tag{4.6}$$

A perfect classifier will have $TPR = 1$ and $FPR = 0$. In a Receiver Operating Characteristic (ROC) [44] curve we plot FPR on the x-axis and TPR on the y-axis, which indicates the tradeoff between false alarms and accurate classifications. By measuring the Area Under Curve (AUC), we can quantify this tradeoff.

4.5.3 Experimental Results

To test *iBOAT*, we selected a trajectory t as an ongoing trajectory from a dataset T and used both *iBOAT* and *fix-window* approaches with $\theta = 0.05$. In Section 4.5.4.1 we will discuss the effect different choices of θ has on the performance. In Figure 4.8(a), we display the output of our method for a test trajectory from T-6, where we plot the set of trajectories $T - \{t\}$ in light blue; for the test trajectory (t), the anomalous points are drawn in red and the rest (normal points) in dark blue. As can be seen, our method can accurately detect which parts of a trajectory are anomalous and which are normal. In Figure 4.8(b) and (c), we plot $support(T - \{t\}, t)$ (see equation (4.3)) and the *score* (see equation (4.4)) for the ongoing trajectory t. We can see that the value of *support* is a clear indication of when trajectories become anomalous, and that there is little difference between the different variants of *iBOAT*. However, it should be noted that there is a trailing *lag* for the fixed-window approach, equal to k. This is because the last anomalous point in an anomalous sub-trajectory will be included in the following k sub-trajectories. Although setting $k = 1$ will solve the lag problem, this minimal window size contains no contextual information of the trajectory, and will therefore have poor prediction quality. This was observed in [158] (therein referred to as the density method), and will be evident in the figures on the next page.

4.5.4 Varying Parameters

To better understand *iBOAT*, we conduct experiments to study its performance (in terms of running time and accuracy) under different parameter settings. We choose the three largest datasets (T-2,T-6 and T-7). We begin by varying the choice of θ in section 4.5.4.1. In Section 4.5.4.2 we vary the size of the datasets; specifically, for each of the three datasets, we choose $n = \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, \ldots\}$ trajectories randomly to serve as the historical trajectories. We measure the *average time*, which is the average amount of processing time per *completed* trajectory.

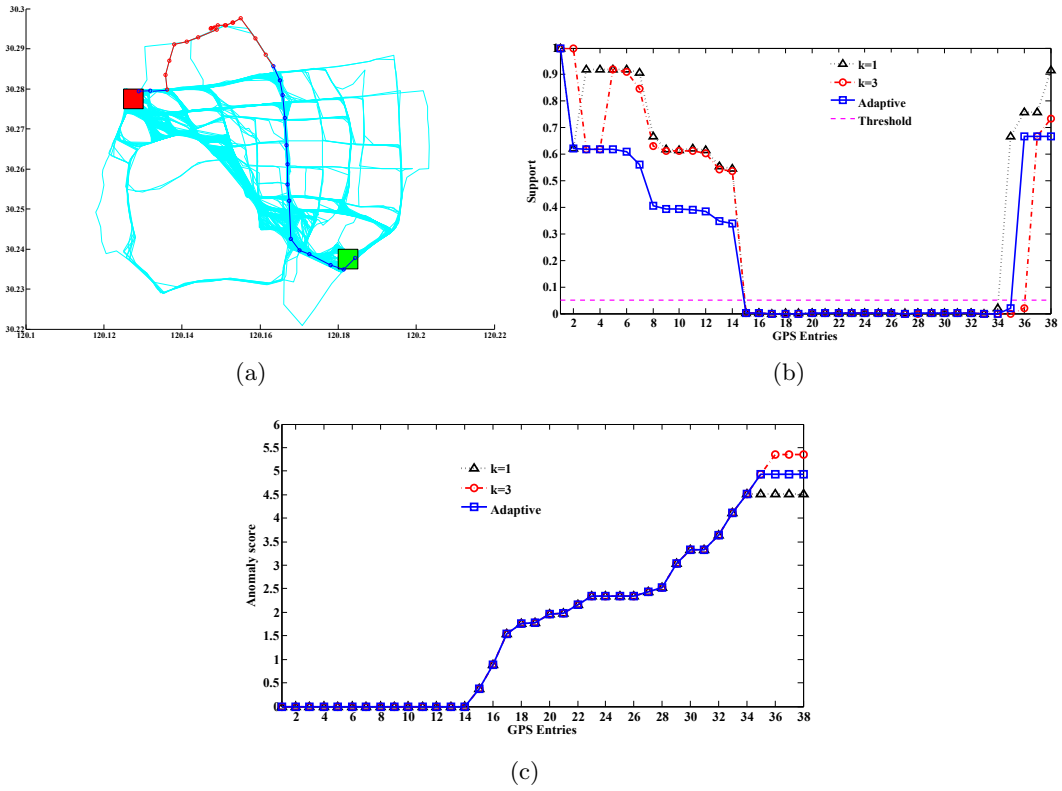


Figure 4.8: Detected anomalous sub-trajectories from T-6 using *iBOAT* (a); Plot of ongoing *support* (b); Plot of ongoing *score* (c).

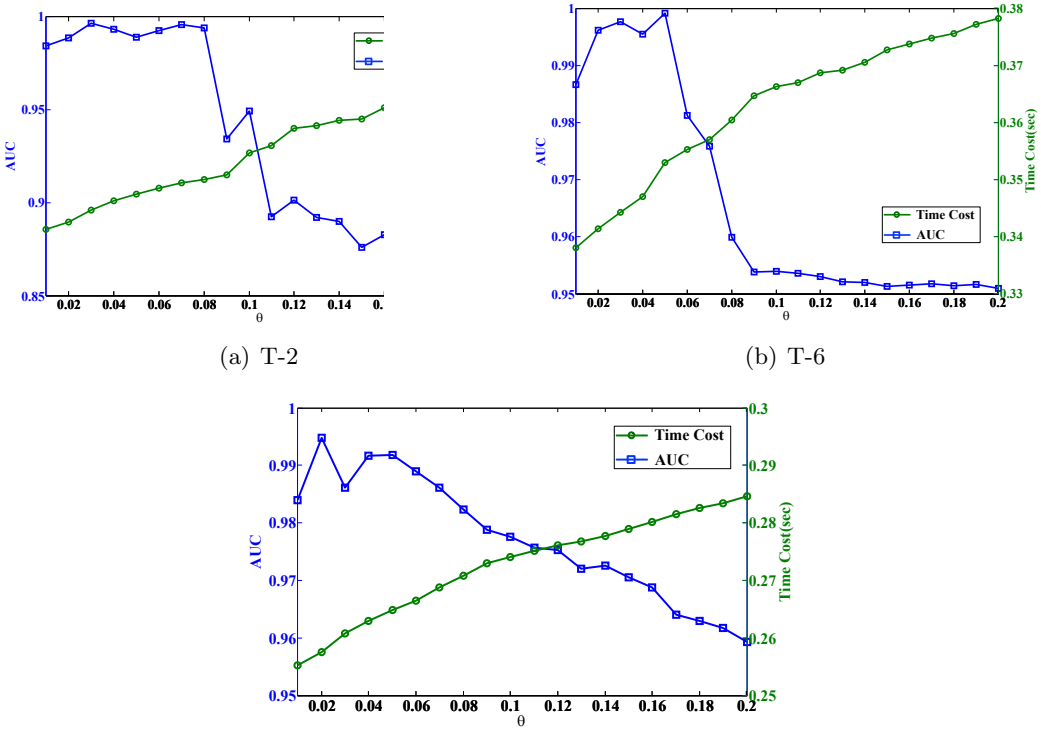
4.5.4.1 Varying θ

Since θ is the threshold value for determining anomalousness, it is important to investigate its effect on the performance of the algorithm. We study the effect on performance when θ ranges between 0.01 and 0.2.

In Figure 4.9, we plot the AUC and average time for different values of θ. We can see that θ should not be set any higher than 0.1, since beyond this the performance would decrease significantly. The average time increases with θ. This is because as θ becomes larger, the working set is reset more frequently, resulting in larger working sets on average. We can also see that our choice of $\theta = 0.05$ is reasonable, as it has good accuracy with low average time.

4.5.4.2 Varying n

It is evident that the average time will be longer with larger values of n, since there are more comparisons necessary for each incoming GPS point; on the other hand, if n is too



(a) T-2

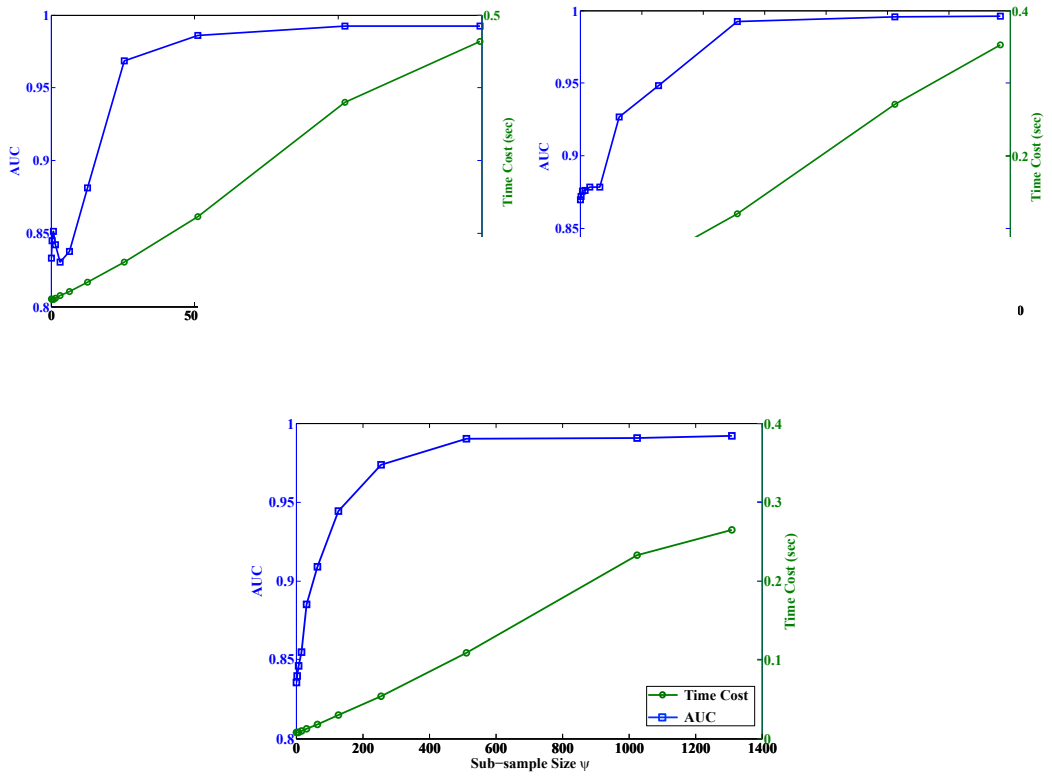
(b) T-6

(c) T-7

Figure 4.9: The AUC value (blue) and average time(green) under varying θ.

small, then more trajectories will be isolated, since there are fewer trajectories to support it. It is thus important to investigate how many trajectories are necessary between two endpoints for *iBOAT* to return accurate results. In Figure 4.10, we plot the AUC value and average time for different values of n. We can see that *iBOAT* achieves remarkable performance even with a small sub-sample size; the results suggest that datasets have around 500 trajectories to guarantee a reasonable performance.

The above analysis suggests that if we maintain a fixed number of trajectories, we can ensure good performance at a low computing and storage cost. Trajectories can be maintained in a First-in-First-out (FIFO) queue, as new trajectories are coming and processed, they can replace the oldest ones in the queue. This technique can also *capture* the change of distribution of trajectories. By applying this technique in road network change detection, it will result in a fast and effective detection of even newly opened or closed roads, which will be discussed further in Section 4.6. This also suggests that if one has limited resources, it is possible to maintain datasets of *fixed* size. Trajectories are maintained in a FIFO queue, so as new trajectories are completed, they replace the oldest trajectories in the database. This idea may also result in road network changes being reflected in the database faster



(c) T-7

Figure 4.10: The AUC value (blue) and average time(green) under varying n.

than if we maintain all past trajectories; this will be discussed further in Section 4.6.

4.5.5 Adaptive versus Fixed-window Approach

In Figure 4.11 we plot the ROC-curve for T-1 and T-8 respectively. From figures, we can see our proposed *iBOAT*, *iBAT* and baseline algorithms all achieve quite high True Positive Rate (TPR) with quite low False Positive Rate (FPR), and *iBOAT* performs the best among them. In more detail, for *iBOAT* algorithm, the TPR reaches around 95% at a very low FPR (5%).

We also display the AUC values of the different approaches on the nine datasets in Table 4.2. While the density approach ($k = 1$) has the worst performance, our proposed *iBOAT* method slightly outperforms the fixed sliding widow approach with $k = 2$, and the fixed sliding window method with $k = 2$ is better than that with $k = 1$ and $k \geq 3$. As explained in Section 4.2, the fixed sliding window method with $k = 1$ is worse than that of $k = 2$ because the anomalies detected are *fewer* than the actual ones; while the fixed sliding

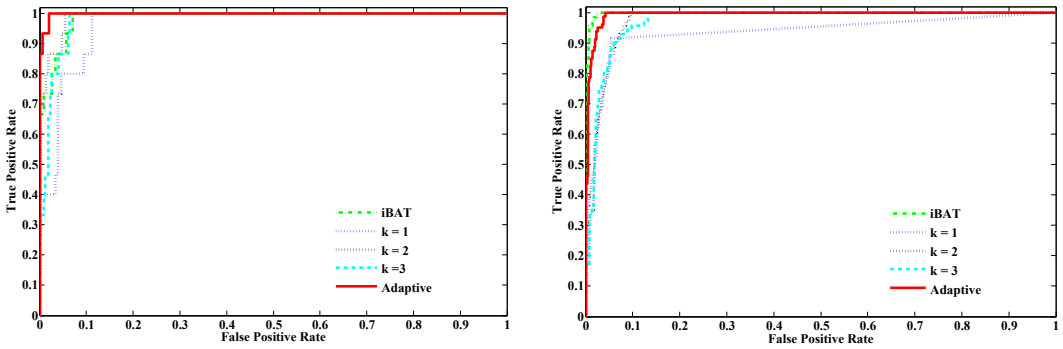


Figure 4.11: The ROC curves for T-1 (left) and T-8 (right).

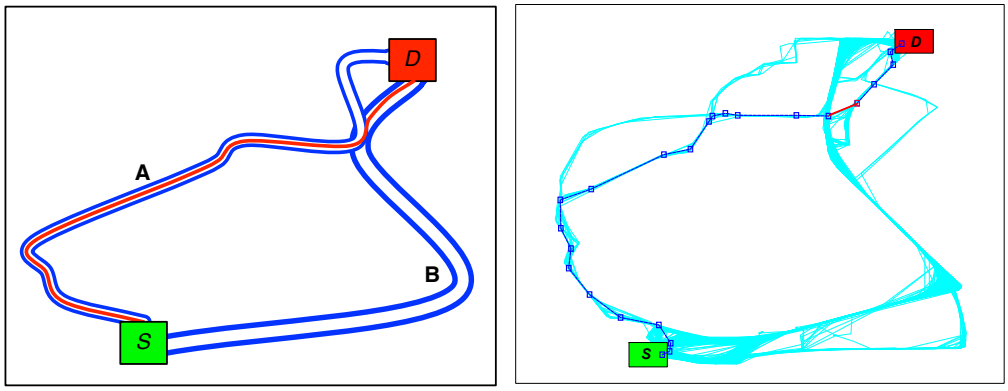
window method with $k = 3$ is worse than that of $k = 2$ because the anomalies detected are much *more* than the actual ones. The performance of the fixed-window approach with $k = 2$ and the adaptive approach are nearly identical. This is because for the anomalous sections, the adaptive approach ends up using a window of size 2, just as $k = 2$. The advantage of the fixed-window approach is that it requires a very small amount of memory for real-time anomalous detection, while the adaptive method requires memory proportional to the size of the longest "normal" sub-trajectory. In practice, this difference is negligible. In the following paragraph, we will use an example to demonstrate that the adaptive approach has an advantage over the fixed-window approach due to its use of longer historical "contexts".

Table 4.2: AUC values of the different algorithms.

	$k = 1$	$k = 2$	$k = 3$	*Adaptive*
T-1	0.9635	0.9904	0.9811	0.9985
T-2	0.9367	0.9902	0.9887	0.9952
T-3	0.8140	0.9733	0.9152	0.9962
T-4	0.9005	0.9586	0.9575	0.9890
T-5	0.9323	0.9885	0.9821	0.9967
T-6	0.9227	0.9912	0.9840	0.9952
T-7	0.8806	0.9853	0.9849	0.9937
T-8	0.9438	0.9739	0.9724	0.9937
T-9	0.9788	0.9991	0.9987	0.9995

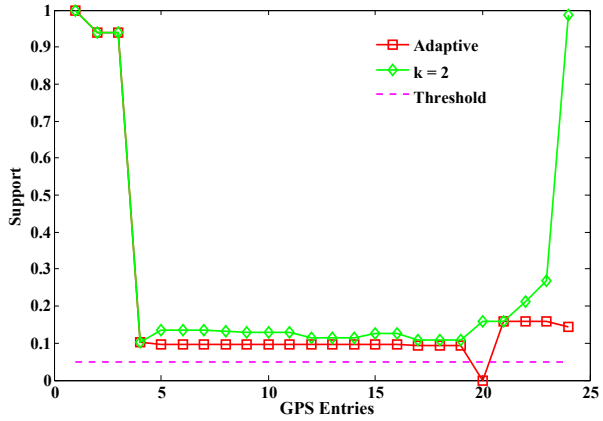
In Figure 4.12, we display an anomalous trajectory that "switches" from one normal route to another. The fixed-window method with $k = 2$ is not able to detect this anomalous switch. Going from point 19 to point 20 seems normal since this sequence occurs in route A, and going from point 20 to point 21 also seems normal since it occurs in route B. On the other hand, *iBOAT* would maintain the entire route up to the point when the driver

switches routes and would immediately detect it as an anomalous point. Although this example is specific to window sizes equal to 2, similar situations (with longer overlaps between routes) will produce a similar effect.



(a) Illustration of situation.

(b) Real trajectories.



(c) Ongoing *support* from *iBOAT*.

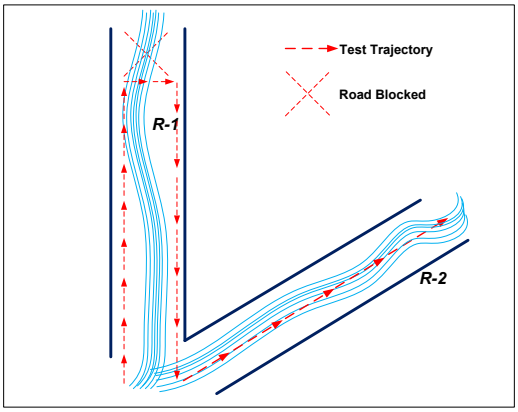
Figure 4.12: A situation the fixed-window method ($k = 2$) fails to classify as anomalous: two normal routes (route A and B) are in dark blue; an anomalous trajectory (in red) switches from route A to route B at their intersection.

4.5.6 *iBOAT* versus *iBAT*

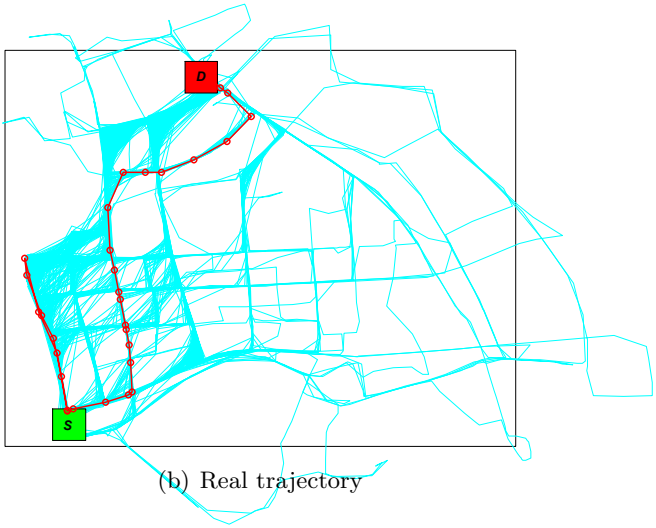
iBAT is our preliminary version of anomaly detection method [158], which only works when the trajectory is completed. In order to determine whether a trajectory is anomalous, *iBAT* picks cells from the testing trajectory at random to split the collection of trajectories into those that contain the cell and those that do not. This process is repeated until the trajectory is isolated, or until there are no more cells in the trajectory. Usually the number

of cells required to isolate anomalous trajectories will be much less than the number of cells in the trajectory. This isolation procedure is repeated a number of times and $\mathbb{E}(n(t))$, the average number of cells required to isolate a trajectory, is used to compute the score, which is proportional to $2^{-\mathbb{E}(n(t))}$.

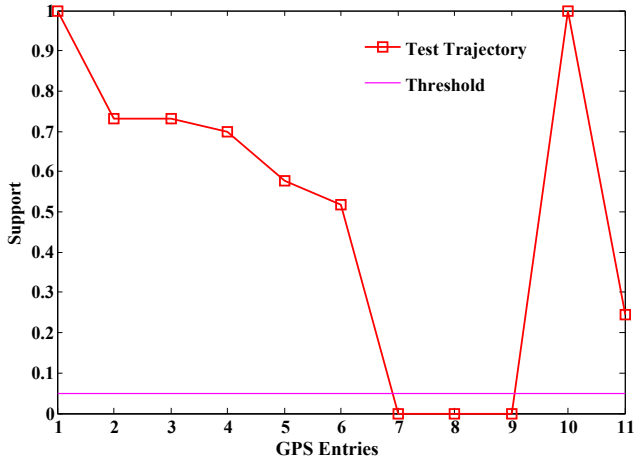
Our proposed method is a clear improvement over *iBAT* on two levels. First of all, we are able to determine which *parts* of a trajectory are anomalous, in contrast to *iBAT* which only classifies *full* trajectories as anomalous. Second of all, our method works in *real-time*: we can detect anomalous sections as soon as they occur, and do not require a full trajectory as an input.



(a) Illustration of situation



(b) Real trajectory



(c) Ongoing *support* from *iBOAT*.

Figure 4.13: A trajectory where the taxi had to retrace its path due to a blocked route.

In Figure 4.13, we show an example where a road block has forced a taxi to retrace its path and search for another route to its destination. We focus on the first part of the

trajectory where the taxi retraces its steps. In Figure 4.13(c), we can see the *support* is accurately identifying the anomalous section of the trajectory. We determined what anomalous ranking (based on the scores) both methods assign this partial trajectory in comparison with all other trajectories[2]. Out of 1418 trajectories, *iBOAT* ranked this trajectory in 48th place, while *iBAT* ranked it in 831th place. Furthermore, *iBAT* assigned this trajectory a score of 0.4342, which is below their usual 0.5 threshold [158]. Thus, while *iBAT* is unable to detect that this trajectory is anomalous, *iBOAT* has ranked it amongst the top 3% of anomalous trajectories, as well as identifying which part is anomalous. The reason *iBAT* fails in this example is that their method does not take the *order* the points appear in into consideration; despite the fact that the taxi is retracing its steps and actually going *away* from the destination, it is only visiting "normal" grid cells.

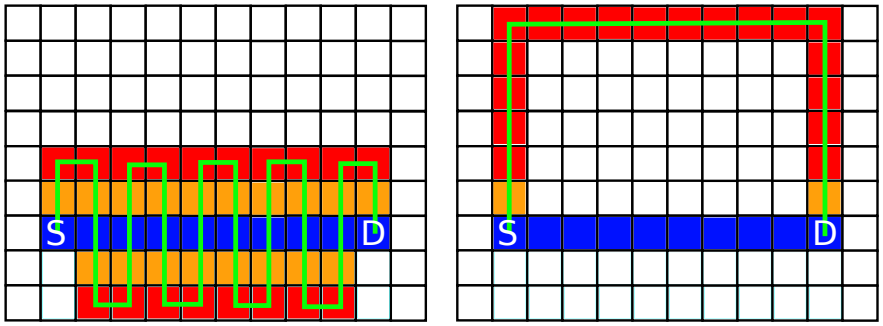


Figure 4.14: Two anomalous trajectories of different types. The normal trajectory between S and D is in blue, cells adjacent to normal cells are in orange, and anomalous cells in red.

Now consider the hypothetical example in Figure 4.14 which highlights the differences in the two scoring functions. In this simple situation, the value $\mathbb{E}(n(t))$ for *iBAT* is just the expected number of times their algorithm must pick cells before an anomalous cell (in red) is picked. This is essentially a *Bernoulli trial*[3] with "success" probability p equal to the proportion of anomalous cells to total number of cells in the trajectory. It is well known that the expected number of trials before reaching success in a *Bernoulli trial* is given by $1/p$. Let n be the number of cells in the straight line between S and D, then trajectories of the form on the left will have $2n-2$ anomalous cells and $5n-4$ total cells, while trajectories of the form on the right will have $2n-2$ anomalous cells and $2n+2$ total cells. It follows that for trajectories of the form on the left $\mathbb{E}(n(t)) = \frac{5n-4}{2n-2} \to \frac{5}{2} \Rightarrow score \approx 0.1768$; for trajectories of the form on the right $\mathbb{E}(n(t)) = \frac{2n+2}{2n-2} \to 1 \Rightarrow score = 0.5$. Thus, *iBAT* will qualify trajectories of the form on the right as more anomalous than those on the left. This runs contrary to intuition, which would perceive trajectories like the one on the left

2. A higher ranking means higher degree of anomalousness.
3. http://en.wikipedia.org/wiki/Bernoulli_trial

at least as anomalous as the one on the right, given that the path taken is much longer and they are clearly taking longer routes than necessary. The scoring method of *iBOAT*, on the other hand, would assign the left trajectory an anomalous score around 33% higher than the one on the right.

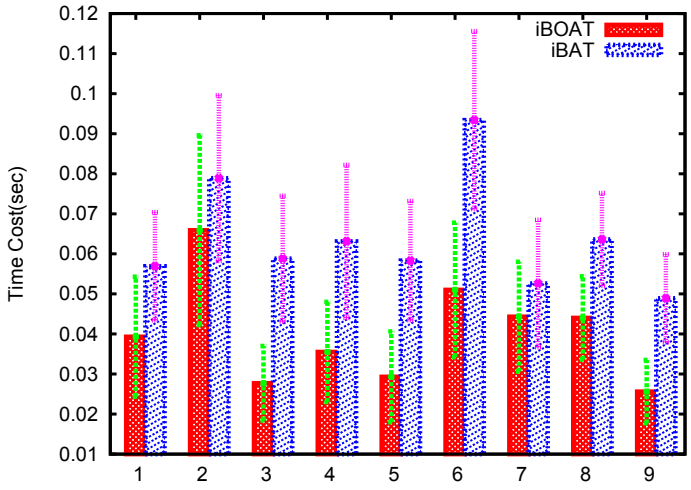


Figure 4.15: Running times of *iBOAT* and *iBAT* on all the datasets.

Finally, we compared the running time of both algorithms on all the datasets, and we display the results in Figure 4.15. We computed the running time for checking each trajectory in each dataset, and averaged over the size of the dataset. Although *iBAT* will usually check fewer grid cells than *iBOAT* (since one anomalous cell is enough to classify the trajectory as anomalous), *iBAT* is based on random cell selections, so they must average over m runs; as in [158], we set $m = 50$. We can see that *iBOAT* is consistently faster than *iBAT* on all datasets.

4.6 Applications

Aside from its use for real-time detection of anomalous behaviours, *iBOAT* can also be used for a number of other applications. The first application is that it can be used to deny possible excuses for fraud behaviours, because some cunning taxi drivers may use detour reasons such as traffic accidents on roads as excuses. Before elaborating the application, we first perform an in-depth analysis of detected anomalous trajectories. The second application is for detecting changes in the road networks.

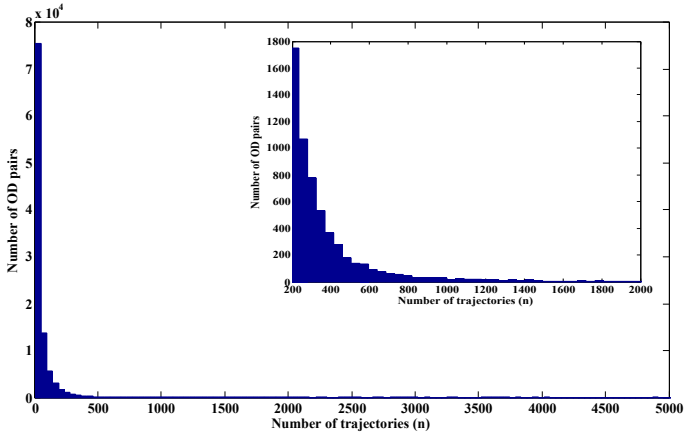


Figure 4.16: Relationship between the number of OD pairs ($y - axis$) and the number of trajectories between an OD pair ($x - axis$).

4.6.1 Statistical Study [130]

One main motivation for this work is fraud detection and the ability to alert passengers of fraudulent behaviours. Travel distance and time are two crucial parameters to judge if a certain taxi trajectory is a long detouring trip committed by fraudulent behaviours. Thus, in this sub-section we perform an analysis of the anomalous trajectories to attempt to discover whether anomalous behaviours are the result of conscious decisions to commit fraud, by visualizing where most of the anomalous trips begin and comparing the average distance and travel time of anomalous routes with that of normal ones.

For this analysis, we collected around 441 million GPS records in March 2010. In Figure 4.16, we show a histogram for the number of trajectories between an OD pair. Inside the figure, an enlarged view with the number of trajectories in the range of [200 2000] is also shown. We can see that most of OD pairs have trajectories less than 50, which may be not enough for detecting anomalous trajectories correctly, so we just exclude those OD pairs for study; the number of trajectories between an OD pair can be over 3,000. In this study, the statistical results are obtained over OD pairs which have more than 200 trajectories. For those OD pairs, we detected about 438,000 anomalous trajectories out of 7.35 million trips. This provides us with an opportunity to perform a statistical analysis of the anomalous trajectories, in the hope of uncovering common characteristics of the trajectories and driving "trends" of those responsible for anomalous behaviours.

In Figure 4.17, we display the areas where most of the anomalous trips began. We can see that many of the places are long-distance coach stations, where tourists would generally arrive. It is not surprising that they are responsible for a large fraction of the anomalous trajectories. This provides strong evidence that anomalous behaviours are conscious deci-

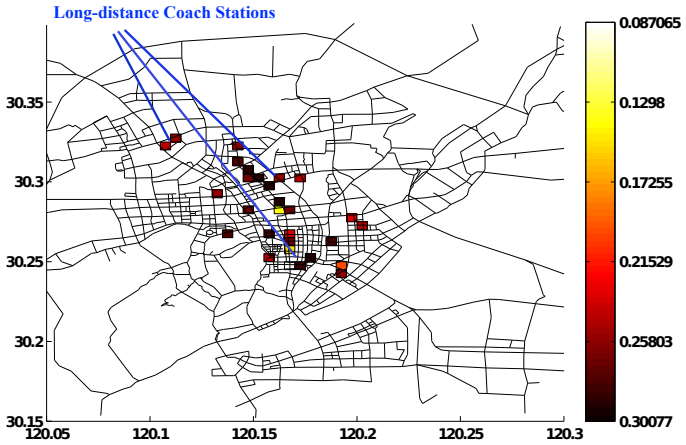


Figure 4.17: Areas where most of the anomalous trips began.

sions. Note that it is possible that some passengers who are not familiar with the city can not provide the detailed address of destination. This might have certain impact on choosing the best route for drivers. However, as we group the historical trajectories with vague source and destination (in an area with around $500m \times 500m$), and judge the ongoing trajectory by comparing it with historical ones with same source and destination, thus it will not cause problem when the system shows the reasonably longer trajectory travelled to the unfamiliar passengers.

Table 4.3: Distribution of anomalous trajectories with respect to travelling distance and time.

	Travel time		
Trip length	$[0, minT)$	$[minT, maxT]$	$(maxT, \infty)$
$[0, minD)$	0.0013	0.0137	0.0117
$[minD, maxD]$	0.0062	0.1063	0.0881
$(maxD, \infty)$	0.0045	0.1522	0.6162

In most research revolving around detecting anomalous taxi driving behaviours, one is mainly interested in detecting fraudulent activities. We believe that many of these fraudulent trips will take passengers along routes that are much longer than what is considered normal. Given our database of historical trajectories, we can determine the length of the longest normal trip between a source and a destination; we can safely say that an anomalous trip is *detouring* if the trip distance is longer than this maximal distance. For a source-destination pair, we denote $maxD$ and $minD$ as the maximal and minimal lengths amongst the normal trips. It may be the case that a longer trip is actually a faster route, placing in doubt whether the driver's actions were fraudulent. We could try to determine $maxT$ and

$minT$ for the travelling time taken between two points, but due to varying traffic conditions, these values have a high variability. Because of this, for each source-destination pair, we compute the mean time amongst the normal trajectories, μ_T, as well as the standard deviation σ_T. We then define our boundaries as $maxT = \mu_T + \sigma_T$ and $minT = \mu_T - \sigma_T$. In Table 4.3, we display the distribution of the anomalous trips with respect to these classifications. We can see that over 60% of the anomalous trajectories are taking longer time and distance than the maximal normal trajectories, clearly suggesting that fraud is one of the main motivating factors behind anomalous taxi driving behaviours.

4.6.2 Deny Possible Excuses

From the evidence provided in Table 4.3, we can see that a large proportion of detected anomalous trajectories are actually due to detours. Some cunning drivers who took a detour may ague that 1) they are unfamiliar with this area; or 2) unexpected car accidents or heavy traffic occurred. There exist some reasons due to the motivations of passengers, such as some passengers may ask taxi drivers to take detour for either picking up his/her friend or avoiding traffic jam. In these cases, they will not complain even the system shows the longer detour trajectory to them.

(a) The anomalous trajectory (red solid) is compared with previous trips of the same driver (green dashed).

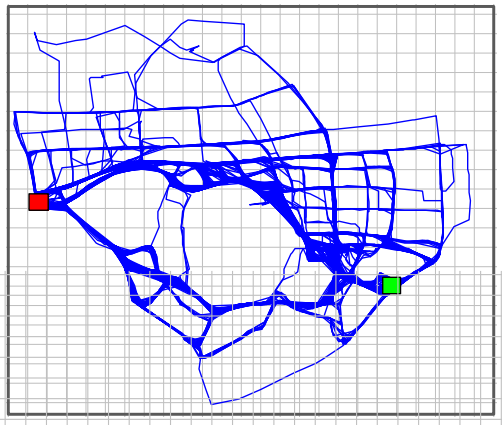
(b) The anomalous trajectory is compared with trajectories in the same time slot (green dashed).

Figure 4.18: Avoiding excuses for taxi driving fraud detection.

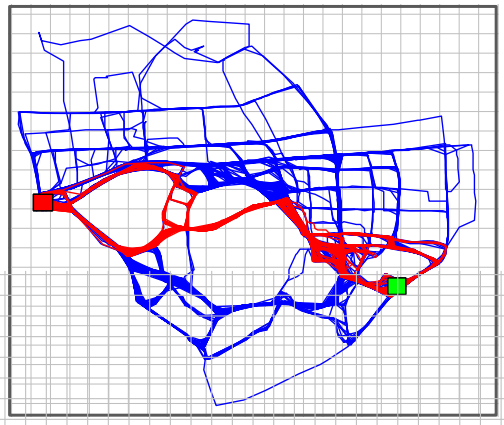
In order to justify these excuses, more evidence needs to be provided. To deal with the first excuse, if an anomalous trajectory is detected, we can get all previous trajectories of the corresponding driver to verify whether he truly has had little previous experience driving through this area. Note that one taxi may be operated by more than one driver, so a mechanism for detecting driver shift change may be necessary. This is an interesting problem in itself but outside of the scope of this work. For the second excuse, we can find all the trajectories that took place around the same time and area to check whether there is some traffic disturbance. In Figure 4.18, we give an illustrative example. Suppose we detect

an anomalous trajectory (solid red line in the left panel of Figure 4.18). We compare it with the driver's previous trips (dashed green lines) between the same source and destination and can verify that this driver has experience driving between these two points. In parallel, we recall all the trips (dashed green lines in the right panel of Figure 4.18) that happened around that same time slot since time-of-day has impact on the occurrence of anomalous routes. Since in this example we can see that many other drivers did not detour, it is unlikely that there is a traffic disturbance. Having discredited both types of excuses, we can be more confident in our assessment of fraud.

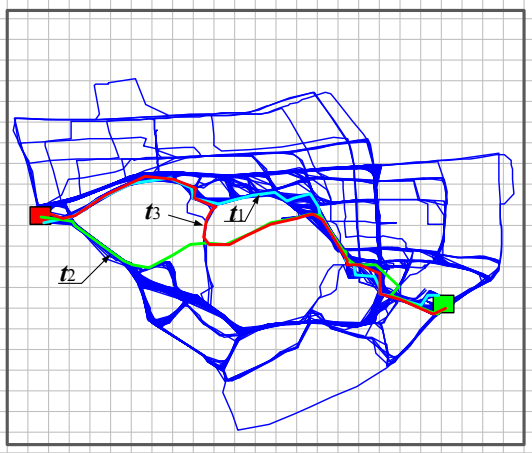
4.6.3 Detecting Road Network Changes



(a) Delivering road network before newly built road.



(b) Delivering road network after newly built road.



(c) Considered trajectories.

Figure 4.19: Application of new road detection case (Best viewed in the digital version).

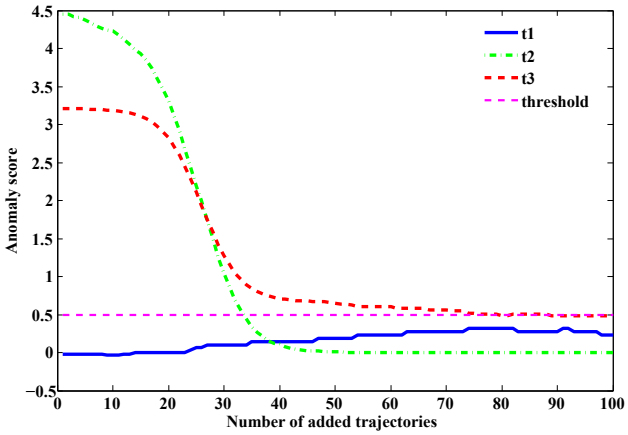


Figure 4.20: Anomaly score change with the accumulating of trajectories

Unfortunately, we do not have information about the closure or re-opening of roads in the road network, so our experiments will rely on testing various simulated cases. We begin by specifying what we mean by road change: we classify any of the following events as a road change:

♢ A new road has been added to the network.

♢ A previously closed road has been re-opened.

♢ A previously opened road has been closed.

We will address how to detect each of these changes separately. We simulate a road opening by first picking a road segment that will be "added" later. We then proceed by removing all historical trajectories that pass through this road segment. Throughout the simulation, we incrementally add these trajectories, thereby simulating the increased usage of the "newly" added road. Figure 4.19(a) displays the historical trajectories before the new road was "added", and Figure 4.19(b) shows the incrementally added trajectories that pass through the newly added road (red solid lines).

Our proposed method for detecting a new road using *iBOAT* is as follows.

1. We maintain a historical database, implemented as a FIFO queue, containing a fixed number of the most recent trajectories.
2. We group together identical anomalous trajectories and maintain counts for each distinct group of anomalous trajectories. We say two trajectories A and B are **identical** trajectory if and only if $hasPath(\{A\}, B) = \{A\}$ and $hasPath(\{B\}, A) = \{B\}$
3. If the count for any one of these anomalous groups begins increasing quickly and regularly, it is a good indication of a new road. As more trajectories begin using this new road, the anomaly score will gradually decrease until reaching a normal level.

4. We capitalize on *iBOAT*'s ability to detect anomalous sub-trajectories, and use this to properly identify the newly added segments from the anomalous group in question.

Consider the three sample trajectories displayed in Figure 4.19(c). Trajectory t_1 represents a path frequently taken by taxi drivers which does not cross the newly added road. Trajectory t_2 represents the trajectories usually followed by drivers when using the newly added route; trajectory t_3 is similar to t_2 but with a lower frequency. In Figure 4.20 we plot the anomaly scores for these three types of trajectories over the number of added trajectories using the new road. We can see that for t_1, the anomaly score increases slightly due to some drivers being diverted to the newly added road, thereby decreasing its *support*. Trajectory t_2 is quick to fall below the anomalous threshold, as it becomes a popular route amongst drivers. Trajectory t_3 has a similar shape, but it fails to fall below the threshold due to its lower popularity amongst drivers.

By virtue of implementing the historical database as a FIFO queue, an identical approach can be used to detect whether previously closed roads have re-opened or whether a road has been recently closed.

4.7 Concluding Remarks

In this chapter, we have proposed a new algorithm for fast *real-time* detection of anomalous trajectories obtained from GPS devices equipped in taxis. Rather than using time and distance to judge whether a test trajectory is anomalous or not directly, we compare it against a set of sampled historical trajectories with same source-destination pair. In addition to classifying completed trip trajectories as anomalous or normal, *iBOAT* can work with ongoing trajectories and can determine which parts of a trajectory are responsible for its anomalousness. We validated *iBOAT* on a large dataset of taxi GPS trajectories recorded over a month and found our method achieved excellent performance (AUC$\geq$ 0.99 for all datasets) which is comparable to *iBAT*'s performance; however, we demonstrated a number of examples that highlight *iBOAT*'s advantage over *iBAT* and the sliding window method. We further showcased *iBOAT*'s use for fraudulent behaviour analysis and detecting road network changes. The result suggests that most anomalous trajectories are in fact due to fraud. We also provide evidence to deny possible excuses for fraud behaviours.

In the future, we plan to broaden this work in several directions: 1) We plan to explore using statistical approaches to enhance detection performance and data processing efficiency; 2) We also plan to develop a real-life anomalous trajectory detection system with the proposed method; 3) To address the issue that some source-destination pairs may not have enough samples, we would like to either cluster source and/or destination areas in a *principled* way to "combine" trajectories from different source-destination pairs, or

simply collect more historical data, or partition the map into different grid sizes; 4) We would like to conduct further analysis on the GPS traces obtained to better understand the motivations and characteristics of fraudulent activities.

Chapter 5

B-Planner: Planning Bidirectional Night Bus Routes

Contents

5.1 Introduction

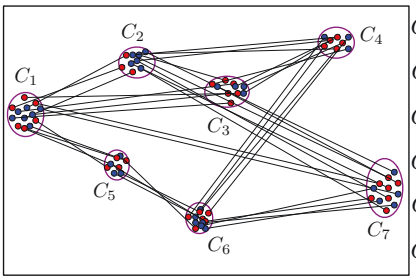
Buses are a popular and economical way for people to travel around the city, and they are generally "greener" than cars and taxis as they help decrease traffic congestion, fuel

consumption, carbon dioxide emission and travel cost [2]. Thus, for sustainable city development, people are encouraged to take public transportations, such as buses, for commuting between home and work, for visit, etc. In many cities, the daytime bus transportation systems are usually well designed; however, during late nights, most bus systems are out of service, leaving taxis as the only option for travelling around the city. In order to provide cost-effective and environment-friendly transport to citizens, many cities start to plan night-through bus routes.

Previously, bus route planning mainly relied on costly human surveys to understand people's mobility patterns in a city scale [11, 54]. Although this approach was proved to be workable, the time and cost spent in the survey process were quite substantial. Moreover, such an approach is not able to accommodate the frequent change in the road network and traffic, especially for cities which experience rapid development. Fortunately, with the wide deployment of GPS devices and wireless communication in taxis, rich information about taxis including where and when passengers are picked-up or dropped-off, how much driving time is needed to travel between two points, are hidden in the taxi GPS data. Better understandings of the *social dynamics* about where are the popular passenger pick-up/drop-off locations and origin-destination pairs and the *traffic dynamics* about how much driving time is needed to travel between popular OD pairs at nighttime make it possible to accurately plan new night-bus routes which expect the maximum number of passengers along the routes.

In this work, we intend to explore the bi-directional night-bus route design problem leveraging the taxi GPS traces. This problem can be divided into two sub-problems: 1) the candidate bus stop identification; 2) the best bi-directional bus route selection. For the first sub-problem, we need to identify the candidate bus stops which are associated with locations having big number of taxi passenger pick-up and drop-off records (*PDRs*). Additionally, the bus stops should be evenly distributed in the "hot" districts to facilitate people's access. After the candidate bus stops are identified, the next step is to select a bi-directional bus route which connects the bus origin and a sequence of bus stops to the destination, expecting the maximum number of passengers in both directions given a specific bus operation time, frequency, and total travel time. Fortunately, the taxi GPS traces contain quantitative spatial-temporal information about all taxi trips. By mining the taxi GPS data, we can deduce where are the "hot" areas for taxi passengers and how many passengers would potentially travel along a certain route in a specific time duration. Therefore, the bi-directional night-bus route design becomes a problem of comparing the number of passengers of all valid bus routes giving certain time constraints.

However, identifying the candidate bus stops from taxi GPS data and enumerating the top-ranked bi-directional bus routes efficiently are not trivial and straight-forward. To the



Passenger Flow

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	0	15	7	0	9	0	5
C_2	18	0	12	8	0	0	12
C_3	14	3	0	20	0	0	9
C_4	0	5	10	0	0	22	0
C_5	12	0	0	0	0	9	0
C_6	0	0	0	13	0	0	10
C_7	3	17	13	0	0	4	0

Travel Time(in minute)

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_1	0	8	10	15	9	15	25
C_2	9	0	4	10	16	19	22
C_3	11	3	0	6	7	8	14
C_4	16	12	7	0	29	27	10
C_5	8	14	6	25	0	5	12
C_6	14	18	7	26	4	0	10
C_7	27	21	15	12	14	9	0

Figure 5.1: An illustrative example of the taxi GPS traces (left); the passenger flow (middle), and the travel time among bus stops (right).

best of our knowledge, there is still no work reported on this topic. For example, given the taxi GPS trajectories of night time for a certain time period, let us say that seven dense taxi pick-up/drop-off locations (i.e. $C_1 \sim C_7$) have been identified as candidate bus stops, with the process shown in Figure 5.1, where C_1 and C_7 are the bus origin and destination, respectively, and the corresponding passenger flow and travel time among stops are shown in the middle and right panels of Figure 5.1. The objective of bi-directional bus route design is to find a bi-directional bus route ($C_1 \rightarrow C_7$ and $C_7 \rightarrow C_1$) with maximum number of passengers expected given the bus operation time constraints. Apparently, to design an effective bus route, the following research challenges need to be addressed:

- ♢ ***Candidate bus stop identification***: The taxi passenger pick-up and drop-off points are distributed in the whole city, with some areas having more pick-up/drop-off records (*PDRs*) than other areas, but there is no clear guideline about where the bus stops should be put.
- ♢ ***Trade off between the number of passengers and travel time***: To deliver more passengers, the best bus route should go through more bus stops (e.g. go through all stops between C_1 to C_7), but this will take more travel time. Hence, a non-trivial trade-off is needed.
- ♢ ***Passenger flow accumulation effect***: Assuming there is no taxi passenger travelling from C_4 to C_7 , if we plan the route as $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_7$, then the significant passenger flow in $C_2 \rightarrow C_4$ and $C_3 \rightarrow C_4$ cannot be accommodated. Alternatively, by including C_4 in the route as $C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4 \rightarrow C_7$, this passenger flows can be accommodated with the cost of adding one stop. Therefore, we need to consider this accumulation effect, which tends to lead to a globally better solution.
- ♢ ***Dynamic passenger flow***: The passenger flows are usually different from time to

time, for example, the passenger flow during 23:00-24:00 can be very different from that during 3:00-4:00, thus we need to consider this dynamics when planning bus routes.

♢ ***Asymmetry of passenger flow and travel time***: It is easy to see that the best route in terms of passenger flow and travel time for one direction (from C_1 to C_7) is probably not the best one for the opposite direction (from C_7 to C_1), we thus need to select the bus route with maximum accumulated number of passengers in two directions.

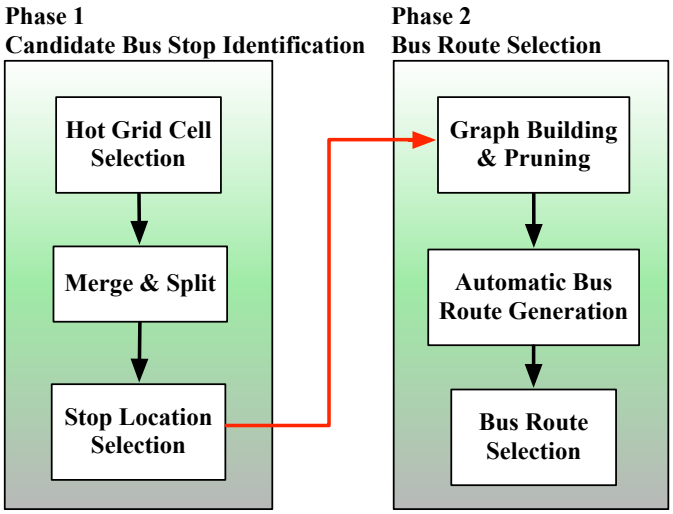


Figure 5.2: The two-phase bus route planning framework.

In this work, we address the above-mentioned challenges using a two-phase approach, with the process illustrated in Figure 5.2. Roughly speaking, in the first phase, we identify candidate bus stops by clustering and splitting hot areas; and then in the second phase, we propose several strategies to find best bus routes. Specifically, the main contributions of this work can be summarized as follows:

♢ First, we propose a two-phase approach to tackle the bi-directional night bus route design problem leveraging the taxi GPS traces. To the best of our knowledge, this is the first work on bi-directional night bus route design exploiting the taxi travel speed, time, pick-up and drop-off information in large-scale, real-world taxi GPS traces.

♢ Second, we develop a novel process with effective methods to cluster "hot" areas with dense passenger pick-up/drop-off, split big "hot" areas into walkable size ones and identify candidate bus stops. Moreover, we study how different thresholds in the merge and split algorithms affect bus stop identification results and final selected bus

routes.

- ♢ Third, we propose rules to build and prune the directed bus route graph. Based on the graph, we propose a new heuristic algorithm, named Bi-directional Probability based Spreading (*BPS*) algorithm, to select the best bi-directional bus route which can achieve the maximum number of passengers expected in two directions. It is verified that the *BPS* algorithm outperforms the top-k approach in the selection of best bus route. We also investigate the impact of different bus stop distances on the final bus routes selection.
- ♢ Finally, we determine the night bus capacity by computing the maximum number of passenger on buses for the selected bus route at different stops and different bus frequencies. To understand the impact of the new opened bus route on taxi services, we further report the passenger flow change along the bus route before and after the new bus route opened date.

The rest of this chapter is organized as follows. In Section 5.2, we first review the related work and show the difference from other work. In Section 5.3 we present the process for candidate bus stop identification and in Section 5.4 we elaborate the process for bus route graph building and pruning, automatic bus route generation and best route selection. Extensive evaluation results are reported in Section 5.5 to verify the effectiveness of the proposed approach. Finally, we conclude the work and chart the future directions in Section 5.6.

5.2 Related Work

The work about "route planning" in the *operational dynamics* which we have reviewed in Chapter 2 is relevant. In particular, the work [15] with the focus of designing public routes using taxi GPS data is of great relevance. The main goal of [15] is to mine historic taxi GPS trips to suggest a flexible bus route. The work first clusters trips with similar starting time, duration, origin and destination; it then attempts to identify the route that connects multiple dense taxi trip clusters. The work is different from ours as it only chooses the route which maximizes the sum of each connected trip cluster. In another word, it does not consider the time constraints and the accumulated effects among connection stops, thus it would never include the path like $C_4 \rightarrow C_7$ of Figure 5.1 in the planned bus route, while our approach might include the path as long as the route expects the maximum number of accumulated passengers and the total travel time constraint can be met.

The work focusing on the bus network design, other than exploiting taxi GPS traces is also relevant. Bus network design is an intensively studied area in the urban planning and transportation field [8, 137, 138, 160]. The bus network design is known to be a complex,

non-linear, non-convex, multi-objective NP-hard problem [93,107,109,115,131]. The aim is to determine bus routes and operation frequencies that achieve certain objectives, subject to the constraints and passenger flows. The popular objectives include shortest route, shortest travel time, lowest operation cost, maximum passenger flow, maximum area coverage and maximum service quality while the constraints include time, capacity and resources [30,36, 69,163].

However, the selection of the objectives should take care of the operator as well as user requirements which are often conflicting, leading to design trade-off rather than an optimal solution. As noted in [109,115], early bus network design was mainly based on human survey to get passenger flows and user requirements, it relied heavily on heuristics and intuitive principles developed by a designer's own experience and practice. Recent work on bus network design also assumes that the passenger flows are given by user survey or population estimation, many complex optimization approaches have been proposed, and among them the best solving algorithms are based on heuristic procedures [73] to find near-optimal solutions. A detailed review about route network design can be found in [54].

Despite the renewed attention for bus network design, there is still no work addressing the bi-directional night-bus route design problem leveraging the taxi passenger OD flow data. Different from existing research, our work aims to find a bi-directional bus route with a fixed frequency, maximizing the number of passengers expected along the route subject to the total travel time constraint. This problem is different from the traditional Travelling Salesman Problem (TSP) [9,90] in nature, which aims to find the shortest path that visits each given location (node) exactly once. TSP evaluates different routes with exact N locations, which means all candidate stops should be included in the route. Our problem is also different from the shortest path finding problem [140], which intends to get the shortest path for a given OD pair. In our case, we have to consider the accumulated effect (passenger flows) from all previous stops to the current stop for choosing the bi-directional bus route.

5.3 Candidate Bus Stop Identification

In the proposed two-phase bus route planning framework, the objective of the phase one is to identify candidate bus stops by exploiting the taxi *PDRs*. Here, we describe our proposed process for identifying candidate bus stops. As illustrated in Figure 5.2, the whole process consists of three steps:

1. Divide the whole city into small equal-sized grid cells, mark those "hot" grid cells with high taxi passenger *PDRs* for further processing;
2. Merge the adjacent "hot" grid cells to form "hot" areas, divide each big area into

"walkable size" cluster;

3. Choose one grid cell as the candidate bus stop location in each walkable size "hot" cluster, by assuming that passengers from the same cluster would easily walk to the stop to take bus.

5.3.1 Hot Grid Cells and City Partitions

In this work, we first divide the city into equal-sized grid cells, with each cell about $10m \times 10m$ in size. In such a way, the whole city is partitioned into 5000×2500 cells in total. Out of all the grid cells, over 95% of them contain no taxi passenger *PDRs* as they are either lakes, mountains, buildings, and highways that cannot be stoppable by taxis, or suburb areas that people seldom travel to. We plot the Cumulative Distribution Function (CDF) curve for all grid cells having *PDRs*, as show in Figure 5.3. Out of them, over 90% grid cells have the value of *PDRs* per hour greater than 0.2. And we name these grid cells as "hot" ones. Most of hot grid cells have the value of *PDRs* per hour in the range of [0.2 1]. The percentage of hot grid cells is only about 0.11% of all grid cells (including grid cells do not have *PDRs*). It should be noted that the statistical results here are obtained only counting the taxi GPS data during the night time.

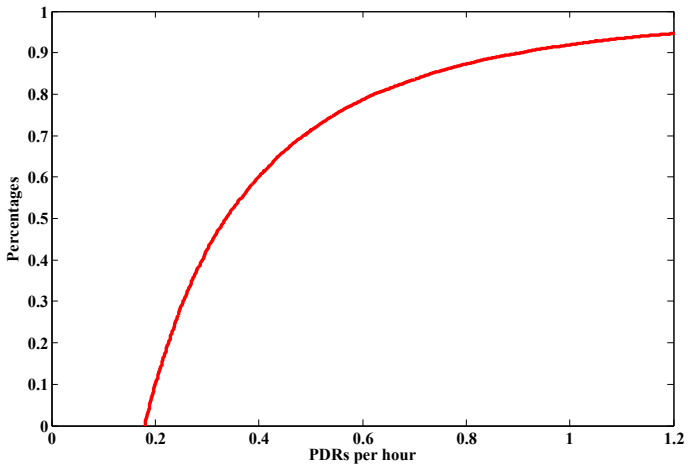


Figure 5.3: CDF result of grid cells having *PDRs*.

As each grid cell has maximum eight neighbors, if we define the connectivity degree (CD) of a "hot" grid cell as the number of "hot" neighboring cells, the CD of any grid cell will range from 0 to 8, where the "hot" grid cell with CD equals to 0 is called isolated cell. As the city is composed of mixed hot grid cells and common grid cells, both hot cells and common cells form irregular "hot areas" and "common areas" as a consequence of same type of cells being adjacent to each other. These "hot areas" are also called city partitions,

Figure 5.4: City partitions near Hangzhou Railway Station.

as shown in Figure 5.4. Apparently, some small partitions (e.g., the green one in Figure 5.4) can be very close to some big ones (e.g., the red one in Figure 5.4). It would be necessary to consider all the city partitions globally in order to plan the bus stop locations, thus city partitions close to each other had better merge to form big clusters for better overall bus stop distribution. In the next section, we propose a simple strategy to merge the close partitions into bigger clusters.

5.3.2 Cluster Merging and Splitting

We present the cluster merging and splitting approach in Algorithm 5.1. After obtaining all city partitions, we sort them in a descending order according to the number of *PDRs* (Line 1). To merge the partitions close to each other iteratively, we propose to use the hottest partition to *absorb* its nearby partitions according to the descending order of *PDRs*, until no more nearby partitions meet the merging criterion (Line 8). Then we choose the next hottest partition to repeat the same process until all the partitions are checked (Lines 8∼12). The location of each partition is first initialized by computing the weighted average location of all grid cells using Eq. 5.1.

$$loc(P) = \frac{\sum_{i=1}^{N}(PDRs(g_i) * loc(g_i))}{\sum_{i=1}^{N} PDRs(g_i)} \tag{5.1}$$

where $loc(g_i)$ refers to the longitude/latitude of the member grid cell g_i.

Algorithm 5.1 Merge Algorithm

Input: List of partitions $\{P_i\}$
Output: List of clusters $\{C_i\}$
1: $P \leftarrow sort\ (P), (i = 1, 2, \cdots, n)$ // Sort P according to amount of its *PDRs* by descending order
2: $i = 1$;// Initialization
3: **while** $P \neq \emptyset$ **do**
4: $C_i = \{P_1\}$;
5: $P = P \backslash \{P_1\}$ // Remove P_1 from P
6: $k = |P|$;
7: **for** $j = 1 : 1 : k$ **do**
8: **if** $dist(C_i, P_j) < Th_1$ **then**
9: $C_i = C_i \cup P_j$ //*absorb* the closer partition
10: $P = P \backslash \{P_j\}$ //Remove P_j from P
11: **end if**
12: **end for**
13: $i = i + 1$;
14: **end while**

After merging one partition, the location of the combined cluster is updated (Line 9) and the absorbed partition is removed from the partition list (Line 10). The *dist* function refers to the distance between two given partitions. The algorithm will be terminated until no partitions can be merged to a new cluster (Line 3). A main parameter in the merge algorithm is Th_1 (Line 8), which controls *how far* a big cluster can absorb its nearby clusters. Intuitively, a bigger Th_1 would allow big clusters to absorb more nearby clusters, leading to fewer number of clusters in total but more big clusters. We will further investigate how Th_1 would affect the resulted best route parameters quantitatively in Section 5.5.2.2.

In general, the merged clusters can be classified into three groups according to their size (the size of cluster is defined as the minimal rectangle which covers all the grid cells): 1) with both height and width greater than Th_2; 2) with either height or width greater than Th_2; and 3) with both height and width less than Th_2 (where Th_2 is the maximum distance that passengers are willing to walk to reach a bus stop).

As for large clusters (Group 1 and 2), we adopt a simple strategy to split them. Specifically, for clusters in Group 1, we first split the big cluster into two sub-clusters, aiming to minimize the difference of *PDRs* of the resulted clusters both in horizonal and vertical directions; while for clusters in Group 2, we only need to split the cluster in one direction. We split the cluster in the horizonal direction if its height is greater than width, otherwise, we split it in the vertical direction, again with the goal of minimizing the number difference of *PDRs* of the split sub-clusters. With one split, one big cluster would produce two smaller

sub-clusters. Thus, a smaller Th_2 would need more splitting times, and also leads to more smaller clusters finally.

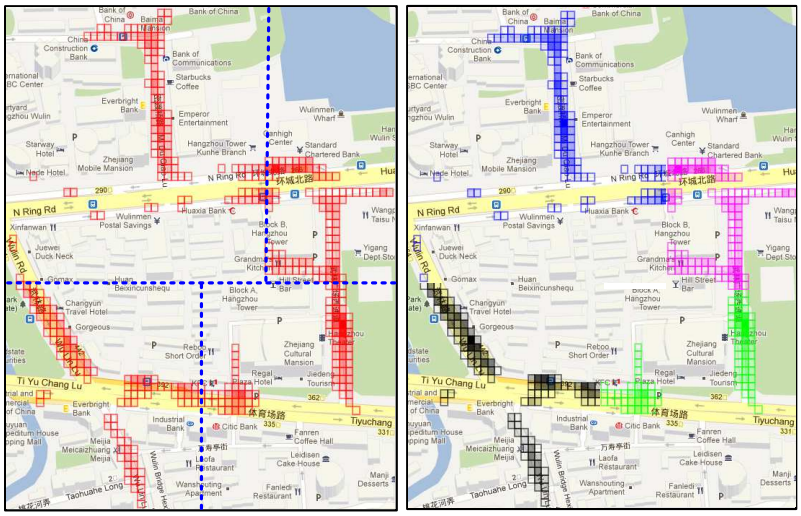


Figure 5.5: Illustrative example of splitting. Big cluster formed via merging (left). Big cluster split into 4 walkable size clusters (right, in four different colors).

Figure 5.5 shows an illustrative example of splitting a cluster into four sub-clusters with the proposed splitting strategy. The initial cluster belongs to Group 1 (Figure 5.5 (left)), the splitting is first done in the horizontal direction to produce two sub-clusters with similar *PDRs*. After the first splitting, two sub-clusters with width greater than Th_2 are generated (Th_2 is set to 500 meters for this example), thus both sub-clusters require a further splitting in the vertical direction. The final result with four split sub-clusters is shown in Figure 5.5 (right). We will also study how Th_2 would affect the resulted best route parameters in Section 5.5.2.2.

5.3.3 Candidate Bus Stop Location Selection

After *merging* and *splitting* operations, we obtain a big number of "hot" clusters with the size smaller than $Th_2 \times Th_2$, scattered in the dynamic districts of the city during late night. The next step is to select a *representative* grid cell in each cluster to serve as the location of candidate bus stop.

To select this *representative* grid cell, both the connectivity degree (CD) and the number of *PDRs* of each cell (i.e. hotness) in the cluster are taken into consideration. While the CD of a grid cell characterizes the accessibility of the cell, the number of *PDRs* is an indicator of its "hotness". The grid cell having the maximum value defined in Eq. 5.2 in each cluster is selected as the "center" of the cluster, marked as the location of the candidate bus stop.

$$\arg\max_i \left[w_1 \times \frac{CD(i)+1}{9} + w_2 \times \frac{PDRs(i)}{\sum_{i=1}^{n}(PDRs(i))} \right] \tag{5.2}$$

We set $w_1 = w_2 = 0.5$ in the evaluation, and totally we get 579 candidate bus stops in the city by using the taxi GPS data from Hangzhou, China. Note that different weight settings (i.e. w_1 and w_2) in Eq. 5.2 would only affect locations of the bus stop, and have no impact on the total number of bus stops.

5.4 Bus Route Selection

After fixing the candidate bus stops in Phase I, the aim of Phase II is to find the best bus route for a given OD, expecting to maximize the number of passengers expected under the time constraints in two directions (i.e. O→D and D→O).

In this section, we first approximate the passenger flow and the travel time between any two candidate stops using taxi GPS traces, then we present the bus route selection method which contains the following three-steps (shown in Figure 5.2):

1. Build the bus route graph and remove invalid nodes and edges iteratively based on certain criteria;
2. Automatically generate candidate bus routes with two proposed heuristic algorithms;
3. Select the bus route by comparing the expected number of passengers under the same total travel time constraint.

5.4.1 Passenger Flow and Travel Time Estimation

We record the travel demand and time information in two matrix, named passenger flow matrix (FM) and bus travel time matrix (TM). Each element in a matrix refers to the number of passengers or the bus travel time from one stop (ith) to another stop (jth, $i \neq j$). We count the total taxi trips from ith cluster to jth cluster as each stop is responsible for its cluster. We set the maximum waiting time for passengers at the stop as 30 minutes (equal to the bus operation frequency), so any pick-up or drop-off events taking place in this time window are counted. We simply assume the passenger flows among candidate bus stops remain unchanged during each 30-minutes duration. The final FM is got by averaging all flow matrix at different bus frequencies. We also assume TM keeps unchanged across the night time. $tm(s_i, s_j)$ is the average travel time multiplied by α, which is a constant. We set $\alpha = 1.5$ to consider the speed difference between taxis and buses. For the paths having no taxi trip occurring in history (for instance, nobody travels by taxi due to too short distance), we use $Ddist(s_i,s_j)/v$ to approximate $tm(s_i, s_j)$, where $Ddist(s_i, s_j)$ is the driving distance between s_i and s_j, and v is a constant and is set to 50 km/h. Figure 5.6

shows the final average passenger flow and bus travel time matrix. A pixel stands for the passenger flow or the travel time from one stop to another stop. Specifically, a brighter pixel represents a higher value.

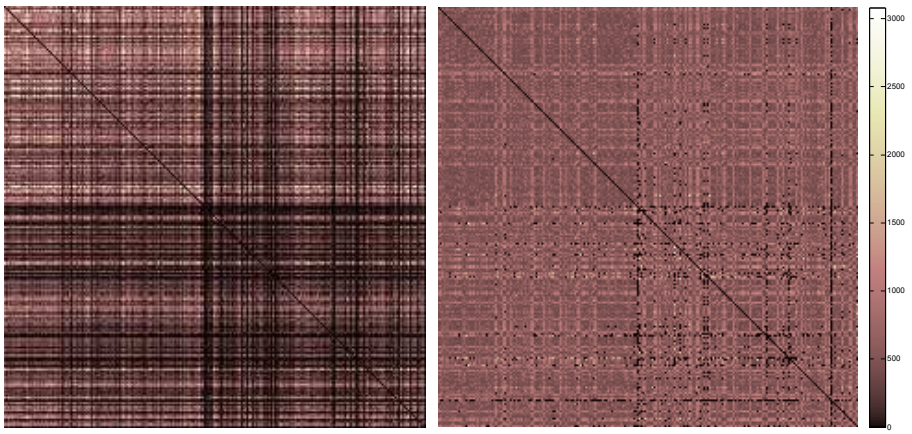


Figure 5.6: Average passenger flow (left) and bus travel time matrix (right).

5.4.2 Bus Route Graph Building and Pruning

Selecting the best bus route is a very challenging problem as two conflicting requirements must be met: one is to ensure that the bus route would traverse intermediate stops and finally reach the destination within a limited time; the other is to maximize the number of passengers accumulated along the route from all previous stops to the destination. For example, if we choose the stop with the heaviest passenger flow from the origin as the first node, and then keep choosing the next stop following the heaviest passenger flow principle, then we might neither be able to reach the destination, nor achieve the objective of having the maximum number of passengers accumulated along the route. To meet the above two requirements and follow the intuitive principles in bus route design, some basic criteria should be set for the building of the bus route graph and selection of the candidate bus route.

5.4.2.1 Route Graph Building Criteria

Obviously, there would be numerous stop combinations for a given OD pair, and only a small proportion of them meet the first or second requirement. In order to reduce the search space of possible stops and routes, we can build a bus route graph starting from origin to destination using heuristic rules. These rules are either derived from one of the above two requirements, or from the intuitive bus route design principle. For instance, from the shortest travel time perspective, the bus route should extend from origin towards the

direction of destination, which can be further converted into three rules: each new selected stop should be farther from the origin, closer to the destination, and farther from previous stops. From the intuitive bus route design principle, the bus stops should not be too far from each other, also the bus route should not comprise sharp zig-zag paths. These can also be translated into two criteria in building the bus route graph. Specifically, given the OD pair (s_1, s_n) and the candidate route $R = \langle s_1, s_2, \cdots, s_n \rangle$, we should follow the following criteria when building the bus route graph with stops (nodes) and directed edges among nodes.

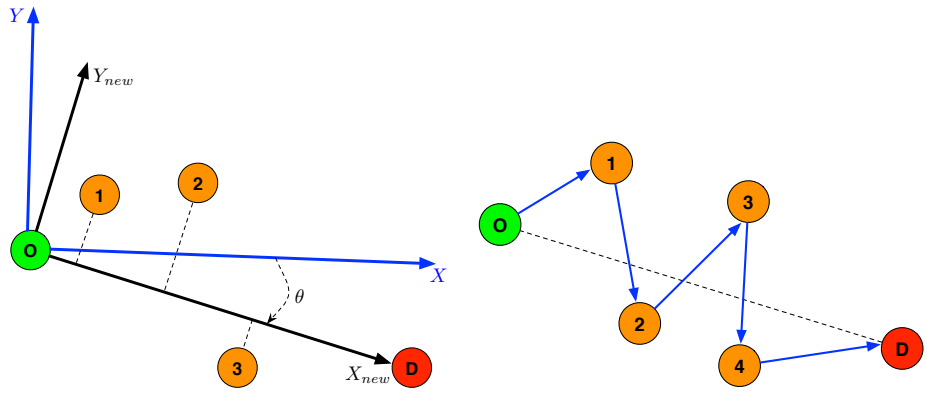


Figure 5.7: Demonstration of Criterion 2 (left) and Criterion 5 (right).

- *Criterion 1: Adequate stop distance*

$$dist(s_{i+1}, s_i) < \delta \quad (i = 1, 2, \cdots, n-1)$$

where δ is a user-specified parameter. It means the maximum distance between two consecutive stops. We will study the effect of varying δ values on the best route parameters in Section 5.5.

- *Criterion 2: Move forward*

$$x_{new}(i+1) > x_{new}(i) \quad (i = 1, 2, \cdots, n-1)$$
$$x_{new}(i) = x(i)\cos\theta + y(i)\sin\theta$$
$$\theta = \tan^{-1}\frac{y(n)}{x(n)}$$

$(x(i), y(i))$ of s_i is got by simply subtracting the longitude and latitude value to that of s_1. x_{new} is the X-axis value of stop in the new coordination which is with s_1 as the new origin, and from s_1 to s_n as the new direction of X-axis (see the left panel in Figure 5.7). This criterion guarantees the bus will always move forward along the OD direction.

- *Criterion 3: Origin-farther*

$$dist(s_{i+1}, s_1) > dist(s_i, s_1) \quad (i = 1, 2, \cdots, n-1)$$

This ensures that the bus will move away from the origin s_1 farther in each step.

- *Criterion 4: Destination-closer*

$$dist(s_{i+1}, s_n) < dist(s_i, s_n) \quad (i = 1, 2, \cdots, n-1)$$

This ensures the bus will move closer to the destination s_n in each step.

- *Criterion 5: No zigzag route*

$$\arg\min_{s_j}(dist(s_{i+1}, s_j)) = s_i \quad (j = 1, 2, \cdots, i)$$

Criterion 5 ensures the smoothness of the route. Sharp zigzag path along the OD direction is not allowed. The route demonstrated in the right panel of Figure 5.7 should not happen, as it violates the criterion. We can see $\arg\min(dist(s_3, s_j)) = s_1 \neq s_2 \quad (j = 1, 2)$, also $\arg\min(dist(s_4, s_j)) = s_2 \neq s_3 \quad (j = 1, 2, 3)$.

5.4.2.2 Graph Building & Pruning

The aim of graph building is to construct a directed graph with nodes and links given an OD pair, in which the nodes are the stops, and edges link the stop to its next possible stops, regardless of passenger flows among them. While the goal of graph pruning is to remove invalid edges and nodes according to the proposed criteria.

Graph Building: Given the bus route origin and destination, their locations are firstly used to narrow down the choice of valid candidate stops, only the candidate stops lying between them are under consideration. For each stop within the range, we determine links to its next possible stops according to the proposed *Criterion 1∼4*. The process will terminate when all stops have been checked. At last, stops having no edges would be excluded. We show the graph building procedure in Algorithm 5.2 (Line 2∼4). For each node, we summarize the method of how to determine its links in Algorithm 5.2. Links will be determined (Line 4) if pair (s_i, s_j) meets the proposed *Criterion 1∼4* (Line 3).

As *Criterion 5* is related to all stops in one bus route, so we use it to prune the route graph after it is built. Figure 5.8 (left) shows an illustrated example about a generated bus route directed graph. Note that the graph is built based on the geographical constraints, so the edge may have no taxi passenger flow on itself.

Graph Pruning: Some nodes and edges can be further pruned because they are not valid for candidate bus route selection. To be specific, nodes without in-coming edges (if

Algorithm 5.2 Graph Building Algorithm

Input: S – List of stops in the range (nodes);
Output: $G = (S, E)$ – Graph;

1: **for** Each node (s_i) in the list **do**
2: **for** Each node (s_j) in the list (exclude s_i) **do**
3: **if** $x_{new}(i), x_{new}(j) \in [x_{new}(1), x_{new}(n)]$ and //We suppose $x_{new}(1) < x_{new}(n)$
$dist(s_j, s_i) < \delta$ and //Criteria 1
$x_{new}(j) > x_{new}(i)$ and //Criteria 2
$dist(s_j, s_1) > dist(s_i, s_1)$ and //Criteria 3
$dist(s_j, s_n) < dist(s_i, s_n)$ //Criteria 4
then
4: $E(s_i, s_j) = 1$ //Link s_i to s_j:
5: **end if**//Identify its links to other nodes
6: **end for**
7: **end for**

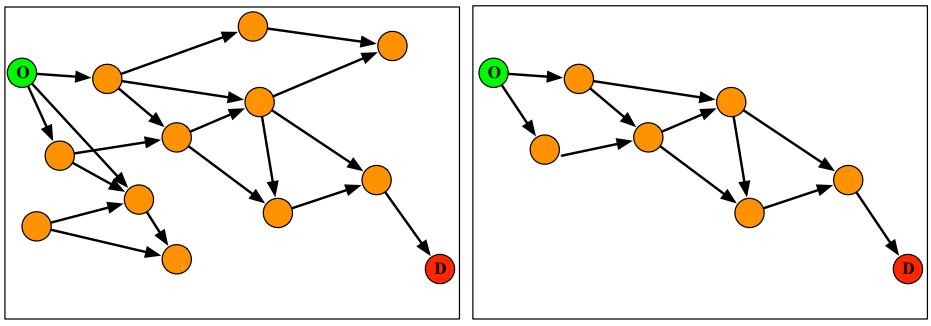


Figure 5.8: A bus route directed graph for a given OD. The route graph is got by graph building algorithm (left) and its corresponding graph after applying graph pruning (right).

not origin) or out-going edges (if not destination) should be deleted as they will not form any valid routes with the bus route OD pair.

We first calculate all the nodes' in-coming and out-going degrees. Afterwards nodes (excluding the given OD) together with related edges would be iteratively deleted from the graph if their in-coming or out-going degree is zero. At last a graph with only one zero in-coming degree node (i.e. the given origin) and one zero out-going degree node (i.e. the given destination) would be generated. After graph pruning, all the bus routes starting from the source and following the edges in the graph would eventually reach the destination. Figure 5.8 (right) displays the resulted graph after applying pruning to the graph in Figure 5.8 (left).

Graph for D→O: An intuitive way of building route graph for D→O is to run the previous two steps again, with the D as the new origin and O as the new destination.

However, *Theorem* 5.4.1 below ensures that the route graph from D to O is just the same as that from O to D, with all the edges having opposite directions.

Theorem 5.4.1. *If $R = \langle s_1, s_2, \cdots, s_n \rangle$ is a candidate bus route for pair (s_1, s_n), then its reversed route $\bar{R} = \langle s_n, s_{n-1}, \cdots, s_1 \rangle$ will be the candidate bus route for (s_n, s_1) pair.*

Proof. To prove $\bar{R}$ is the candidate bus route for (s_n, s_1) pair, we just need to check whether it meets all the five criteria. It is obviously that $\bar{R}$ meets the first four criteria. For *Criterion 5*, given a particular node s_i $(1 < i < n-1)$ in R, we can derive its two closest nodes are s_{i-1} and s_{i+1}. Thus $\arg\min_{s_j}(dist(s_i, s_j)) = s_{i+1}$ $(j = n, n-1, \cdots, i+1)$ will hold. □

5.4.3 Automatic Candidate Bus Route Generation

Based on the graph constructed in the previous section, we first propose our probability based spreading algorithm for O→D, then followed by the Bi-directional probability based spreading (*BPS*) approach, which can select the best bus routes in both directions.

Probability based Spreading Algorithm: Though we have removed invalid nodes and edges through graph pruning, the problem of enumerating all possible routes from given source to destination is proved to be NP hard. Indeed, it is also unnecessary to enumerate all possible routes and compare them all, because most of routes are *dominated* by few others.

Definition 1. We say R_i *dominates* R_j iif: 1) $\mathcal{T}(R_i) \leq \mathcal{T}(R_j)$; 2) $Num(R_i) > Num(R_j)$. The route which is not *dominated* by others in the route set is defined as a *skyline route*.

where $\mathcal{T}$ and Num are the total travel time and number of expected delivered passengers. We compute them based on Eqns. 5.3 and 5.4. The *skyline route* definition is similar to that in [49], and the rational behind is that only routes with less travel time but larger number of passengers should be selected. *Skyline* detector [20] will prune the routes which are dominated by skyline routes in the candidate set. Thus, the comparison can be done among detected *skyline routes*.

$$\mathcal{T} = \sum_{i=1}^{n-1} tm(s_{i+1}, s_i) + (n-2) \times t_0 \tag{5.3}$$

$$Num = \sum_{i;j(j>i)}^{n} fm(s_i, s_j) \tag{5.4}$$

where t_0 is the average time needed to board at each stop. The time needed to board-on/-off the bus at a stop might increase when the number of passengers of that stop gets high, however, for simplicity, we just set it to a constant (i.e. 1.5 minutes).

Algorithm 5.3 Probability based Spreading

Input: $G(S, E)$: Single directional graph for the given OD pair; *FM*: Flow matrix; *TM*: Travel time matrix

Output: $\mathcal{R}^*$: the set of skyline routes

1: $\mathcal{R} = \emptyset$
2: **Repeat**
3: $currentR = s_1$ //starts from the given origin s_1
4: Choose the next stop s_i^* with respect to $currentR$ according to Eq. 5.5
5: $R = currentR \cdot s_i^*$ //$\cdot$ operation appends s_i to $currentR$
6: **Repeat** Lines 4∼5 **Until** $s_i^* = s_n$ //ends at the the given destination s_n
7: $\mathcal{R} = \mathcal{R} \cup R$
8: Get corresponding *skyline routes* $\mathcal{R}^*$
9: **Until** $\mathcal{R}^*$ keeps unchanged

The key idea of our proposed *probability based spreading algorithm* is to randomly select the next stop among the possible candidate stops in each step, where the candidate stops having high accumulated passenger flow with previous stops are given high probability for random selection. The idea of the proposed *probability based spreading heuristic* is close to the well-known family of heuristics called "Probabilistic Greedy Heuristics" [7, 74]. The difference is that we choose a very specific possible function $P(\cdot)$ which takes the passenger flow accumulation into consideration during the spreading (can be seen in Eq. 5.5). We describe the approach in Algorithm 5.3. The spreading starts from the given source (Line 3). The next stop in the candidate route is chosen based on Eq. 5.5.

$$P(s_i^* | \langle s_1, s_2, \cdots, s_j \rangle) = \frac{\sum_{m=1}^{j} fm(s_m, s_i^*)}{\sum_{i=1}^{|S^*|} \sum_{m=1}^{j} fm(s_m, s_i^*)} \tag{5.5}$$

where $fm(s_m, s_i^*)$ is the passenger flow from s_m to s_i^*, and S^* contains the next possible stops of s_j (child nodes of s_j in the route graph).

We can see the selection of next stop in the candidate route is not only determined by the current stop, but also all the previous stops. The output of this algorithm is one candidate bus route with the number of stops associated with the number of spreading steps. The spreading would be terminated when the given destination is reached (Line 6). For each run, we get either a repeated route or a new route, thus the candidate route set $\mathcal{R}$ would increase as the spreading algorithm is activated. Then a question arises: *how many running times are sufficient to get the best results ?* Based on *Definition* 1 about the skyline routes, we should consider if the skyline route set $\mathcal{R}^*$ remains changed or unchanged.

Theorem 5.4.2 below ensures that when the skyline route set stays unchanged with the increase of spreading algorithm runs, then the best route has been discovered.

Theorem 5.4.2. *$\mathcal{R}_1^*$ and $\mathcal{R}_2^*$ are the detected skyline routes from $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. If $\mathcal{R}_1 \subseteq \mathcal{R}_2$, then we have: $\forall R_i \in \mathcal{R}_1^*$, $\exists R_j \in \mathcal{R}_2^*$; $R_i = R_j$ or R_i is dominated by R_j.*

In Algorithm 5.3, we have $\mathcal{R}_{t_1} \subseteq \mathcal{R}_{t_2}$ if the running time $t_1 < t_2$, and the algorithm would be stopped when no better skyline routes are returned with the increase of running times, that is $\mathcal{R}_{t_1}^* = \mathcal{R}_{t_2}^*$ (Line 9). The computation complexity of the algorithm is $\mathcal{O}(N)$.

Instead of choosing only one stop randomly at each spreading step like in the *probability based spreading algorithm*, an intuitive way is to select top-k stops each time, where those k nodes should have highest accumulated passenger flow with previous stops. In such a way, the first step selects top-k nodes, thus leading to k routes from the origin to those nodes. In the second step, each k nodes would select another top-k nodes, thus the total candidate routes would be k^2. Assume that n steps are needed to the destination, then the total candidate routes generated would be k^n in the end. Thus, the computation complexity of this algorithm is $\mathcal{O}(k^n)$, which grows exponentially with the spreading step (n). We use this **top-k spreading** method as the baseline.

Bi-directional Probability based Spreading (BPS) Algorithm: In practice, for a particular bus line, buses can run on the same route in both directions. Algorithm 5.3 can get the best bus route in one direction (e.g. from ZJU to Railway Station), however, it cannot guarantee the same route in the opposite direction (i.e. from Railway Station to ZJU) would still expect the maximum number of passengers, as the passenger flows in two directions of the route are generally asymmetrical. To get a bus route which has overall maximum expected number of passengers in both directions, we propose the *BPS* algorithm, whose basic idea is to run the *probability based spreading algorithm* in both directions so that we generate one candidate "optimal" route in each direction, and the best route is selected by evaluating all the candidate routes in two directions.

We illustrate the procedure in Algorithm 5.4. The key idea behind is to run Algorithm 5.3 in both directions (Line 3∼4), and generate one candidate route for each direction at each run (Line 5). The skyline routes are selected based on the total travel time and expected number of passengers in both directions of each candidate route (Line 6), and the selection process terminates also when no more better skyline routes can be generated (Line 7).

5.4.4 Bus Route Selection

Given the bus operation frequency (once every 30 minutes), the total travel time constraint, and the taxi passenger flow from 21:30 to 5:30, we obtain the candidate bus routes for a given OD pair using the two different heuristic spreading algorithms, and the skyline route which achieves the maximum expected number of passengers will be selected as the operating route.

Algorithm 5.4 *BPS* Algorithm

Input: $G_{O\to D}(S,E)$: Graph for $O \to D$; $G_{D\to O}(S,E)$: Graph for $D \to O$; *FM*: Flow matrix; *TM*: Travel time matrix

Output: $\mathcal{R}^*$: the set of skyline routes

1: $\mathcal{R} = \emptyset$
2: **Repeat**
3: Run Line 2∼6 in Algorithm 5.3 for $G_{O\to D}(S,E)$, and the output is $R_{O\to D}$
4: Run Line 2∼6 in Algorithm 5.3 for $G_{D\to O}(S,E)$, and the output is $R_{D\to O}$
5: $\mathcal{R} = \mathcal{R} \cup R_{O\to D} \cup R_{D\to O}$
6: Get corresponding *skyline routes* $\mathcal{R}^*$
7: **Until** $\mathcal{R}^*$ keeps unchanged

With the planned bus route consisting of the selected bus stops, the next step is to find a physical bus route in the real setting, which consists of road segments corresponding to the planned route. The selection of each road segment is done by following the dense and fine trajectories of taxis if they allow buses to operate; Otherwise similar bus routes near the planned ones can be adopted as a refined solution.

5.5 Experimental Evaluation

In this section, we validate the proposed approach with a large-scale real-world taxi GPS dataset which is generated from 7,600 taxis in a large city in China (Hangzhou) in one month, with more than 1.57 million of night passenger-delivering trips. All the experiments are run in Matlab on an Intel Xeon W3500 PC with 12-GB RAM running Windows 7.

5.5.1 Evaluation on Bus Stops

We compare the bus stop results generated with our proposed method with that generated by the popular k-means clustering method. We set $k = 579$, which is the same as our method. We adopt the Eulerian distance as the similarity metric. The centroid of each cluster is selected as the stop. Figure 5.9 shows the comparison results. Comparing with the popular k-means approach, our proposed candidate bus stop identification method has at least the following two advantages:

1. The centroid of each cluster got by k-means is the average location of all its members, and it may fall into non-reachable places like river, as highlighted by the black circles in Figure 5.9 (left). In our proposed method, both hotness and connectivity of each grid cell is considered for the bus stop location selection, and the selected bus stops are meaningful and stoppable places;

2. Several identified stops by k-means fall into a small area (highlighted by the blue circle) as the size of clusters got by k-means is very different, while our proposed method generates candidate bus stops that are evenly distributed in the hot areas, which better meets the commonsense design criteria of bus stops.

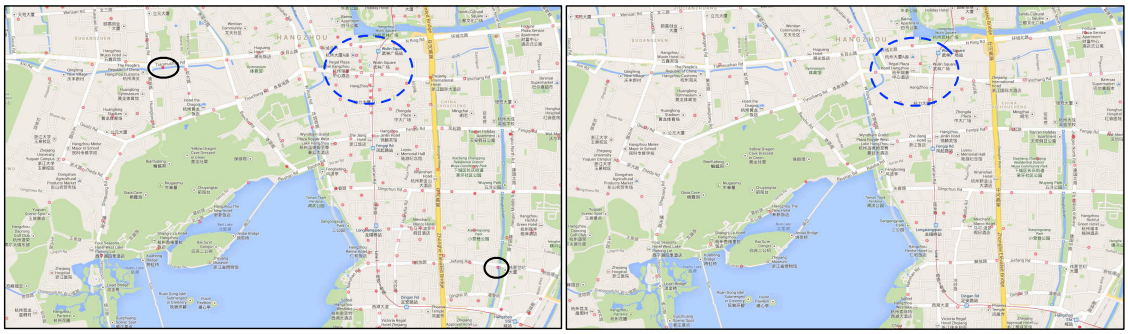


Figure 5.9: Comparison results with k-means (best viewed in the digital version). Results got by k-means (left) and results got by our method (right).

5.5.2 Evaluation on Bus Route Selection Algorithm

We first show the convergence of the proposed algorithm, and followed by a parameter sensitivity study. Then we perform a quantitative statistical analysis of all the candidate routes generated for three given OD pairs. We also give the computed *skyline route* results. Finally, we validate that our proposed bus route generation approach outperforms the baseline approach. Table 5.1 shows the details of three OD pairs for night-bus route design experiment, where more than 70 candidate bus stops are in the candidate bus route selection list.

Table 5.1: Detailed information about studied OD pairs.

	OD Pairs	Distance (km)	Number of Stops
1	ZJU - Railway	5.70	104
2	Railway - East Railway	5.86	75
3	East Railway - ZJU	8.80	144

5.5.2.1 Convergence Study

As illustrated in Algorithm 5.3 and 5.4, our proposed bus route generation process would be terminated if the resulted *skyline routes* keep unchanged. We study the similarity of consecutively generated *skyline routes* from 5,000 to 150,000 runs, with a constant interval

of 5,000 runs. We measure the similarity (sim) of two sets A and B as follows:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.6}$$

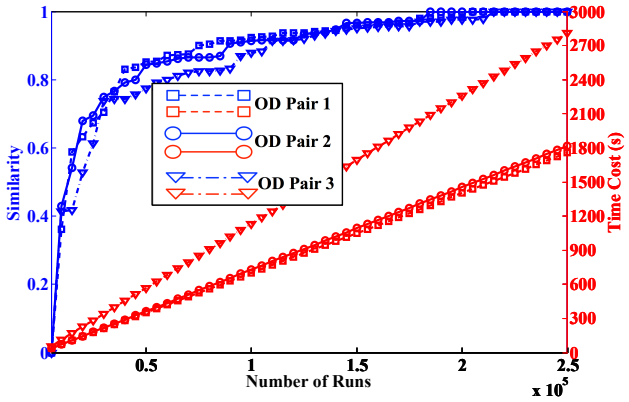


Figure 5.10: Convergence study of the proposed *BPS* algorithm.

The similarity results of the consecutively generated skyline routes with a 5,000 run interval are shown in Figure 5.10, and the time cost is put in the diagram as well. In this study, we can see that *sim* values gradually reach 1 with the increase of runs for all three OD pairs, meaning that in all three cases the best bus route converges to one. Also the time cost is almost linearly increased with the number of runs, suggesting that the spreading time cost at each run is almost constant. It is also noted that the three curves for three OD pairs have different slopes, the reason is probably because the bus routes corresponding to different ODs have different lengths and varied number of candidate bus stops, thus the spreading time and candidate bus stop selection time should be also different.

5.5.2.2 Parameter Sensitivity Study

To better understand the bus stop identification and the bus route selection algorithms, we conduct experiments under different parameter settings to study how they affect the number of expected passengers of selected routes and running time. We examine three parameters in the process, while two of them are in the bus stop identification phase, the remaining one is in the route graph building algorithm.

Varying parameters (Th_1 and Th_2) for the cluster merge and split algorithms: As discussed in Section 5.3, a bigger Th_1 would produce more large clusters, and likewise, a bigger Th_2 would also generate more large clusters. Figure 5.11 shows the *Cumulative Distribution Function* (CDF) of finally produced clusters in terms of size after cluster merging and splitting under various Th_1($\in$[100:50:300] meters) and Th_2 ($\in$[400:50:650]

meters) respectively. We also show the skyline route results under different Th_1 and Th_2 in Figure 5.12. From these results, we can see that choosing the relatively smaller Th_1 and larger Th_2 will lead to better skyline routes.

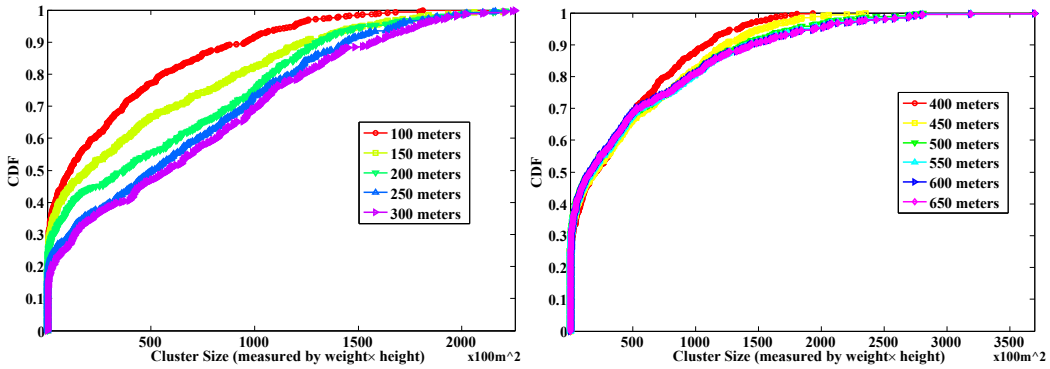


Figure 5.11: CDF results of cluster size under different Th_1 ($Th_2 = 500\ m$ (left) and under different Th_2 ($Th_1 = 150\ m$) (right).

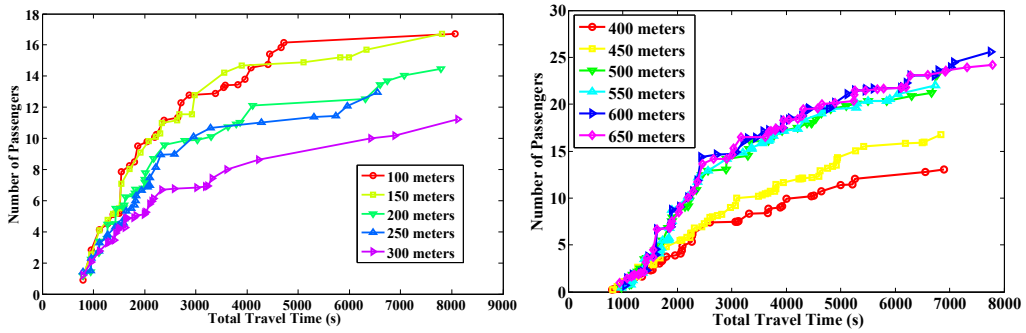


Figure 5.12: Skyline route results under different Th_1 ($Th_2 = 450\ m$) (left) and under different Th_2 ($Th_1 = 150\ m$) (right).

Figure 5.13 shows the maximum number of expected passengers for the selected bus route and the time cost, respectively, under different Th_1 and Th_2 combinations. Note that the time cost is the total cost of the candidate bus stop identification phase and the bus route selection phase. We also find that combinations of bigger Th_1 and smaller Th_2 are not good as they often result in lower number of passengers but higher time cost. Specifically, the minimum number of passengers and the maximum time cost occurs at $Th_1 = 300\ m$ and $Th_2 = 400\ m$. This is probably because: for the candidate bus stop identification phase (i.e. Phase 1), a bigger Th_1 would first generate more large clusters in the cluster merging procedure, then a smaller Th_2 would require more spitting operation times during the cluster splitting, at last more number of small-size clusters would be identified; for

the bus route selection phase (i.e. Phase 2), the route graph would become more complex with the increase of the number of candidate bus stops, and meanwhile, the number of passengers decreases as the walkable distance is set short. Finally, we choose $Th_1 = 150\ m$ and $Th_2 = 500\ m$ throughout the paper as it expects larger number of passengers while consuming relatively less time. Additionally, 500-meter distance is an acceptable walk distance for passengers.

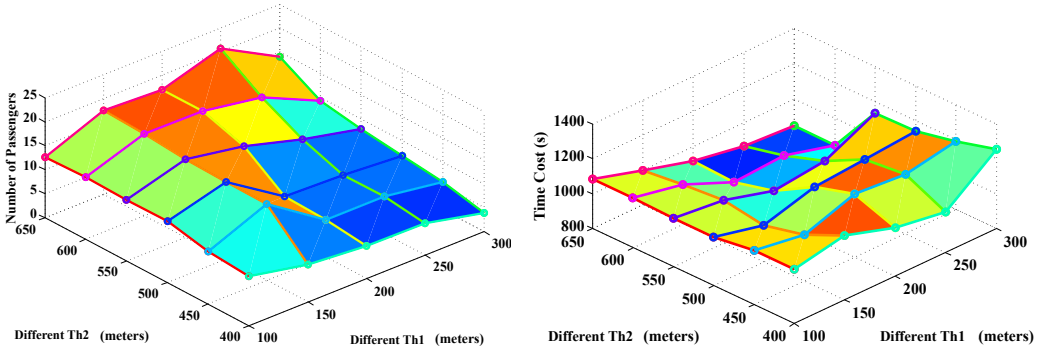


Figure 5.13: The maximum number of passengers under different Th_1 and Th_2 combinations (left); Time cost under different Th_1 and Th_2 combinations (right).

Varying the parameter δ for the graph building algorithm: Here, we study the impact of δ selection on the expected number of passengers of the selected bus route and time cost. For a particular stop s_i, larger δ would lead to more child nodes. Mathematically, we have: $\forall s_i \in S$, $S'_{\delta_1}(s_i) \subseteq S'_{\delta_2}(s_i)$ if $\delta_1 \leq \delta_2$, where $S'(s_i)$ is the child node of s_i in the route graph. And we also have $\mathcal{R}_{\delta_1} \subseteq \mathcal{R}_{\delta_2}$. Therefore, with the increase of δ value, better route can be obtained. Meanwhile, the route graph would become more complex, resulting in the increase of computation time.

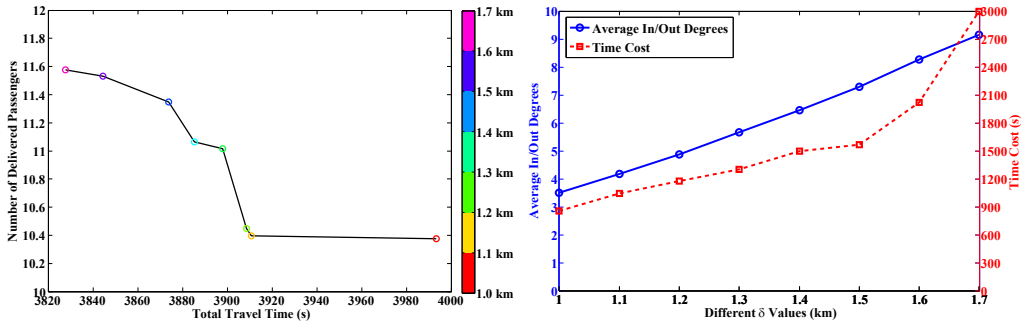


Figure 5.14: Selected bus routes at different δ (left); The route graph complexity and time cost under different δ (right).

We investigate different δ in the range of [1.0 km, 1.7 km] for OD pair 2, with a constant

interval of 0.1 km. The left figure in Figure 5.14 shows two metrics of the selected bus route under different δ values. One point on the plane stands for the selected route under a given δ. We can see that the selected route becomes steadily better with the increase of δ (deliver more passengers with less travel time). However, the difference is negligible after $\delta \geq 1.5$. We also show the complexity of the route graph and the time cost under different δ values in the right figure in Figure 5.14. The complexity of graph is simply quantified by the average In-coming/Out-going degrees. They are equal to the ratio of the total number of edges to the total number of nodes in the route graph. From the figure, we can see that the average In-coming/Out-going degrees under 1.7 km is twice more than that under 1.0 km. Furthermore, more computation time is needed when δ increases, because the route graph becomes more complex. We set $\delta = 1.5$ km throughout the paper as it leads to good performance with low time cost.

5.5.2.3 Candidate Routes Statistics

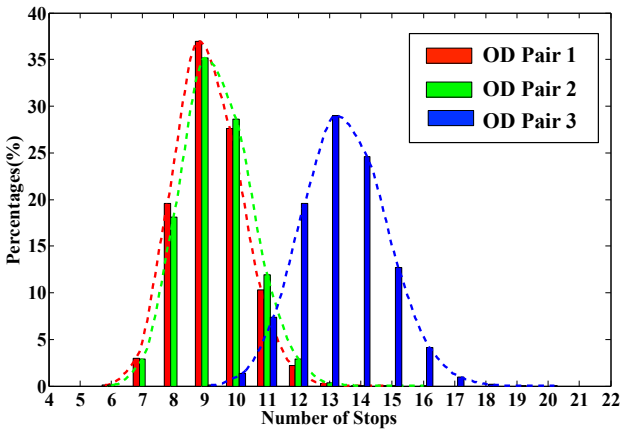


Figure 5.15: The number of stops of candidate route stops statistics for 3 OD pairs.

Figure 5.15 shows the statistical information about the number of stops of candidate routes. Several interesting observations can be obtained:

1. For OD pair 1, routes with 8∼10 stops take up over 80% of the cases (both origin and destination are included). Few routes can reach the destination by traversing only 4 stops, or passing more than 11 stops.
2. For OD pair 2, over 60% of the routes contain 9 or 10 stops. Similar to the case of OD pair 1, some routes can reach the destination by passing 4 stops.
3. For OD pair 3, most of the routes contain 10 to 18 stops due to the longer OD distance, and almost half of the routes include 13 or 14 stops.

4. The statistical results comply with the intuition that the longer distance of a given OD pair, the more stops the route would contain.

We also provide the statistics of the total travel time of candidate routes having the same number of stops (mean and standard deviation), which is shown in Figure 5.16. We can see that, for all three OD pairs, the average total travel time almost increases linearly with the number of stops, suggesting the total travel time constraint is related to the constraint of the total number of stops.

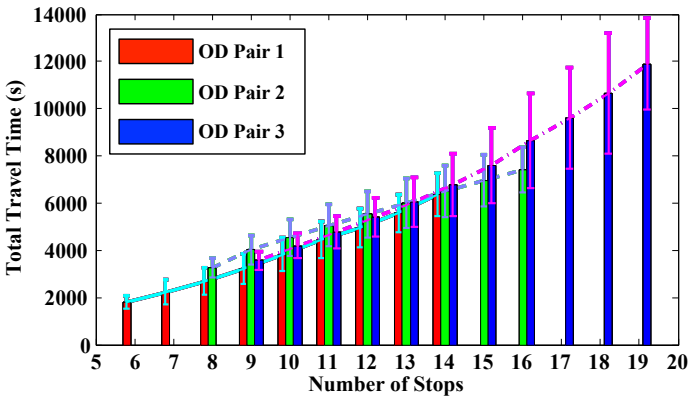


Figure 5.16: The relationship between the number of stops and total travel time statistics for 3 OD pairs.

5.5.2.4 Skyline Routes

We show the *skyline routes* for the OD pair 3 in Figure 5.17. Each point in the plane represents a candidate route. The x-axis stands for the total travel time of candidate route, while the y-axis represents the expected number of passengers. From Figure 5.17, we can see that the *skyline routes* are connected to form a curve above all the points representing common routes, and over 99% of the routes are dominated by the few skyline routes. Specifically, we get 36 skyline routes across all the travel time frames, out of hundreds of thousands of routes for the case of OD pair 3. Similar phenomena have been observed for other two cases as well.

5.5.2.5 Comparison with top-k spreading algorithm

In the top-k spreading algorithm, the selection of k is vital to the skyline routes generated as well as the time needed to generate all the candidate routes. In particular, when $k_1 < k_2$, we have $\mathcal{R}_{k_1} \subseteq \mathcal{R}_{k_2} (k_1 \leq k_2)$. Theorem 5.4.2 guarantees that a bigger k would lead to a better set of skyline routes. However, the greater k also results in significant

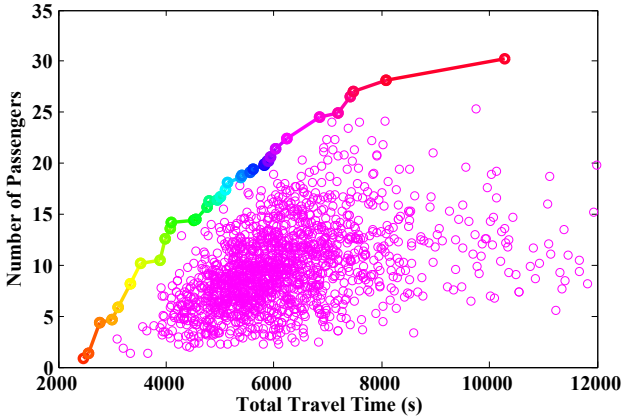


Figure 5.17: Detected *skyline routes* and other candidate routes.

increase of time cost. We compare the *skyline routes* generated from the *BPS* method with that from the *top-k spreading* method with different k values for the case of OD pair 1, which is shown in Figure 5.18. We can see that the *BPS* approach outperforms the *top-k algorithms* even when k is set to 5. Again, similar conclusion can be also drawn for the other two OD pairs.

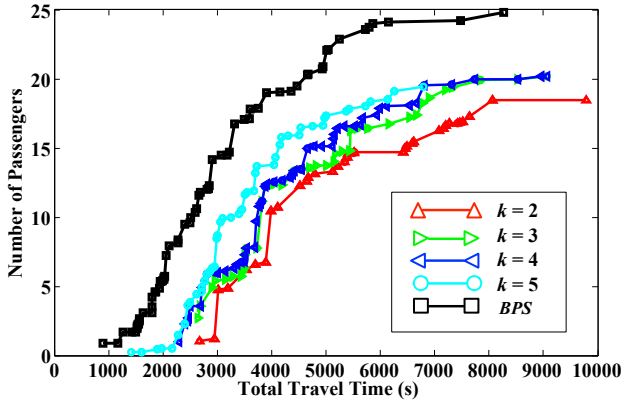


Figure 5.18: Comparison results with baseline under different k values.

5.5.3 Bidirectional vs Single Directional Bus Route

In real life, bus route got by Algorithm 5.3 may 1) be the *skyline route* in both directions; 2) be the *skyline route* in only one direction; 3) not be the *skyline route* in any direction. It is noteworthy to compare the overall best bidirectional bus route obtained by Algorithm 5.4 to the best routes in single direction. We have drawn all the seleted bus routes on the city digital map in Figure 5.19 for the OD pair 3. They are different routes, which means

the bidirectional bus route is neither the *skyline route* in the ZJU→East Railway Station direction, nor in the East Railway Station→ZJU direction. A reasonable explanation is that the passenger flow and the travel time among stops is often asymmetrical, and thus the bus route which carries the maximum number of passengers under the given time constraints in one direction would probably fail to deliver the same performance in the opposite direction. However, they all have 13 stops in total and share several common stops near the ZJU stop, especially for the route $R_{O\to D}$ (left figure in Figure 5.19) and $R_{O\leftrightarrow D}$ (bottom figure in Figure 5.19). By further checking, we find that these common stops are popular night life centers.

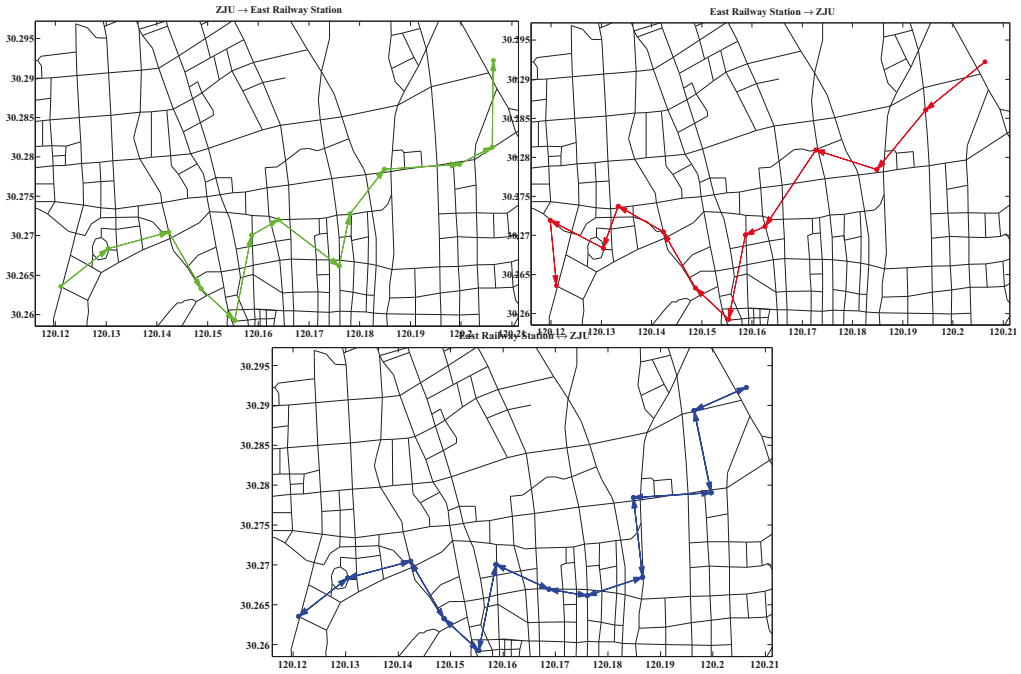


Figure 5.19: Comparison results of the selected bus routes in two directions to that in one direction. $R_{O\to D}$ (top left); $R_{D\to O}$ (top right); $R_{O\leftrightarrow D}$ (bottom).

We show the average travel time and the number of expected delivered passengers of these three bus routes in Table 5.2, and note that heavier passenger flow can be found from East Railway Station to ZJU direction ($R_{D\to O}$). While $R_{D\to O}$ takes slightly less time and delivers a larger number of passengers than $R_{O\to D}$, it carries about 48 more passengers on average per night. $R_{O\leftrightarrow D}$, however, takes the least time, and the average number of delivered passengers lies between $R_{O\to D}$ and $R_{D\to O}$.

Table 5.2: Two metrics of the selected bus routes.

	Direction	Average Travel Time (in second)	Number of Passengers
$R_{O\to D}$	ZJU→East Railway	5406.7	17.25
$R_{D\to O}$	East Railway→ZJU	5352.2	20.31
$R_{O\leftrightarrow D}$	ZJU↔East Railway	5320.2	18.73

5.5.4 Comparison with Real Routes and Impacts on Taxi Services

As the taxi GPS dataset we have was collected from April 2009 to March 2010, we are very interested in knowing if there was any new night-bus route created during this year and how the planned bus route generated with our approach compares with the manually created route. Fortunately we were told that a night-bus route was created in February 2010. We could access all the taxi passenger flows before and after the route started date. It is noted that the route is designed by local experts and the user demands are obtained from expensive human survey. We first draw the newly started night-bus route R_3 on Google map as shown in Figure 5.20 (left bottom), then we draw our proposed night-bus route R_1 in Figure 5.20 (left top). Through comparison we see that they are quite different. With the newly started route, we decide to take a similar route in our selected candidate bus routes (not the best one), and we find R_2 as shown in Figure 5.20 (left top). It is noted that the main difference between R_2 and the newly started route R_3 is that R_2 includes an additional Stop J in the route. By comparing the passenger flow in segment I↔K with that in segment J↔K at different time slots, it is found that the passenger flow in path J↔K is even greater than I↔K in the first two time slots, as shown in Figure 5.20 (right top). Considering further the accumulation effects, including Stop J in the bus route would significantly increase the expected number of passengers along the route. This is evidenced by Figure 5.20 (bottom right). The *accumulated effect* is more remarkable at the first three frequencies. Thus, our candidate bus route R_2 would outperform the newly added bus route R_3, at the cost of adding one more bus stop and more travel time.

We also compare our proposed best route R_1 with the candidate route R_2. The difference between R_1 and R_2 lies in two different paths taken from C to H. While R_2 passes the famous shopping street (Yan'an Road) in Hangzhou ($C \leftrightarrow E \leftrightarrow F \leftrightarrow H$), R_1 traverses the famous night-club areas along the West Lake. If we compare the number of passengers in R_1 and R_2, it can be seen from Figure 5.20 (right bottom) that the passenger flow of R_2 is heavier than that of R_1 only around 22:00, and it is much lighter soon after 23:00. With the rest of the stops being the same for both R_1 and R_2, there is no doubt about why R_1 has been selected as the best night-bus route. If we take a closer look at R_1, R_2, and

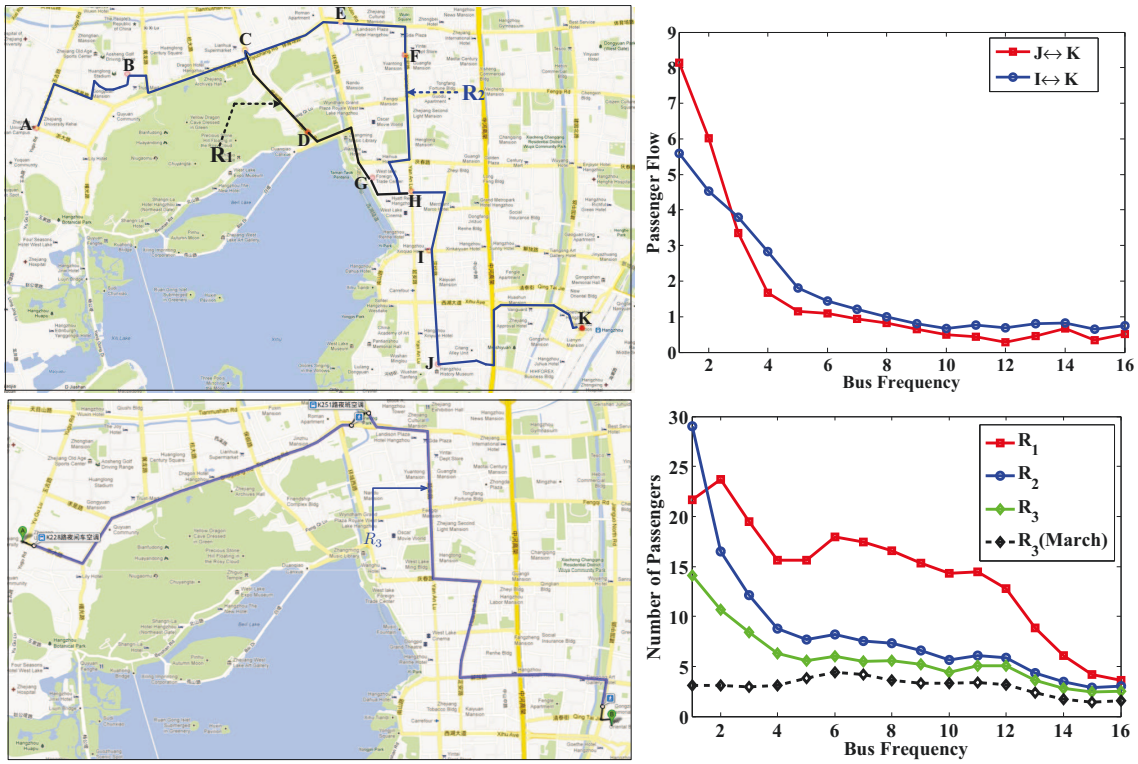


Figure 5.20: Results comparison. Planned routes (top left); Passenger flow comparison of two segments at different frequency (top right); Opened night-bus route (bottom left); Number of delivered passengers at different frequency (R_1, R_2 and R_3, bottom right).

the newly started route R_3, as R_1 takes a much shorter route than R_2 and needs similar travel time as the newly started route R_3 does (shown in Table 5.3), but R_1 expects much more passengers than R_2 and the newly started route R_3, thus it is reasonable to conclude that the selected night-bus route with our proposed approach is better than the current route-in-service in terms of travel time as well as expected number of passengers.

Table 5.3: Total travel time of the bus routes.

Bus Routes	Total Travel Time (in second)
R_1	3583.8
R_2	4664.9
R_3	3624.0

It is understood that introducing of new public services (i.e. new Metro/bus lines) would affect taxi services in the city [3]. It is interesting to compare the taxi passenger flow

change along the new bus route before/after it was opened. We choose the new night bus route (R_3) opened in February, 2010 for this study. We prepare taxi GPS data collected in January and March, 2010, and calculate the corresponding taxi passenger flow along the new bus route across all bus frequencies, which is shown in the right bottom subfigure of Figure 5.20. We can see that the number of passengers who travel by taxi along the bus route in March is much smaller but quite stable across all the bus frequencies. This may be interpreted by the fact that while some passengers might switch to public services, a certain number of passengers still prefer to take taxis at night.

5.5.5 Bus Capacity Analysis

After selecting the best bus route for operation, the next important thing is to determine the proper bus capacity to save operation cost. The essence for bus capacity estimation is to determine the maximum number of passengers on the bus across all the frequencies. For the bus route R_1 of OD pair 1, Figure 5.21 shows the number of passengers on the bus across all the frequencies for both directions. As can be seen from the results, choosing buses with 20 seats could well meet the requirements. Besides, we also have the following three observations:

1. More passengers are often expected in both directions for the first operation frequency, except for the 11th and 12th frequencies when the bus runs from C to D.
2. Buses running close to the capacity only last for 3 stops (from A to K) or 4 stops (from K to A).
3. Night buses heading towards different directions have quite different passenger flow patterns.

5.6 Concluding Remarks

In this work, we have investigated the problem of bi-directional night-bus route design by leveraging the taxi GPS traces. The work is motivated by the needs of applying pervasive sensing, communication and computing technology for sustainable city development. To solve the problem, we propose a two-phase approach for night-bus route planning. In the first phase, we develop a process to cluster "hot" areas with dense passenger pick-up/drop-off, and then propose effective methods to split big "hot" areas into clusters and identify a location in cluster as the candidate bus stop. In the second phase, given the bus route origin, destination, candidate bus stops as well as bus operation frequency and maximum total travel time, we derive several criteria to build bus route graph and prune the invalid stops and edges iteratively. Based on the graph, we further develop two heuristic algorithms

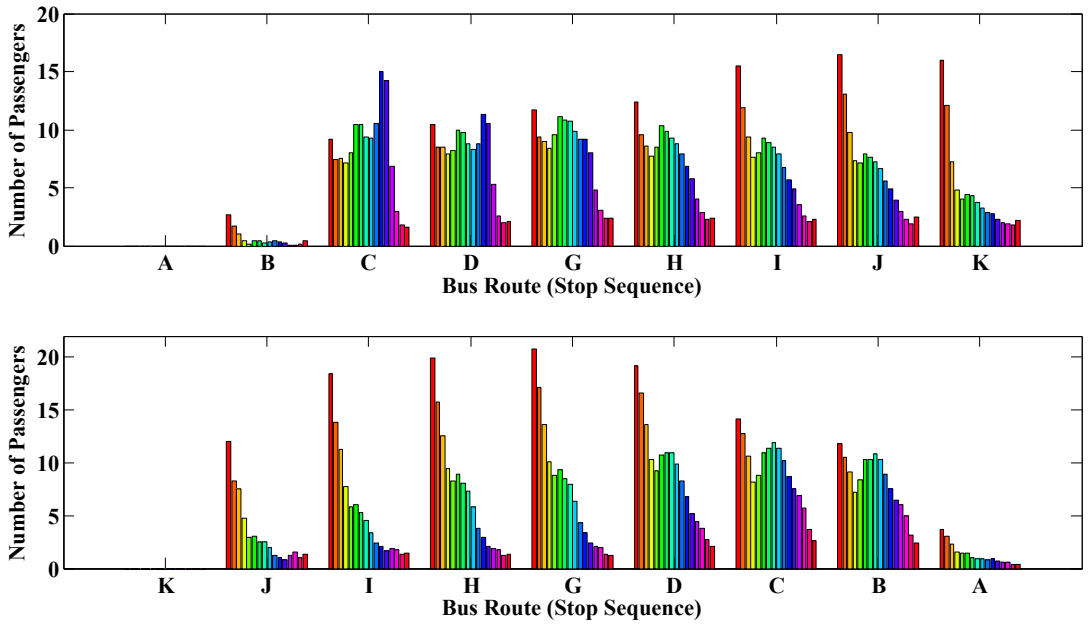


Figure 5.21: The number of passengers on the bus before reaching the stop for OD pair 1.

to automatically generate candidate bus routes in both directions, and finally we select the best route which expects the maximum number of passengers under the given conditions. On a real-world dataset which contains more than 1.57 million passenger delivery trips, we compare our proposed candidate bus stop identification method with the popular k-means clustering method and show that our method can generate more reasonable and meaningful results. We further extensively evaluate our proposed *BPS* algorithm for automatic bus route generation and validate its effectiveness as well as its superior performance over the heuristic top-k spreading algorithm. Further more, we show the selected night-bus route with our proposed approach is better than a newly started night-bus route-in-service in Hangzhou, China.

For this work, we consider the effective design of only one bus route. In the future, we plan to broaden and deepen this work in several directions. First, we attempt to investigate the optimal bus route design with more real-life assumptions. For example, for the bus stop identification, the grid cells in geographical proximity might not be walkable due to physical barriers; for bi-directional bus route selection, one-way routes should be excluded or changed in actual design; Second, we also plan to explore the issue of designing more than one night-bus route in an optimal way; Third, we would like to develop practical systems leveraging on taxi GPS traces, enabling a series of pervasive smart transportation services.

Chapter 6

TRIPPLANNER: Personalized and Traffic-aware Trip Planning

Contents

6.1 Introduction

Planning an itinerary is one of the most important and time-consuming travel preparation activities [41, 134]. In order to plan a trip for visiting a popular tourist city, one needs to select a number of preferred Point of Interests (POIs) among hundreds of possible venues[1], figure out the order in which they are to be visited, ensure the time it takes to visit each POI and to transit from one POI to the next, and meet one's time budget. Let us take the following use case as an example:

John is transiting through San Francisco. He rents a car at the SFO airport at 9:00 am and would like to spend several hours for sightseeing, and then leaves for San Jose by train at 15:00 pm from the Caltrain Station. He wants to visit the Golden Gate Bridge, Lombard Street and Fisherman's Wharf. If time permits, he also wants to squeeze in visits to an art museum and/or one of the Boudin Bakery locations for lunch. In addition, he also prefers to having lunch before visiting the Fisherman's Wharf.

As shown in the above use case, three main factors have to be considered in the design of a trip planning system: 1) *the venue constraints*, which include the trip starting location (the Airport), the trip ending location (the Caltrain Station), the POIs expected to be covered in the itinerary (e.g. the Golden Gate Bridge), the POI categories which might be added if time permits (e.g. art museum), and the POI visiting order; 2) *the time constraints*, which include a trip starting and ending time (time budget), the duration of visit time for each POI which can be estimated and controlled by users, the transit (driving) time between POIs which varies depending on the traffic condition of the time of the day, and the proper time of visiting a certain POI which is determined by the operation time of the POI; and 3) *user's preference scores* about a specific POI and an itinerary at certain time of the day which are assumed to be computable. The *objective* of the trip planning system is to interact with users to inform if the user-specified POIs can all be covered in one recommended route within the time budget. If the answer is "no", the system would iteratively prompt the user to remove one POI at a time until the POIs specified can be fit into one route without compromising the time constraint. If the answer is "yes", the system would automatically generate an "optimal route" which contains the specified POIs and preferred POI categories, and meets the time constraint according to the predicted driving time of the day.

Apparently, the above problem cannot be solved using the approaches proposed for route search in the previous research [25, 27, 81, 146], as they often assume that the transit time between POIs is constant. In our scenario, the purpose of route search is to find a route that can cover a series of requested POIs specified by users while meeting a time budget.

1. We use *venue* and *POI* interchangeably throughout this chapter.

The above issue is also different from route recommendation. Many route recommendation systems suggest routes directly based on the similarity between user's visiting history in other contexts and other people's trip records in the targeted city [166]. Others identify venues according to a user's preference and recommend routes based on certain criteria [62, 103]. Another group of route planning work aims to find the fastest or shortest paths in road networks based on the time-varying assumption of each road segment [150]. These studies care only about the edge information in the network, ignoring totally the attributes associated with the nodes (POIs). Unlike this body of work, we need to consider the characteristics of each POI in the route selection process, e.g. its attractiveness, operation hours, and order of visit. In summary, this study intends to build a *personalized, interactive and traffic-aware* trip planning service.

In order to achieve personalization in trip planning, we first need to acquire the information about the POIs and links among them to build a POI network model. So far, different data sources have been exploited, including: 1) websites, Wikipedia, web blogs which contain tourists' profiles as well as comments that reveal preferences and experiences with POIs [16, 23]; 2) social media sites such as Facebook, Flickr, and LBSN (e.g. Foursquare and Gowalla), which can inform the popularity, functions, operating hours of the POIs as well as individual user's travel history [27, 76, 103]; and 3) GPS trajectories of people and taxis, which can indicate the stay time in each place and transit time between two places [12, 150, 166]. Apparently, each data source has its strength and weakness in characterizing certain facets of the POI nodes and edges required by the model. Integrating heterogeneous data sources can provide a more complete picture of the POI network.

In this chapter, we develop a novel trip planning framework called TripPlanner. In the front end, TripPlanner allows users to interactively specify their venues of interests with varied constraints. In the back end, it leverages heterogeneous crowdsourced digital footprints for POI network model construction. Through a two-phase query resolution process, TripPlanner could recommend to the user a personalized route with the highest trip score under the total travel time constraint. In summary, the main contributions of this study are:

— First, we define an under-explored trip planning problem, which allows users to specify not only the must-visit venues but also optional venue categories if the time permits, given a total travel time budget. We further make more realistic assumption about the transit time between venues that varies with time of the day and day of the week. In other words, the total travel time of the same route may be different.

— Second, we attempt to construct a dynamic POI network model of a city, leveraging *heterogeneous crowdsourced digital footprints* (i.e. Foursquare check-ins and taxi GPS traces) to better utilize the strengths of each data source in characterizing the

attributes of the nodes and links of the POI network.

— Third, we propose a *two-phase approach* for *personalized, interactive and traffic-aware* trip planning. We also propose a new way to score an itinerary, considering both the popularity and individual preference of venues. Specifically, in the *route search phase*, the system works *interactively* with users to generate candidate routes with *specified venues*; In the *route augmentation phase*, the system employs heuristic algorithms to add *user-preferred venues* (i.e. optional venues if time permits) to the candidate routes iteratively, with the objective of maximizing the route score and satisfying both the venue visiting time and total travel time constraints.

— Finally, we validate the efficiency and effectiveness of TripPlanner by extensive evaluations using large-scale real-world data sets. Through a case study of planning three trips with different starting time and user-preferences, it is shown that TripPlanner can recommend appropriate routes which fully meet user's requirements yet take into consideration the traffic condition along the chosen routes at the specified time.

The remaining of this chapter is structured as follows. In Section 6.2, we first review the related work and show show our work is different from prior work. In Section 6.3, we introduce the framework of our proposed TripPlanner system. After presenting the process of constructing the POI network by leveraging the Foursquare check-in and taxi GPS data sets in Section 6.4, we elaborate on our two-phase approach in Section 6.5. Extensive evaluation results are reported in Section 6.6 to verify the effectiveness of the proposed approach. Finally, we conclude the paper and chart the future directions in Section 6.7.

6.2 Related Work

The related work is organized in two subsections. We first review previous work on extracting information from different data sources to build the POI network model, and then discuss about how to recommend a trip to users based on certain assumptions.

6.2.1 Construction of POI Network

In trip planning research and applications, people have exploited different data sources to extract node and edge information needed to build a POI network model. For example, in [10, 23, 27, 41, 76, 104, 167], many researchers have used geo-tagged photos from photo-sharing sites (e.g. Flickr) to *derive* the information about POIs, such as locations, popularity, characteristics, and proper visiting time and order. In addition, demographics

and social relationships of visitors to these POIs can be extracted. However, it is hard to estimate the dynamic transit time between POIs from social media data. More recently, people began to explore user-generated LBSN digital traces since such data contains rich information that can be used to *directly* characterize each POI in a tourist city and users' preferences to each POI [13, 62, 92, 103, 147]. Unfortunately, similar to geo-tagged photo data, LBSN traces also do not contain dynamic transit time between POIs, especially when driving is considered for travelling in a city. Another popular type of data is GPS trajectory, which can be used to predict the fastest route at certain time of the day in a city [150]. Previous studies have shown that GPS trajectory traces can precisely characterize the transit time between POIs, which is more accurate than Google Maps[2] results [12, 31, 67]; the point-to-point transit time estimated by Google Maps was about 35% off from the actual values on average [12].

Building on existing work, we leverage taxi GPS trajectory and LBSN trace data to construct a POI network model. Such approach allows us to better characterize both the POI nodes and the edges in the network, making it possible to address a more realistic trip planning problem and design a better trip planning system.

6.2.2 Trip Planning

There has been quite some work on trip planning [128], which can be roughly classified into three categories. The first category is *route search*, which aims to answer a user's route queries over a given POI network. *Traveling Salesman Problem* (TSP) is a classical problem on route search [90]. Given a specified set of POIs in a graph and their pairwise distances, the goal of TSP is to find the shortest route that visits each POI exactly once and returns to the original location. However, situations may be much more complicated in the real world. Destination of a trip may be different from the starting point. Furthermore, users may simply have in mind a type of POIs of interests rather than a specific POI location. *Trip Planning Query* (TPQ) is proposed to address the problem [81]. The goal of TPQ is to find the shortest path between two given locations that covers all of the user-specified node categories. Some research has looked into variations of TSP and TPQ problems with additional constraints [27, 72, 126, 146], but most of studies assume that the transit time/distance between POIs is constant, except for few papers [58, 60, 79]. Different from prior work, we allow both POIs and POI categories (i.e. types) to be specified in the route query. We also assume that the transit time between POIs is time-dependent according to the traffic conditions.

The second category is *route recommendation* which usually suggests POIs or routes to users based on the user preferences. It usually assumes that users will not provide

2. http://maps.google.com

Table 6.1: A brief comparison between different work and ours.

	Paper	User-preferences	Node constraint	Edge constraint	Budget constraint
Route Search	[90]	×	S, E, Specified POIs	Static	×
	[81]	×	S, E, POI categories	Static	×
	[27]	×	S, E, POI categories	Static	Total time
	[72]	×	S, E, POI categories	Static	×
	[126]	×	S, POI categories	Static	×
	[146]	×	POI categories	Static	×
	[58, 79]	×	S, E, POI categories	Traffic-aware	×
Route Recommendation	[62, 76]	✓	S, E	Static	Total time
	[103]	✓	S, E	Static	Total money
Route Planning	[150, 168]	×	S, E	Traffic-aware	×
Ours		✓	S, E, Specified POIs and categories	Traffic-aware	Total time

[1] × denotes the respected factor is NOT considered; ✓ denotes the respected factor is considered;
[2] S is short for the starting place; E is short for the ending place;
[3] Some papers list have some additional node constraints, such as POI visiting order, POI visiting time. For instance, [62] has a visiting time constraint for the POIs. [72, 126] impose a visiting order constraint for the POIs.

the POIs or POI categories explicitly. For instance, Kurashima et al. develop a probabilistic model which incorporates user preferences, location and available time to suggest personalized routes [76]. Lu et al. present a *Personalized Trip Recommendation* (PTR) framework to recommend personalized venue sequences within a predefined budget (e.g. time, money) [103]. Hsieh et al. propose to utilize users' check-in patterns to recommend time-sensitive popular trips to users [62]. Different from these studies, we already have the POIs and/or POI categories specified in the route query, on top of which we employ users' preferences to estimate the venue score.

The third category is *route planning* with the goal of selecting optimal time-dependent routes. For instance, Yuan et. al. [150] and Ziebart et al. [168] propose to mine the historical taxi GPS traces to provide optimal driving directions between two chosen POIs, assuming that the transit time is affected by different traffic conditions. Unlike this category of research, we also consider the priority of each POI, preferred order of visit, as well as the visiting time constraint of each POI in the route optimization process.

A comparison between our work and existing research is further provided in Table 6.1. In summary, our work differs from the previous work in the *data sources* used, the *problem* defined, the *assumptions* given, as well as in the *methods* developed.

6.3 TripPlanner System

Here, we first introduce several key terminologies. Then, we formally define the research problem of personalized trip planning. Finally, we give a detailed description of the framework of TripPlanner system, which is comprised of three major parts: a dynamic

POI network model, a route search component, and a route augmentation component (Figure 6.1).

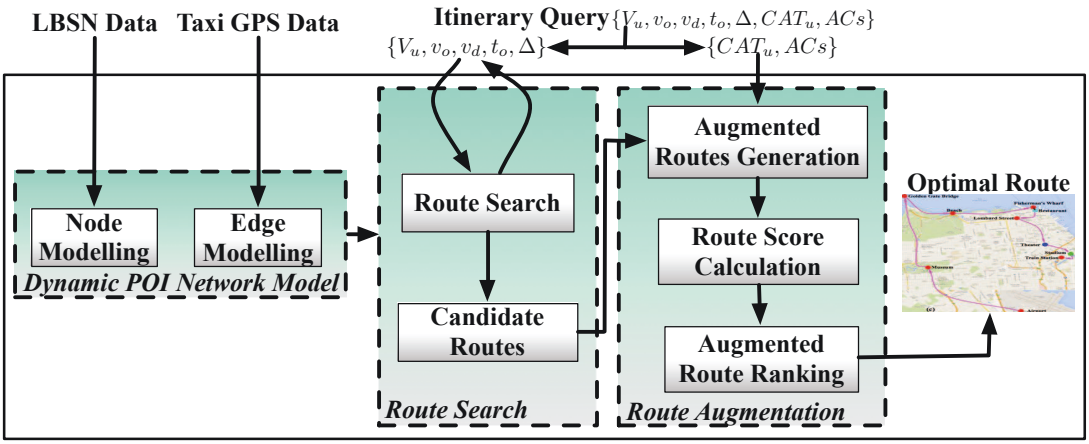


Figure 6.1: The framework of our proposed TripPlanner.

6.3.1 Key Terminologies

Dynamic POI Network Model: The model can be represented by a *directed complete* graph $G = (V, E)$. Each node in V denotes a venue (i.e. POI), which has five attributes: *category, operation time, popularity, geographical location, and stay time (i.e. the duration of visit)*. Each directed edge (v_i, v_j) in E represents a link from node v_i to v_j, which carries the transit time between two venues, denoted as $tt(v_i, v_j)$. The transit time is asymmetric and dynamically changing.

Lemma 6.3.1 (Dynamic POI network has First-Input-First-Output Property)**.** *Given a dynamic network $G = (V, E)$, where the transit time of each edge in G is time-dependent. The network is FIFO since for any arc (i, j) in E, given user A leaves node v_i at time t_0, and user B leaves node v_i at time t_1 $(t_1 > t_0)$, then user B cannot arrive at node v_j before user A.*

Proof. Proof can be found in Appendix A.1. □

Itinerary Query: An itinerary query IQ consists of four parts: 1) a user-specified venue list V_u, that the user intends to cover; 2) starting place v_o and starting time t_o, ending place v_d, and a travel time budget Δ; 3) a set of user-preferred venue categories CAT_u (optional venues to visit if time permits); and 4) additional constraints ACs, such as constraints on the time and the order of venues are to be visited. For instance, a user may want to have lunch at noon and visit museums after that. In summary, the query IQ can

thus be represented as $\{V_u, v_o, v_d, t_o, \Delta, CAT_u, ACs\}$. It should be noted that users may not impose ACs when planning visit, and thus the corresponding field is empty.

Valid Route: A route $R = \langle v_1, v_2 \cdots, v_n \rangle$ is *valid* iif

$$aT(v_i) \geq oT(v_i), lT(v_i) \leq cT(v_i) \quad \forall i \in \{1, 2, \cdots, n\}$$

This implies that the user should visit all venues while they are open. Here $aT(\cdot), lT(\cdot)$ are the users' arriving and leaving time for the given venue, while $oT(\cdot), cT(\cdot)$ refer to the opening time and closing time of the given venue respectively.

Route Score: Route score is defined as the sum of scores of all venues along the route if it is *valid*; otherwise, the route score is defined as 0 (i.e. there exists case in which a user arrives at at least one venue along the route before it opens or after it closes).

Time Margin: It is defined as the difference between the total travel time of the route and the user's time budget.

6.3.2 Problem Statement

Personalized Trip Planning Problem. Given a *dynamic POI network* G in a targeted city and a user's *itinerary query* IQ, our objective is to find the *optimal valid* route with the maximum route score value.

6.3.3 Framework

As shown in Figure 6.1, the proposed framework contains three components: the dynamic POI network model, the route search, and the route augmentation components. While the dynamic POI network model is pre-built and maintained offline, the route search and route augmentation components collaboratively answer users' trip queries in real-time.

(1) The Dynamic POI Network Model. The key problem of POI network model construction is to separately extract attributes of POI nodes from the Foursquare data set and information of the edges from the taxi GPS data set.

(2) Route Search. Given user-specified venues to visit, the starting time, and the time budget, the *route search* component returns routes that traverses all the intended venues from the starting location to the destination. In particular, the returned routes with a time margin greater than a user-determined threshold become candidate input to the *route augmentation* component. However, users might list too many venues to cover within the time constraint, or the planned visiting time does not agree with the operating hours of certain venues. If the TripPlanner system detects any of those cases, it will *interact* with the user to manually modify the venue list.

(3) Route Augmentation. This component aims to augment the candidate routes generated from the *route search* module with user-preferred venues inferred from the intended venue categories in the query, maximizing the route score under the given travel time budget. It first pulls together all of the venues that belong to user-preferred venue categories as candidate venues. Then for each candidate route, it tries to insert venues in the pool into it to *generate an augmented route*, without breaking any constraint. In the end, TripPlanner presents the augmented routes to the user, in an order sorted according to their scores in the *Augmented Route Ranking* module.

In the following two sections, we elaborate on the offline construction of the dynamic POI network, and the online route planning process respectively.

6.4 Dynamic POI Network Modelling

6.4.1 Node Modelling

Each node in the model corresponds to a POI with five attributes: *operation time, category that the venue belongs to, popularity, geographical location, and stay time*. For each venue, users provide their expected stay time, while Foursquare provides the relevant information for the former four attributes (Figure 6.2).

Figure 6.2: Relevant information of the node provided by Foursquare.

Operation time of a venue may vary according to the day of the week and even time of the year.

A venue can be associated with two or more *category labels* with different granularities[3]. Take the *Nick's Crispy Tacos* venue shown in Figure 6.2 as an example. It has three category labels, among which "Food" is a Level 1 label, "Breakfast Spot" is a Level 2 label, and "Multiplex" is a Level 3 label.

To compute the *popularity* of a given venue, we use two indicators: the total number of visitors (tvs) and total number of check-ins (tcs) (Eq. 6.1). The visitor number is usually *smaller* than the total check-in number for the same venue, since some users check-in the same venue multiple times during a visit.

$$Pop(v_i) = \frac{2 \times \frac{tvs(v_i)}{c_1} \times \frac{tcs(v_i)}{c_2}}{\frac{tvs(v_i)}{c_1} + \frac{tcs(v_i)}{c_2}} \tag{6.1}$$

where c_1 is the maximal visitor number of all venues in the targeted city, and likewise, c_2 is the maximal check-in number of all venues. Note that most visited venue may be different from the one with the most check-in record. The venue score is fused by the harmonic mean as we want both values to be relatively higher [105].

Regarding the *geographical location* of a given venue, Foursquare provides the longitude/latitude information together with its address.

Though the exact value of the *stay time* at a given venue cannot be precisely derived from check-in data, it could be roughly estimated by averaging the stay time of tourists. Note that users might specify an expected stay time when planning the trip and adjust it during the actual visit.

6.4.2 Edge Modelling

As the study mainly focuses on suggesting the optimal trip routes, for the sake of not disrupting the whole flow of presentation, we only briefly introduce how to estimate the dynamic edge values using taxi GPS traces here, and leave the technical details in Appendix A.2.

To more accurately estimate the dynamic *transit time by driving* from one node to another (i.e. the values of an edge), we need to consider the time-variant nature of traffic between venues. In this work, we leverage a real world dataset - taxi GPS traces. The taxi GPS data has two unique features: 1) *Spatial coverage*: a certain number of city taxis can fully cover the whole road network; 2) *Time coverage*: taxis usually operate in the whole

3. Foursquare categorizes all venues in a 3-level hierarchy. More details can be found at http://aboutfoursquare.com/foursquare-categories/

day, which is in line with tourists' visiting time. The two unique features of the taxi GPS data enable us to estimate the transit time between any two nodes within any time period.

To simplify the transit time calculation between nodes in the POI network, we first cluster co-located nodes among which walking is the best way to get around. The within-cluster transit time is computed using the average walking speed, while the between-cluster transit time is estimated based on the driving speed at the specific timeslot. Figure 6.3 illustrates a simple dynamic POI network. The small circles in different colors refer to the nodes (POIs). Near-by nodes are grouped into clusters (ellipses in the dashed line). Directed edges inside each cluster carry the walking time information between nodes which is independent of the time of the day; while directed edges across clusters carry the transit time information in between which is time-variant. For instance, during rush hours in the morning, the transit time from the upper right cluster to the bottom cluster is more than twice of the least travel time of the day (refer to the green curve in the bottom right of Figure 6.3 for a whole-day view of dynamic transit time).

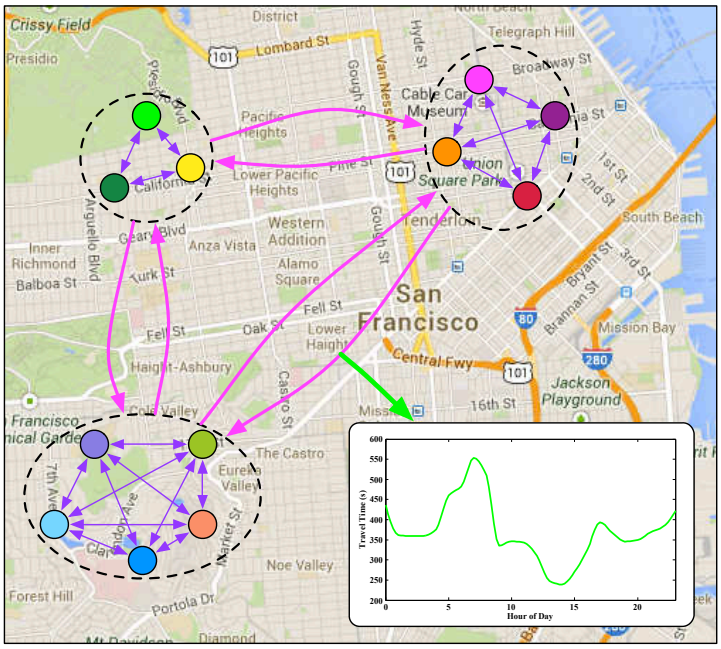


Figure 6.3: Illustration of the dynamic network built with Foursquare and Taxi GPS data sets.

6.5 The Two-Phase Approach

We take a two-phase approach, i.e. *route search* and *route augmentation*, to perform trip planning. *Route search* retrieves candidate routes traversing all *user-specified venues* within the time budget. *Route augmentation* further enriches the candidate routes with *user-preferred venues* as long as time permits, and recommends to users the optimal routes with the highest scores. The set of *user-preferred venues* is a subset of venues in the targeted city, which are obtained based on the user-preferred venue categories (CAT_u) in the itinerary query (IQ) (refer to Appendix A.3 for details).

6.5.1 Phase I: Route Search

The *route search* component works *interactively* with the user. Given a user's starting and ending places, specified venue list, and a travel time budget, it first checks and removes any venue that cannot be visited on the intended date. The module then returns all possible routes between the given origin and destination that cover the valid venues in the list. Note that users may be asked to shorten the venue list *iteratively* to ensure a proper time margin. In this process, the system would suggest the user to remove venue(s) with long distance to the starting and ending places. Consequently, candidate routes with time margins bigger than the user-specified threshold would be generated.

Moreover, according to the *Theorem* below, some candidate routes can be further pruned in this phase because they cannot lead to any valid route after the route augmentation phase.

Theorem 6.5.1. *Route which contains later-arrival venue can be pruned in advance. Here, "later-arrival" means arriving at the venue after its closing time; "earlier-arrival", on the contrary, refers to arriving at a venue before it opens.*

Proof. Based on *Lemma* 6.3.1, inserting a venue before the "later-arrival" venue will further push back the arrival time at this venue; while the "later-arrival" venue would be still later when inserting a venue after it. □

In a word, the output of the route search phase are all candidate routes that have enough time margins and do not contain any "later-arrival" venues.

6.5.2 Phase II: Route Augmentation

The *route augmentation* component tries to insert optional *user-preferred venues* into the candidate routes returned from the previous phase. The aim for optimization is to maximize the route score without exceeding the time budget. Route augmentation is

NP-hard and very challenging as it tries to satisfy two competing requirements: (1) the route should contain as many *user-preferred venues* as possible; (2) the route should meet the travel time budget and the venue visiting time constraints. We have to consider the following two factors when selecting new valid venues to optimize the route score.

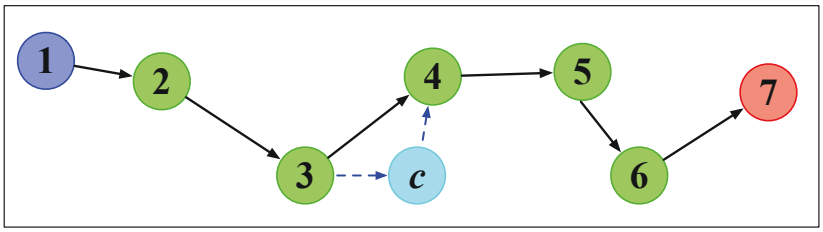


Figure 6.4: An illustrative example of inserting a venue into a candidate route.

1) **Arrival Time Delay by Adding New Venues.** Apparently, inserting new venues into a given route would increase its total visiting time, adding additional transit time and stay time. The arrival time to some of the existing venues may be delayed. Furthermore, the transit time needed between existing venues might also be different due to the time shift. Taking the diagram in Figure 6.4 as an example, after inserting venue v_c in the route, the arrival time to v_4, v_5, v_6, v_7 would be delayed, and the transit time between v_4 to v_7 might also change as the traffic conditions might be different later in the day.

2) **Total Route Score Increased by Adding New Venues.** Generally, adding more *user-preferred venues* would increase the score of a route, but may violate the given constraints if not done properly. We designed a method for route augmentation, which consists of two steps: *venue inserting* and *score maximization*. The former aims to find a suitable position in the candidate route to insert a selected venue, while the latter is responsible for maximizing the score of the updated route.

6.5.2.1 The Venue Inserting Algorithm

There are two principles that we should follow when inserting a new venue: the augmented route should be *valid* and we should minimize the extra cost in time. For a candidate route with n venues and a new venue v_c to insert, if the candidate route does not contain any "earlier-arrival" venue, we need to check $n-1$ positions to determine the final augmented route; however, if the candidate route does contain "earlier-arrival" venues, we only need to check $k-1$ ($< n-1$) positions, where k is the position of the first "earlier-arrival" venue in the candidate route according to *Theorem* 6.5.2.

Theorem 6.5.2. *For a candidate route which contains "earlier-arrival" venues, inserting a candidate user-preferred venue behind the first "earlier-arrival" venue could not lead to a*

valid route.

The pseudo-code of the venue inserting algorithm is shown in Algorithm 6.1. We first check whether the candidate route contains any "earlier-arrival" venue (Line 1). If it does, the possible positions where the new venue can be inserted are in $[2, k]$; otherwise, the range is $[2, n]$ (Lines 2-5). Note that the "wait" for a venue to open is not considered in this paper, as the total travel time is a hard constraint in our case. The core function of Algorithm 6.1 is the *augRoute* function shown in Algorithm 6.2. In this function, the candidate venue is inserted into the given route at each possible position (Lines 3-8). Note that not every position where the candidate venue is inserted can lead to a valid route (Lines 5-7). If no augmented routes are valid or the total travel time cost of all the generated augmented routes exceeds the time budget, the function returns the original input route (Lines 9-11); otherwise, it returns the augmented route with the minimum total travel time (Lines 12-13).

Algorithm 6.1 Venue Inserting Algorithm

Input: A candidate route $R = \langle v_1, v_2, \cdots, v_n \rangle$; A candidate venue v_c A user-specified total travel time budget Δ;
Output: An augmented route $augR$

1: **if** R has "earlier-arrival" venues **then**
2: $k = pos(R)$ //$pos(R)$ gets the index of the first "earlier-arrival" venue in R
3: $augR = augRoute(R, v_c, [2, k], \Delta)$
4: **else**
5: $augR = augRoute(R, v_c, [2, n], \Delta)$
6: **end if**

The algorithms above illustrate how to insert one venue to a candidate route. If there are multiple venues to add, this process will iterate through the list, again following the proposed principles. In the rest of the paper, we use the expression $R + \{v_{c1}, v_{c2}, \cdots, v_{cn}\}$ to denote the operation of inserting the venue list $\{v_{c1}, v_{c2}, \cdots, v_{cn}\}$ to the candidate route R sequentially. Note that for the same set of candidate venues, different inserting orders may result in different augmented routes (e.g. $R + \{v_{c1}, v_{c2}\} \neq R + \{v_{c2}, v_{c1}\}$).

6.5.2.2 Route Score Maximization Algorithms

We first present mathematical formulation of our route score maximization algorithms, then introduce how to compute the route score according to the user's preferences. In the end, we propose three heuristic algorithms to maximize the route score.

Algorithm 6.2 Venue Inserting Function

Function $augR = augRoute\ (R, v_c, [a, b], \Delta)$
$newR = \emptyset$
for $k = a : 1 : b$ **do**
  $tmpR = \langle v_1, v_2, \cdots, v_c, \cdots, v_n \rangle$ //the index of v_c in $tmpR$ is k, and venue orders in R keep unchanged in $tmpR$
  if $tmpR$ is *valid* **then**
    $newR = newR \cup tmpR$
  end if
end for
if $newR$ is empty or $min[\mathcal{TC}(newR)] > \Delta$ **then**
  $augR = R$
else
  $augR = \arg\min \mathcal{TC}(newR)$ //select the newly augmented route with the minimal total travel time cost
end if

Mathematical Formulation. For a given user u_i, a set of candidate venues $\{v_{ci}\}_{i=1}^{N}$, and a candidate route R, the route score maximization problem is:

$$\max \quad \mathcal{RS}(u_i, R + \{x_i v_{ci}\}_{i=1}^{N}) \tag{6.2}$$

Subject to:

$$x_i \in \{0, 1\} \tag{6.3}$$

$$x_1 v_{c1}.cat \cup x_2 v_{c2}.cat \cup \cdots \cup x_n v_{cN}.cat \subseteq CAT_u \tag{6.4}$$

$$\mathcal{TC}(R + \{x_i v_{ci}\}_{i=1}^{N}) \leq \Delta \tag{6.5}$$

where Eq. 6.2 refers to the objective function (i.e. the route score) for maximization. It is subjected to three constraints, as shown in Eqns. 6.3-6.5. Eq. 6.4 defines the constraint for the augmented venue selection, i.e. only the user-preferred venues can be selected for route augmentation, but not necessarily covering all venue categories, due to the total travel time constraint. Eq. 6.5 emphasizes that the total time cost of the newly augmented route should be within the predefined travel time budget Δ.

Route Score Calculation. The route score calculation algorithm is the core of the route augmentation component, which estimates the attractiveness of a candidate route to a given user. *The route score is defined as the sum of all its venue scores*, thus the venue scoring method is vital.

Venue Scoring. On one hand, the score of a venue is determined by its popularity (*Pop*, as shown in Eq. 6.1), which is objective (denoted as $\mathcal{VS}_{obj}$); On the other hand, the venue score is also related to individual user's personal interests revealed in his/her

check-in history, which is subjective. For instance, the scores of "Art & Museum" venues should be higher for a user, if he/she visits venues in this category more often than the others as shown in the Foursquare check-in records. The normalized check-in preference value ($\mathcal{VS}_{sub}$) of the venue v_i for user u_j is calculated by Eq. 6.6. For simplicity, only the level-1 category labels (i.e. the nine category labels defined by Foursquare) are used in the scope of this study.

$$\mathcal{VS}_{sub}(u_j, v_i) = \frac{tcs(u_j, \{v_i.cat\})}{tcs(u_j)} \tag{6.6}$$

where $tcs(u_j)$ represents the total number of check-ins that the user u_j conducted in Foursquare, while $tcs(u_j, \{v_i.cat\})$ stands for the total number of check-ins at venues belonging to the same category v_i.

Finally, the venue score can be computed according to Eq. 6.7, considering both the venue popularity and the user preferences.

$$\mathcal{VS}(u_j, v_i) = \mathcal{VS}_{obj}(v_i) + \mathcal{VS}_{sub}(u_j, v_i) \tag{6.7}$$

Three Heuristic Algorithms. As discussed previously, there are two important steps in route augmentation: *selecting* new venues, and *inserting* them into the candidate routes sequentially. It is not trivial since we have to make a trade-off between the individual venue scores and the total number of venues that can be added. For example, inserting a far away venue with a very high venue score might forbid adding more new venues, since it has already used up the time budget. In contrast, inserting a close-by venue with an average venue score first would allows including more new venues. It is difficult to say which strategy would achieve a higher route score in the end. Hence, we propose three heuristic algorithms for maximizing the route score in the route augmentation phase. Note that added venues are all *user-preferred* venues.

1) *Travel Time Minimizer.* The basic idea of this algorithm is to insert as many new venues as possible, given the fact that the route score would be higher as the number of venues increases in general. Thus at each venue inserting iteration, our proposed heuristic is that the venue closest to the candidate route (measured by the additional travel time) would be selected first for insertion, *regardless of its venue score*. We illustrate the core part of the *travel time minimizer* algorithm in Algorithm 6.3.

For each *candidate route* returned by the route search phase, in each iteration, we need to examine all *new venues* in order to select *one* for the newly augmented route (Lines 2∼10). This is quite time-consuming especially when the size of the venue list is big. We use the venue inserting algorithm as shown in Algorithm 6.1 for each new venue (Lines 3∼4). If the newly augmented route is not valid, its total travel time would be set to $+\infty$;

Algorithm 6.3 Travel Time Minimizer Algorithm

Input: A candidate route $R = \langle v_1, v_2, \cdots, v_n \rangle$; A set of new venues V_c (i.e. user-preferred venues); A user-specified total travel time budget Δ;

Output: An augmented route

1: $augR = \emptyset$
2: **for** $i := 1 : 1|V_c|$ **do**
3: $v_{ci} = V_c(i)$
4: $augR = \{R + v_{ci}\} \cup augR$ //Call the venue inserting algorithm
5: **if** $R + v_{ci} = R$ **then**
6: $TC(i) = +\infty$ //the total travel time cost of the route is set $+\infty$ if the selected venue can not be inserted
7: **else**
8: $TC(i) = \mathcal{TC}(R + v_{ci})$
9: **end if**
10: **end for**
11: **if** $any(TC) \neq +\infty$ **then**
12: $k = \arg\min_i(TC)$
13: $R = augR(k)$
14: $V_c = V_c - v_{ck}$
15: **end if**
16: **Repeat Lines 1∼14**
17: **Until** R **keeps unchanged**

otherwise, it would be updated to that of the newly augmented route (Lines 5∼9). At the end of each iteration, the route in the $augR$ set with the minimum total travel time cost will be selected as the input (Lines 11∼15) for the next round of venue inserting, again via the same heuristic (Line 16). The algorithm would terminate when no new route can be generated (Line 17). Note that the inserted venue needs to be removed from the new venue list before the next iteration (Line 14). Therefore, the computation complexity in each iteration for this algorithm has an upper bound of $\mathcal{O}(n-1)^N$, where n is the number of existing venues in the original candidate route, and N is the total number of user-preferred venues.

2) *Venue Score Maximizer.* The basic idea of this algorithm is to prioritize high-scored venues. Thus in each iteration, the venue with the highest venue score that can lead to a valid route would be inserted first, *no matter how far away it is from the candidate route.* Algorithm 6.4 illustrates the core part of the proposed *venue score maximizer.*

In each iteration, we first *sort* the new venues in the descending order of venue scores defined in Eq. 6.7 (Line 1). This *sorting* operation can save computation time as we only need to check whether the higher-ranked venues can yield a valid augmented route. If yes, there is no need to examine the rest of the venue list as in Algorithm 6.3. In the best case,

Algorithm 6.4 Venue Score Maximizer Algorithm

Input: A candidate route $R = \langle v_1, v_2, \cdots, v_n \rangle$; A set of new venues V_c (i.e. user-preferred venues); A user-specified total travel time budget Δ;

Output: An augmented route

1: $V_c \leftarrow sort(V_c)$ //Sort new venues according to their scores in descending order defined in Eq. 6.7
2: $i = 1; v_{ci} = V_c(i)$
3: $augR = R + v_{ci}$
4: **while** $augR = R$ **do**
5: **if** $i < |V_c|$ **then**
6: $i = i + 1; v_{ci} = V_c(i)$
7: $augR = R + v_{ci}$
8: **else**
9: **Break**
10: **end if**
11: **end while**
12: $R = augR$
13: $V_c = V_c - v_{ci}$
14: **Repeat** Lines 2∼ 14
15: **Until** R **keeps unchanged**

the first venue (with the highest venue score) meets the requirement (Lines 2∼3); in the worst case, all new venues will be checked (Lines 5∼10). At the end of each iteration, the route with the highest route score will become the candidate route for the next iteration (Lines 12∼14). The termination condition is the same as that of the *travel time minimizer* (Line 15). Again, the inserted venue would be excluded from further operations (Line 13). Therefore, the computation complexity in each iteration for this algorithm varies from $\mathcal{O}(Nlog|N|)$ (i.e. the best case) to $\mathcal{O}(Nlog|N| + (n-1)^N)$ (i.e. the worst case). Note that $\mathcal{O}(Nlog|N|)$ is the complexity of *sorting* operation.

The above two algorithms are used as baseline methods. The first heuristic algorithm only considers the number of the venues added, while the second one emphasizes merely on the scores of the inserted venues. As a result, the routes of the first algorithm would be generally longer (i.e. containing more venues), compared to the second algorithm. It is because the second heuristic algorithm, given the same time budget constraint, favors having one venue with a high venue score over two nearby average venues, even though the latter case might lead to a higher route score. To overcome the limitations of these two baseline methods, we propose our *gravity maximizer*.

3) *Gravity Maximizer.* Inspired by *Newton's law of universal gravitation* which is capable of modelling human mobility patterns (the travel behaviors to places, travel patterns, etc.) [25, 133], we introduce a *gravity model* that uses the venue scores and the venue dis-

tances to the candidate route collectively for route augmentation. In our gravity model, the *spherical distance* between the candidate route and the new venue is analogy with the distance defined in *Newton's gravity model*, where the location of the candidate route is obtained by averaging the locations of all venues that it contains. Likewise, the average venue score of the candidate route and the score of new venue corresponds to the *mass*. Finally, the gravity can be computed using Eq. 6.8.

$$G(v_{ci}, R) = \frac{\mathcal{VS}(u_j, v_{ci}) \times \frac{1}{n} \sum_{i=1}^{n} \mathcal{VS}(u_j, v_i)}{dist(v_{ci}, R)^{\lambda}} \tag{6.8}$$

In the proposed *gravity maximizer*, the new venues are *sorted* in the descending order of their gravity values computed via Eq. 6.8, instead of the venue scores. The rest of procedure is exactly the same as that of *venue score maximizer*. Thus the two methods are similar in the computation complexity, with an extra cost of the venue's gravity computation in the *gravity maximizer*.

In fact, the ranking based on gravity values would be degraded to that of venue scores if we set $\lambda = 0$, as gravity values would be determined by venue scores only. In other words, the *gravity maximizer* and *venue score maximizer* algorithms would reach the same result when $\lambda = 0$. On the contrary, as can be inferred from Eq 6.8, if we set λ to be extremely high (e.g. $\lambda > 5$), the gravity values would be dominantly influenced by the distance to the candidate route, introducing a bias towards the "closest" venue (i.e. with the smallest distance to the candidate route). This agrees with the basic idea of the *travel time minimizer* algorithm. Furthermore, with a large negative λ (e.g. $\lambda < -5$), "distant" venues would be ranked higher, which should be avoided. We will investigate how different λ values affect the algorithm's performance in Section 6.6.2.

6.5.2.3 Augmented Route Ranking

The algorithms discussed in Section 6.5 aim to optimally augment the set of candidate routes returned from the Phase I (i.e. route search). *Augmented Route Ranking* operation then picks out the augmented route with the highest route score to answer the user's itinerary query (*IQ*). Note that if multiple "optimally" augmented routes possess the same route score as they may contain the same venues but in different order, the route with a smaller "total travel time" would be ranked higher.

6.6 System Evaluation

Here, we present the evaluation results that aim to: (1) validate the efficiency and effectiveness of the trip planning algorithms and (2) demonstrate the usefulness and personalization capability of the trip planning system. We first describe the experiment setup,

results of the parameter sensitivity study, as well as evaluation on algorithm efficiency and effectiveness, and then discuss several issues which need to be addressed further.

6.6.1 Experiment Setup

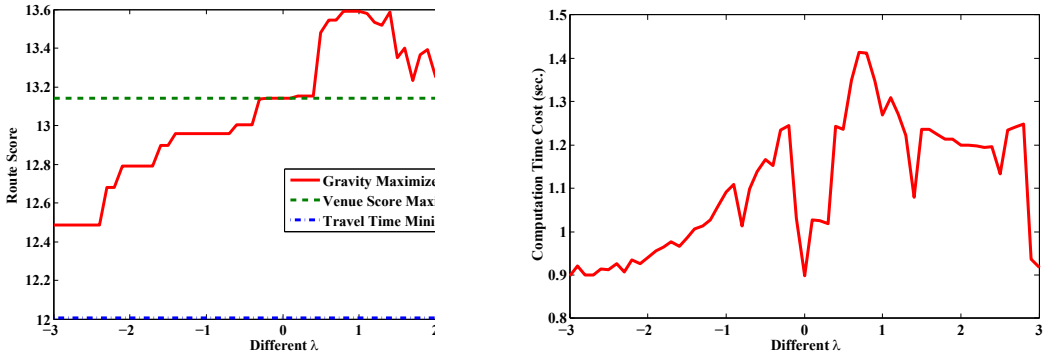
Data Preparation. We used Foursquare check-in data of San Francisco from April 2010 to October 2010, and the taxi GPS traces of the same city from the CabSpotting project (`http://cabspotting.org/`) to construct the POI network of San Fransisco. The Foursquare data contains 110,214 check-ins generated by 15,680 users. The taxi GPS data contains 391,938 passenger-delivery trips generated by 536 taxis in June 2008. We did not include data from the vacant taxis since they might not drive at a normal speed when searching for passengers. Although we could not find two data sets from the same period for evaluation, the process of our proposed framework is data-independent, and the results can be easily updated once we are able to provide these heterogeneous data from the same period. The procedure of the POI network construction has been discussed in Section 6.4, and more details can be found in Appendix A.2.

Evaluation Environment. All the evaluations in the paper are run in Matlab on an Intel Xeon W3500 PC with 12-GB RAM running Windows 7.

6.6.2 Parameter Sensitivity Study

As discussed in Section 6.5, we have only one internal parameter λ in the proposed *gravity maximizer* algorithm (Eq. 6.8), and no internal parameter in the other two baselines. We are thus interested in how it affects the optimal route score. We do not set λ to extreme values as discussed; instead, we vary λ in the range of [-3,3] with an interval of 0.1. The optimal scores under different λ values, in comparison with the two baseline algorithms are shown in Figure 6.5(a). As the figure suggests, the optimal route score generated by the *travel time minimizer* algorithm is always the lowest since it does not take the individual score of candidate venues into consideration. As expected, the optimal route score computed by the *gravity maximizer* algorithm and the *venue score maximizer* algorithm are the same when λ is around 0. We also find that the *gravity maximizer* algorithm yields higher optimal route score than the *venue score maximizer* algorithm when λ is within the range of [0.5 2.3].

We also show the change in computation time of the *gravity maximizer* algorithm under different λ values in Figure 6.5(b). More specifically, the computation time fluctuates with the increase of λ. However, the maximum time cost is no longer than 1.45 seconds, which is acceptable. Considering the trade-off between route score and computation time, we choose $\lambda = 1.5$ for the rest of the evaluations.



(a) Optimal route scores under different λ. (b) Computation time cost under different λ.

Figure 6.5: Results of parameter sensitivity study.

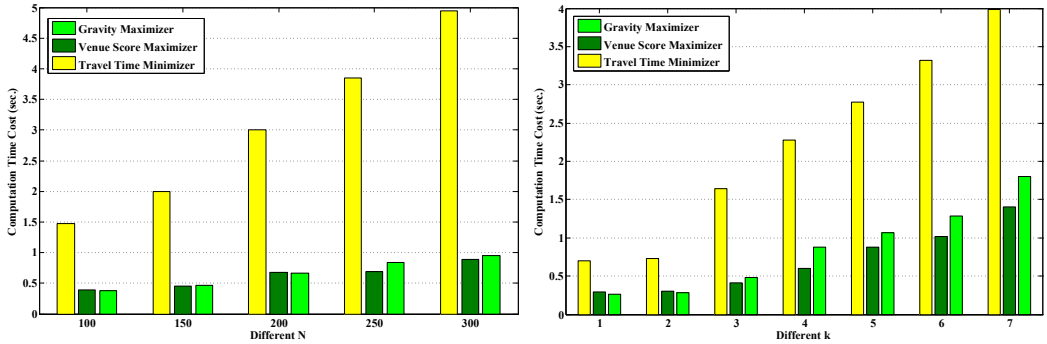
6.6.3 Efficiency Evaluation

The efficiency of the three algorithms depends on several parameters, such as the total number of venues (N) in the targeted city, the number of user-preferred venue categories (k), the number of user-specified venues (m), and user-defined travel time budget (Δ). The first two variables determine the number of user-preferred venues (i.e. candidate new venues). The number of user-specified venues and travel time budget have an impact on the number of candidate routes produced in Phase I (i.e. the route search phase), as well as on the number of user-preferred venues that can be inserted in Phase II. Particularly, at most $m!$ candidate routes can be produced. The number of user-specified venues (m) is common for all three algorithms, affecting the computation time in both the route search phase and the route augmentation phase. For simplicity, we fix $m = 5$ in all the evaluations. In the following experiments, we mainly study how the choice of N, k and Δ affects the computation time of the three algorithms, varying only one parameter at a time.

It should be noted that all the candidate routes are augmented *in parallel*. In other words, the total computation time in the route augmentation phase is equal to the maximum computation time among all candidate routes. The efficiency is measured by the total time cost in both phases.

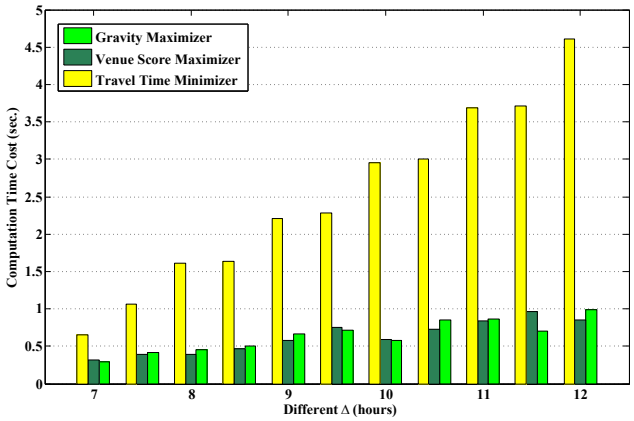
6.6.3.1 Varying N

The relationship between the computation time of the three algorithms and the total number of venues in the city (N) is shown in Figure 6.6(a). Results suggest that the proposed *venue score maximizer* and *gravity maximizer* algorithms are less time consuming compared to the *travel time minimizer* algorithm, which is consistent with the complexity analysis. Furthermore, the computation time of the *travel time minimizer* algorithm is



(a) Computation time cost by varying N.

(b) Computation time cost by varying k.



(c) Computation time cost by varying Δ.

Figure 6.6: Results of efficiency evaluation.

almost proportional to N. This is logical as *travel time minimizer* needs to examine the additional travel time introduced by each venue in the candidate list. On the contrary, the computation time of the *venue score maximizer* and *gravity maximizer* algorithms only goes up slightly as the number of venues increases. Moreover, these two algorithms took less than one second to generate the result. The *gravity maximizer* algorithm generally took a slightly longer time than the *venue score maximizer* because of the additional gravity value calculation for each user-preferred venue. In this experiment, we fix $k = 3$ and $\Delta = 10$ hours.

6.6.3.2 Varying k

We show the computation time of the three algorithms under different k in Figure 6.6(b). In general, the computation time increases with k. This is because a larger k often leads to a bigger number of user-preferred venues for augmentation. Again, the computation

time of the *travel time minimizer* algorithm is much longer than that of the other two algorithms under the same setting, for the same reason as when N varies. For the *venue score maximizer* and *gravity maximizer* algorithms, their computation time increases more significantly as k becomes bigger, compared to that under different N. This is indeed caused by the increase of the number of user-preferred venues. As N increases, both the number of user-preferred and non-user-preferred venues would increase. However, all non-user-preferred venues can be excluded from the route augmentation process and thus have no impact on the computation time. In contrast, any change in k would be completely and directly reflected on the change in the number of user-preferred venues. In this experiment, we fix $N = 300$ and $\Delta = 8.5$ hours.

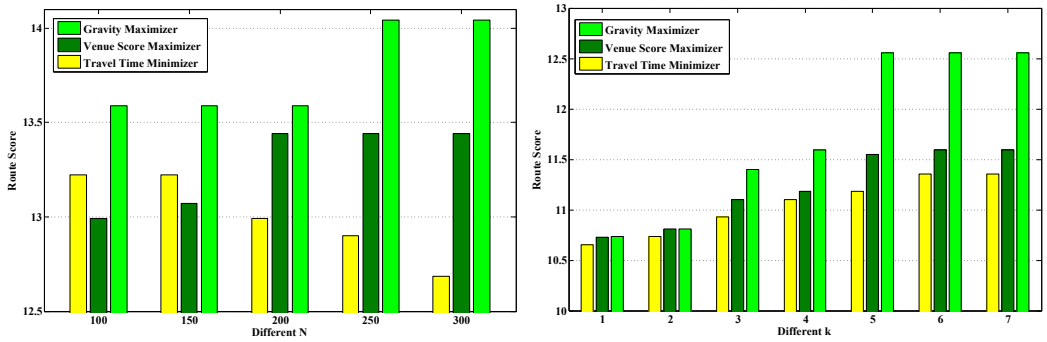
6.6.3.3 Varying Δ

Figure 6.6(c) shows the change in computation time of the three algorithms under given total travel time budget Δ. Similar to the previous two cases, the *travel time minimizer* algorithm needs more time as Δ increases, much more than the other two algorithms of which the computation time was similar and no more than one second. In general, more user-preferred venues are allowed to be added which results in more venue inserting iterations in the route augmentation process, especially for the *travel time minimizer* algorithm since its objective is to minimize the introduced travel time at each iteration. In this experiment, we fix $N = 300$ and $k = 3$.

6.6.4 Effectiveness Evaluation

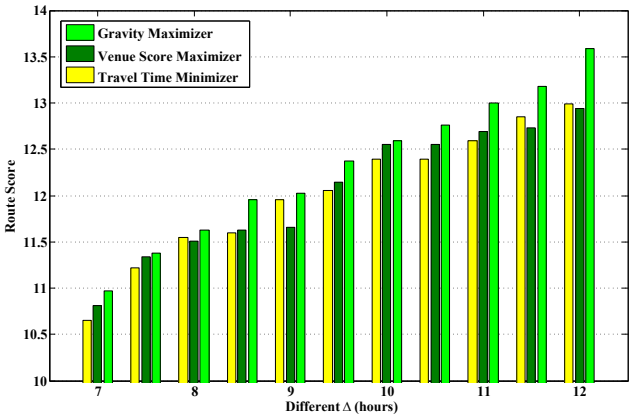
Similar to the study of efficiency, we assessed the effectiveness of route augmentation algorithms under the same settings. The optimal route scores returned by the three algorithms with varying N, k and Δ are shown in Figure 6.7(a), Figure 6.7(b), and Figure 6.7(c) respectively. In Figure 6.7(a), the experiment setting is $m = 5$, $k = 3$, $\Delta = 10$ hours; in Figure 6.7(b), the setting is $N = 300$, $m = 5$, $\Delta = 8.5$ hours; and in Figure 6.7(c), the setting is $N = 300$, $m = 5$, $k = 3$. In all three cases, the proposed *gravity maximizer* algorithm consistently outperformed the other two baseline methods in terms of optimizing the route score. Figure 6.7(a) shows that the optimal route score of the *travel time minimizer* algorithm decreases gradually as N increases, as opposed to the *gravity maximizer* and *venue score maximizer* algorithms. This is because the inherent characteristic of the *travel time minimizer* algorithm biases towards venues that are *closer* but probably with a *smaller* score as N increases. Results also suggest that, compared to the *venue score maximizer* algorithm, the *gravity maximizer* algorithm is more likely to find the global optimal route score. In Figure 6.7(b) and Figure 6.7(c), all three algorithms achieved higher optimal

route score with bigger k and Δ. However, such increase dramatically slowed down when $k > 5$, probably due to the time budget constraint we impose.



(a) Optimal route score by varying N.

(b) Optimal route score by varying k.



(c) Optimal route score by varying Δ.

Figure 6.7: Results of effectiveness evaluation.

6.6.5 Case Study

We further tested the personalization capability of the TripPlanner system in the case that two users with different personal interests submit the same query (IQ_1) to the system. To be more specific, according to their check-in history, one of the users (u_1) preferred *Great Outdoors* and *Restaurants* venues, while the other user favored more of the *Arts & Entertainments* and *Restaurants* venues. To demonstrate the *traffic-aware* capability of our TripPlanner, we designed a second case in which u_1 modified the query and set a different trip starting time (IQ_2). To verify that the route recommended by TripPlanner is optimized, we introduced a third case in which the recommended route in response to IQ_2 by u_1 was compared to an average route. Queries in all three

Table 6.2: The information about three designed cases.

	Users	Starting Time	Recommended Route	Route Score
Case I	u_1	10:00 am	R_1	14.4176
	u_2	10:00 am	R_2	14.6883
Case II	u_1	10:00 am	R_1	14.4176
	u_1	08:30 am	R_3	13.6602
Case III	u_1	08:30 am	R_3	13.6602
	u_1	08:30 am	R_4	12.9087

cases share the following information: i) The users start and end the trip both at the the *Caltrain Station*; ii) User-specified venues include *Museum, Golden Gate Bridge, Beach, Lombard Street and Fisherman's Wharf*; iii) the total travel time budget Δ is set to 11 hours; iv) the optional user-preferred categories are {*Restaurants, Arts & Entertainments, Great Outdoors*}; and v) the dining time is set to [11:00 am, 12:59 pm] for lunch and [17:30 pm, 20:00 pm] for dinner. Table 6.2 lists the information of the three cases we designed, including the corresponding user, starting time, and results of the recommended route.

Case I: Personalization Capability. This case intends to demonstrate the *personalization* capability of TripPlanner with two different users. As shown in Figure 6.8(a) and 6.8(b), given the same time budget, both users can accommodate four more preferred venues in their trips additional to the must-visit venues (i.e. R_1 and R_2). Further investigation showed that, even though not explicitly requested, TripPlanner recommended restaurants to both users around lunch and dinner time since they are food lovers (as shown in Figure 6.8(e)). For the user u_1, the other two venues added belong to the *Great Outdoors* category; while two more museums from the *Arts & Entertainments* category appeared in the augmented itinerary for u_2. As illustrated in Figure 6.8(e), u_1 arrives at the *Caltrain Station* a bit earlier than u_2, suggesting that route R_1 and R_2 have different travel time. These results clearly indicate the ability of TripPlanner to customize both specified and top-ranked preferred venues in the recommended trip, according to users' preferences.

Case II: Traffic-aware Capability. This case looked into the *traffic-aware* capability of TripPlanner. We compared two queries (IQ_1 and IQ_2) of the same user u_1 that only differ in the starting time (Table 6.2). The recommended routes (R_1 and R_3) are shown in Figure 6.8(a) and 6.8(c) respectively. Only three preferred venues can be added in R_3, as it starts around the morning rush hours and thus needs more transit time compared to the other two routes (R_1 and R_2), resulting in a smaller route score of 13.6602. Similar to R_1 and R_2, proper lunch and dinner are planned for the user. In addition, the user is suggested to visit the far-away *Golden Gate Bridge* first since most of venues such as museums are not yet opened early in the morning.

Case III: Route Score Optimization Capability. In this case, we are interested in the *difference* between the *optimal route* versus an *average route*. Figure 6.8(d) shows a randomly selected augmented route (R_4) which is generated by our proposed *gravity maximizer* algorithm under the same query as Figure6.8(c). The user is also suggested to visit the far-away *Golden Gate Bridge* first in R_4. Even though the average route includes all the user-specified venues and meets the total travel time budget constraint, it only accommodates two preferred venues due to the long transit time caused by taking an *inefficient* venue visiting order. As a result, the user only has time to take a quick snack for lunch if taking R_4. Therefore, its route score (12.9087) is much lower than that of the recommended optimal route R_3.

6.6.6 Discussion

In the following, we discuss some issues of TripPlanner, which need to be addressed in future work.

Venue Stay Time. In the current study, we assume that the stay time at a venue can be obtained in advance. However, actually estimating the stay time for each individual user at a particular POI is not trivial. It depends on the user's interest as well as his/her time budget. For instance, the museum lovers might spend the whole day in the *Louvre*, while some people only spend 2 hours to visit the most famous artworks, especially when the trip schedule is tight. In the future, we plan to explore other data sources and techniques to estimate each user's preferred stay time at different venues [41].

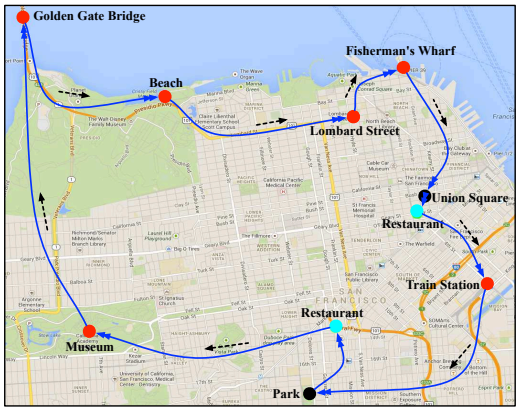
Route Score. There is no objective way to quantitatively characterize the relative importance of different POIs for each individual. In this study, we intentionally add a subjective score based on a user's check-in history to characterize the attractiveness of a POI to him/her, in addition to its popularity. Although the proposed scoring method that leverages the existing literature seems to work well, further research is needed to identify more effective ways to automatically assign attractiveness scores to different POIs and arrange the visiting order accordingly.

6.7 Concluding Remarks

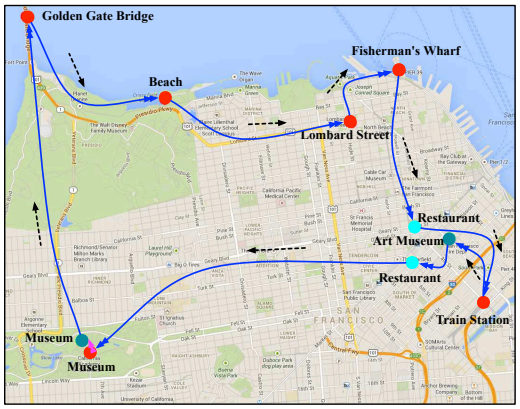
In this study, we have developed a novel framework called TripPlanner for *personalized, interactive and traffic-aware* trip planning. It leverages two heterogeneous data sources and considers factors including the varying transit time between POIs, user preferences, and the total travel time budget. First, we constructed the dynamic POI network model by extracting relevant information from crowdsourced Foursquare and taxi GPS traces. Then we proposed a *two-phase approach* for personalized trip planning with a

comprehensive route scoring method and a novel route *search-augmentation-ranking* process. Using two real-world data sets which contain more than 391,900 passenger-delivery trips and 110,200 check-ins in the city of San Francisco, we compared our proposed route augmentation method with two baseline algorithms, and showed that our method is more efficient and effective than the baseline approaches. We further conducted a case study to validate the capability of our framework in recommending adaptive and optimal itineraries.

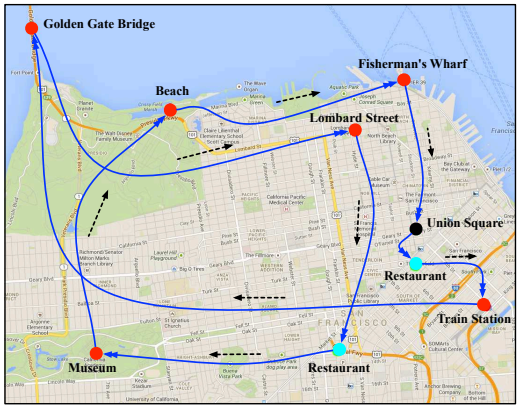
In the future, we plan to broaden and deepen this work in several directions. First, we intend to extend the scenarios to multi-day itinerary planning. Second, we would like to deploy our system on mobile devices, enabling a series of pervasive smart travel and transportation planning services. Third, we plan to test our system with real users in actual practices, collecting feedback on how to improve the service further.



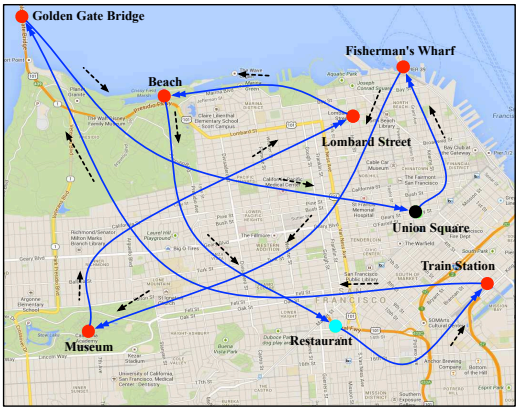
(a) R_1. Starting time: 10:00 am



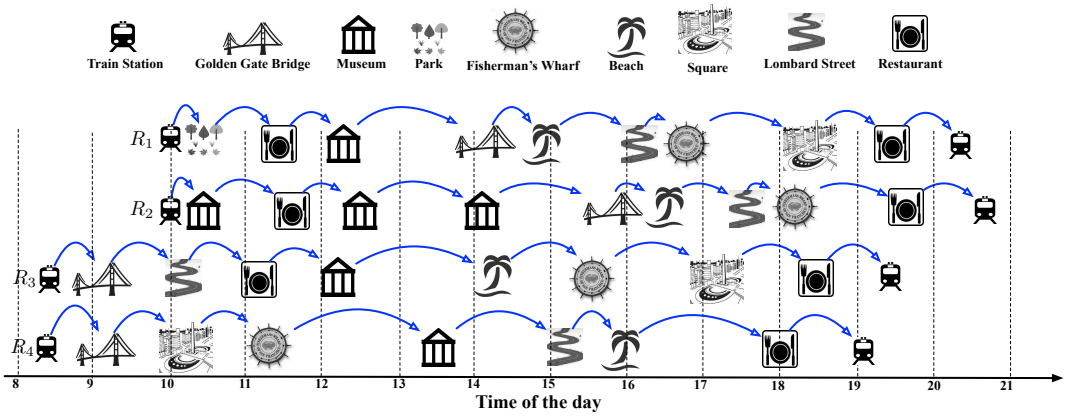
(b) R_2. Starting time: 10:00 am



(c) R_3. Starting time: 08:30 am



(d) R_4. Starting time: 08:30 am



(e) Comparison of four routes in the temporal dimension.

Figure 6.8: Results of the case study. (a)∼(d) show the trip routes on Google map (in the spatial dimension).

Chapter 7

Conclusion and Future Work

Contents

7.1 Conclusion

Taxi GPS traces, in spite of being a very specialized type of digital footprints, have already provided us with a rich data source to uncover many "hidden facts" and insights about the community and city, including *social dynamics, traffic dynamics and operational dynamics*. With the extracted *social and community dynamics*, many useful applications can be further enabled to meet the real-world needs. In this thesis, we explored certain aspects of social and community dynamics to offer diverse urban services to certain end users, such as taxi passengers, taxi drivers, city planners, and regular city citizens.

First, we proposed a novel and effective algorithm for fast real-time detection of anomalous trajectories obtained from GPS-equipped taxis that can use fixed and variable window sizes. In addition to classifying full trajectories as anomalous, *iBOAT* can work with ongoing trajectories and can determine which parts of a trajectory are responsible for its anomalousness. We further showcased its use for fraudulent behaviour analysis and detecting road network changes. The result suggests that most anomalous trajectories are in fact due to fraud. We also provided evidence to deny possible excuses for fraud behaviours.

Second, we investigated the problem of night-bus route planning by leveraging the taxi GPS traces, which is motivated by the needs of applying pervasive sensing, communication and computing technology for sustainable city development. We proposed a two-phase

approach to solve this problem. In the first phase, we developed a process to cluster "hot" areas with dense passenger pick-up/drop-off, and then propose effective methods to split big "hot" areas into clusters and identify a location in cluster as the candidate bus stop. In the second phase, given the bus route origin, destination, candidate bus stops as well as bus operation frequency and maximum total travel time, we derive several criteria to build bus route graph and prune the invalid stops and edges iteratively. Based on the graph, we further develop two heuristic algorithms to automatically generate candidate bus routes in both directions, and finally we select the best route which expects the maximum number of passengers under the given conditions.

Finally, we investigated the problem of personalized trip planning, which is motivated by the needs of considering real-world traffic conditions, user preferences, and the travel time budget. This work is among our initial attempts to apply pervasive computing techniques in achieving better trip planning of smart cities. To solve the problem, we propose the TripPlanner framework, leveraging a combination of LBSN and taxi GPS data sources. The foundation of the framework is the dynamic POI network model, where the LBSN data is used for venue node feature extraction, and the taxi GPS data is used for obtaining the time-dependent edge weights (i.e., the transit time) among nodes. We proposed a two-phase approach for trip planning. The route search phase works interactively with users to ensure users to specify proper POIs, and returns candidate routes covering user-specified POIs. The route augmentation phase adds the POIs belonging to the preferred categories iteratively, aiming to maximize the route score.

To sum up, the first work discussed provided application informing taxi passengers whether the taxi drivers take the honest routes during their rides "on the fly", while the second and third work enabled applications about route planning and itinerary planning in transportation networks, subject to different constraints and optimization objectives.

7.2 Future Work

Although we have shown the superior capabilities of the taxi GPS traces in uncovering many "hidden facts" about city dynamics, they may be *biased and not representative*. Specifically, travelling around the city by taxi takes up only a small fraction of the total public transportation due to its relatively higher price. In other words, the usage of taxi may be restricted to some certain groups of people, compared to the usage of public buses. Similar situation may also occur when dealing with the traffic dynamics using the taxi GPS traces, since the taxi volume takes only a small percentages of the total traffic volume (e.g., around 15$\sim$20% in Beijing). Questions, such as, to what degree the uncovered dynamics by taxi users can represent the whole population, remain unknown. Fortunately, many

data sources from multiple domains become increasingly available. For example, many city agencies and authorities are making their data accessible for public usage (i.e., open data)[1], which provides us with unprecedented opportunities to understand social and community dynamics in an *integrated and holistic* view. We believe that many more interesting applications and urban services can be enabled if we couple the taxi GPS data with other complementary data sources. Our work in Chapter 6 has demonstrated the advantages of fusing the taxi GPS data and Foursquare check-in data. The future work will focus on exploring the complementary information provided by the cross-domain data sources to offer city planners and dwellers many insights and services that bring them closer to the vision of a smart city. Along this line, we list three promising directions for future research.

♢ Fusing personal mobile phone and smart card data to re/design better transportation network. The very basic problem of transportation network design is the estimation of city-wide OD flow. More accurate estimation of OD flow can be achieved if we integrate multiple data sources, such as mobile phone data, public transportation usage records data (i.e. smart card data), and taxi GPS data. Specifically, given the fact that mobile phone have become an essential element in the lives of most people in many countries, it is clear that it can reveal the population movement flows among different city regions to a large degree. The public transportation usage records data reflect the actual OD flow in the current transportation networks in a fine spatial and temporal granularity (i.e. the actual OD flow among different stations at different time slots). The taxi GPS data can inform us a very high resolution of taxi passenger flow (i.e., from which point to which point in the city at what time). These different data sources reflect different aspects of the OD flow, and therefore, with appropriately integrating them, we can plan new effective public routes, examine and redefine the current transportation networks. Further more, coupling with the POI data, we can design even better transportation network.

♢ Fusing real-time sensory data and user-generated content data to provide better real-time traffic services. Real-time traffic information is of great importance for route planning. With the real-time traffic information provided by the road sensors (e.g., video surveillance, loop sensors) and the GPS streaming data from taxis, better traffic forecasting in the short-time window can be achieved. People may also wonder why the traffic in the road network is abnormally heavy, thus traffic diagnosing is also very important. As the social networking services such as Twitter have also become an essential part of people's lives, many user contents (e.g., pictures, texts) are generated

1. http://www.data.gouv.fr
http://data.london.gov.uk
https://nycopendata.socrata.com

in near real-time, which can be used to diagnose the traffic.

♢ Fusing personal smart phone sensory data to support crowd-sensing/sourcing tasks. Rich sensors are embedded in smart phones, such as accelerometer, acoustic, and even air quality measurement sensors. Given the facts that smart phones are widely used among most of people, and taxi drivers are driving continuously in the road network, many novel crowd-sensing/sourcing tasks can be enabled if fusing smart phone sensory data and the taxi GPS data. For instance, the collected accelerometer data have been recognized as a powerful source to identify the state of their users, such as walking, running, staring. With the accelerometer data from smart phones, we can detect accurately when the taxi stops. With further integration of the taxi GPS data, we can identify the traffic lights in the road network in the city. Moreover, the vibration patterns recorded in the smart phones during driving, can be also used to detect road potholes and road roughness levels, which impacts transport safety and driving comfort. Through a crowd-sourced way, a full picture about the traffic light locations and road roughness of road network can be obtained. This research is very promising, and the main difficulties are "how to design proper crowd-sensing/sourcing tasks which can emphasize the inherent characteristics of the taxi GPS traces", "how to design incentive mechanisms to stimulate taxi drivers to participate the tasks", "how to keep data fidelity for applications while protecting privacy","how to select/trust users effectively", subject to certain constraints such as the spatial coverage, the time cost, and so on.

Bibliography

[1] New york city taxi fraud. http://www.consumertraveler.com/today/nyc-taxi-drivers-overcharge-passengers-8-3-millioin/.

[2] Public transportation factbook. Technical report, American Public Transportation Association, 2011.

[3] Zhejiang online news. http://biz.zjol.com.cn/05biz/system/2013/01/25/019113078.shtml, 2013.

[4] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 504–509, 2006.

[5] J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11(6):6 – 28, dec. 2004.

[6] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. In *Proceedings of the ACM International Symposium on Advances in Geographical Information Systems*, pages 22:1–22:8, 2007.

[7] A. Amiri. A probabilistic greedy algorithm for channel assignment in cellular radio networks. *IEEE Transactions on Communications*, 58(11):3286–3295, 2010.

[8] C.-N. Anagnostopoulos, I. Anagnostopoulos, V. Loumos, and E. Kayafas. A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):377–392, 2006.

[9] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.

[10] Y. Arase, X. Xie, T. Hara, and S. Nishio. Mining people's trips from large scale geo-tagged photos. In *Proceedings of the International Conference on Multimedia*, pages 133–142, 2010.

[11] J. Aslam, S. Lim, X. Pan, and D. Rus. City-scale traffic estimation from a roving sensor network. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pages 141–154, 2012.

[12] R. K. Balan, K. X. Nguyen, and L. Jiang. Real-time trip information service for a large taxi fleet. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, pages 99–112, 2011.

[13] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pages 199–208, 2012.

[14] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. Werneck. Route planning in transportation networks. Technical Report MSR-TR-2014-4, Microsoft Research, 2014.

[15] F. Bastani, Y. Huan, X. Xie, and J. Powell. A greener transportation mode: flexible routes discovery from GPS trajectory data. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 405–408, 2011.

[16] S. Basu Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In *Proceedings of IEEE International Conference on Data Engineering*, pages 15–26, 2011.

[17] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *The Journal of Machine Learning Research*, 3:1229–1243, 2003.

[18] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 79–88, New York, NY, USA, 2012. ACM.

[19] J. Biagioni and J. Eriksson. Inferring road maps from GPS traces: Survey and comparative evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, page to appear, 2014.

[20] S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of IEEE International Conference on Data Engineering*, pages 421–430, 2001.

[21] G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis*. Wiley, 2008.

[22] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 93–104, 2000.

[23] I. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso. Where shall we go today? planning touristic tours with tripbuilder. In *Proceedings of the ACM International Conference on Information & Knowledge Management*, pages 757–762, 2013.

[24] Y. Bu, L. Chen, A. W.-C. Fu, and D. Liu. Efficient anomaly monitoring over moving object trajectory streams. In *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 159–168, 2009.

[25] F. Calabrese, G. Di Lorenzo, L. Liu, and C. Ratti. Estimating origin-destination flows using mobile phone location data. *IEEE Pervasive Computing*, 10(4):36 –44, 2011.

[26] F. Calabrese, J. Reades, and C. Ratti. Eigenplaces: Segmenting space through digital signatures. *IEEE Pervasive Computing*, 9(1):78–84, 2010.

[27] X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. *PVLDB*, pages 1136–1147, 2012.

[28] P. S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan. From taxi GPS traces to social and community dynamics: A survey. *ACM Computing Surveys*, pages 17:1–17:34, 2013.

[29] P. S. Castro, D. Zhang, and S. Li. Urban traffic modelling and prediction using large scale taxi GPS traces. In *Proceedings of International Conference on Pervasive Computing*, pages 57–72, 2012.

[30] A. Ceder and N. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986.

[31] V. Ceikute and C. Jensen. Routing service quality - local driver behavior versus routing services. In *Proceedings of IEEE International Conference on Mobile Data Management*, pages 97–106, 2013.

[32] S. Chawathe. Segment-based map matching. In *IEEE Symposium on Intelligent Vehicles*, pages 1190–1197, 2007.

[33] S. Chawla, Y. Zheng, and J. Hu. Inferring the root cause in road traffic anomalies. In *Proceedings of IEEE International Conference on Data Mining*, pages 141–150, 2012.

[34] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, and S. Li. Real-time detection of anomalous taxi trajectories from GPS traces. In *Proceeding of the 8th International ICST Conference on Mobile and Ubiquitous Systems*, pages 63–74, 2011.

[35] G. Chen, X. Jin, and J. Yang. Study on spatial and temporal mobility pattern of urban taxi services. In *Proceddings of the International Conference on Intelligent Systems and Knowledge Engineering*, pages 422–425, 2010.

[36] T. A. Chua. The planning of urban bus routes and frequencies: A survey. *Transportation*, 12(2):147–172, 1984.

[37] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[38] K. L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14:493–498, 1966.

[39] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, pages 32–40, New York, NY, USA, 2007. ACM.

[40] E. M. Daly, F. Lecue, and V. Bicer. Westland row why so slow? Fusing social media and linked data sources for understanding real-time traffic conditions. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 203–212, New York, NY, USA, 2013. ACM.

[41] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of ACM Conference on Hypertext and Hypermedia*, pages 35–44, 2010.

[42] B. Ding, J. X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In *Proceedings of the International Conference on Extending Database Technology: Advances in Database Technology*, pages 205–216, 2008.

[43] Y. Ding, S. Liu, J. Pu, and L. Ni. HUNTS: A trajectory recommendation system for effective and efficient hunting of taxi passengers. In *Proceedings of IEEE International Conference on Mobile Data Management*, volume 1, pages 107–116, June 2013.

[44] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006.

[45] C. Furtlehner, J.-M. Lasgouttes, and A. de La Fortelle. A belief propagation approach to traffic prediction using probe vehicles. In *Proceedings of IEEE International Conference on Intelligent Transportation Systems*, pages 1022–1027, Sept 2007.

[46] Y. Ge, C. Liu, H. Xiong, and J. Chen. A taxi business intelligence system. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 735–738, 2011.

[47] Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou. A taxi driving fraud detection system. In *Proceedings of the IEEE International Conference on Data Mining*, pages 181–190, 2011.

[48] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 899–908, 2010.

[49] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 899–908, 2010.

[50] Y. Ge, H. Xiong, Z.-H. Zhou, H. Ozdemir, J. Yu, and K. C. Lee. Top-EYE: top-k evolving trajectory outlier detection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1733–1736, 2010.

[51] 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 trajaectory data. *The VLDB Journal*, 20:695–719, 2011.

[52] J. Greenfeld. Matching GPS observations to locations on a digital map. In *Proceedings of the 81st Annual Meeting of the Transportation Research Board*, 2002.

[53] A. Gühnemann, R. Schäfer, and K. Thiessenhusen. Monitoring traffic and emissions by floating car data. In *Institute of transport studies Australia*, 2004.

[54] V. Guihaire and J.-K. Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251 – 1273, 2008.

[55] Y. Han and F. Moutarde. Analysis of large-scale traffic dynamics using non-negative tensor factorization. In *Proceedings of 19th World Congress on Intelligent Transport Systems (ITSwc'2012)*, pages 22–26, 2012.

[56] Y. Han and F. Moutarde. Statistical traffic state analysis in large-scale transportation networks using locality-preserving non-negative matrix factorization. *IET Intelligent Transport Systems,*, 7(3):283–295, 2013.

[57] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern RecognitionLetters*, 24(9-10):1641–1650, 2003.

[58] I. Hefez, Y. Kanza, and R. Levin. TARSIUS: A system for traffic-aware route search under conditions of uncertainty. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 517–520, 2011.

[59] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen. Estimating arterial traffic conditions using sparse probe data. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 929–936, 2010.

[60] A. V. Hill and W. C. Benton. Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. *The Journal of the Operational Research Society*, 43(4):343–351, 1992.

[61] A. Hofleitner, R. Herring, A. Bayen, Y. Han, F. Moutarde, and A. de La Fortelle. Large scale estimation of arterial traffic and structural analysis of traffic patterns using probe vehicles. In *91st Transportation Research Board Annual Meeting*, number 12-0598, 2012.

[62] H.-P. Hsieh, C.-T. Li, and S.-D. Lin. Exploiting large-scale check-in data to recommend time-sensitive routes. In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pages 55–62, 2012.

[63] H. Hu, Z. Wu, B. Mao, Y. Zhuang, J. Cao, and J. Pan. Pick-up tree based route recommendation from taxi trajectoriesp Tree Based Route Recommendation. In *Proceedings of the International Conference on Web-Age Information Management*, pages 471–483, 2012.

[64] X. Hu, S. Gao, Y. Chiu, and D. Lin. Modeling routing behavior for vacant taxi cabs in urban traffic netwroks. *Transportation Research Record*, pages 81–88, 2012.

[65] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589, Nov 2011.

[66] T. Hunter, P. Abbeel, and A. M. Bayen. The path inference filter: Model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, to appear, 2014.

[67] T. Hunter, P. Abbeel, R. Herring, and A. Bayen. Path and travel time inference from GPS probe vehicle data. In *Neural Information Processing Systems (NIPS)*, 2009.

[68] T. Hunter, A. Hofleitner, J. Reilly, W. Krichene, J. Thai, A. Kouvelas, P. Abbeel, and A. M. Bayen. Arriving on time: estimating travel time distributions on large-scale road networks. *CoRR*, abs/1302.6617, 2013.

[69] S. Jerby and A. Ceder. Optimal routing design for shuttle bus service. *Transportation Research Record: Journal of the Transportation Research Board*, 1971:14–22, 2006.

[70] B. Jiang, J. Yin, and S. Zhao. Characterizing human mobility patterns in a large street network. *Physical Review E*, 80:021136, 2009.

[71] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *Proceedings of the International Conference on Data Engineering*, page 10, 2006.

[72] Y. Kanza, R. Levin, E. Safra, and Y. Sagiv. Interactive route search in the presence of order constraints. *Proc. VLDB Endow.*, 3(1-2):117–128, 2010.

[73] S. Kim, S. Shekhar, and M. Min. Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1115–1129, 2008.

[74] R. Kohli and R. Krishnamurti. Probabilistic greedy heuristics for satisfiability problems.

[75] D. Krammer. Smart cities will need big data. *Physics Today*, 66(9):19–20, 2013.

[76] T. Kurashima and et al. Travel route recommendation using geotags in photo sharing sites. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 579–588, 2010.

[77] J. Lee, I. Shin, and G.-L. Park. Analysis of the passenger pick-up pattern for taxi location recommendation. In *Proceedings of the International Conference on Networked Computing and Advanced Information Management*, pages 199–204, 2008.

[78] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of 24th IEEE International Conference on Data Engineering*, pages 140 –149, 2008.

[79] R. Levin and Y. Kanza. TARS: traffic-aware route search. *GeoInformatica*, pages 1–40, 2013.

[80] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang. Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Proceedings of 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 63 –68, 2011.

[81] F. Li, D. Cheng, M. Had., G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *Proc. of SSTD*, pages 273–290, 2005.

[82] Q. Li, Z. Zeng, T. Zhang, J. Li, and Z. Wu. Path-finding through flexible hierarchical road networks: An experiential approach using taxi trajectory data. *International Journal of Applied Earth Observation and Geoinformation*, 13(1):110 – 119, 2011.

[83] Q. Li, Z. Zheng, B. Yang, and T. Zhang. Hierarchical route planning based on taxi GPS-trajectories. In *Proceedings of the International Conference on Geoinformatics*, pages 1–5, 2009.

[84] X. Li, M. Li, W. Shu, and M. Wu. A practical map-matching algorithm for GPS-based vehicular networks in Shanghai urban area. In *IET Conference on Wireless, Mobile and Sensor Networks*, pages 454–457, 2007.

[85] X. Li, X. Li, J. Han, S. Kim, and H. Gonzalez. ROAM: Rule- and motif-based anomaly detection in massive moving object data sets. In *Proceedings of 7th SIAM International Conference on Data Mining*, 2007.

[86] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data. In *Proceedings of 25th IEEE International Conference on Data Engineering*, pages 1319–1322, 2009.

[87] X. Li, G. Pan, Z. Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science in China*, 6(1):111–121, 2012.

[88] X. Liang, X. Zheng, W. Lv, T. Zhu, and K. Xu. The scaling of human mobility by taxis is exponential. *Physica A: Statistical Mechanics and its Applications*, 391(5):2135 – 2144, 2012.

[89] Z. Liao, Y. Yu, and B. Chen. Anomaly detection in GPS data based on visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 51 –58, 2010.

[90] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[91] M. Lippi, M. Bertini, and P. Frasconi. Collective traffic forecasting. In *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases: Part II*, pages 259–273, 2010.

[92] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1043–1051, 2013.

[93] C.-L. Liu, T.-W. Pai, C.-T. Chang, and C.-M. Hsieh. Path-planning algorithms for public transportation systems. In *Proceedings of IEEE Conference on Intelligent Transportation Systems*, pages 1061–1066, 2001.

[94] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data*, 6(1):3:1–3:39, 2012.

[95] L. Liu, C. Andris, and C. Ratti. Uncovering cabdrivers' behavior patterns from their digital traces. *Computers, Environment and Urban Systems*, 34(6):541 – 548, 2010.

[96] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li. Towards Mobility-based Clustering. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 919–928, 2010.

[97] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1010–1018, 2011.

[98] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse GPS traces for map inference: Comparison of approaches. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 669–677, 2012.

[99] X. Liu, Y. Zhu, Y. Wang, G. Forman, L. M. Ni, Y. Fang, and M. Li. Road recognition using coarse-grained vehicular traces. Technical report, HP Lab, 2012.

[100] Y. Liu, C. Kang, S. Gao, and Y. Xiao. Understanding intra-urban trip patterns from taxi trajectory data. *Journal of Geographical Systems*, 14(4):463–483, 2012.

[101] Y. Liu, F. Wang, Y. Xiao, and S. Gao. Urban land uses and traffic 'source-sink areas': Evidence from GPS-enabled taxi data in Shanghai. *Landscape and Urban Planning*, 106(1):73 – 87, 2012.

[102] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 352–361, New York, NY, USA, 2009. ACM.

[103] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 209–218, 2012.

[104] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zhang. Photo2trip: generating travel routes from geo-tagged photos for trip planning. In *Proceedings of the International Conference on Multimedia*, pages 143–152, 2010.

[105] Y. lun Chou. *Statistical analysis with business and economic applications*. Holt, Rinehart and Winston, 1969.

[106] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Proceedings of 29th IEEE International Conference on Data Engineering*, pages 410–421, 2013.

[107] B. M.H. and M. H.S. TRUST: A lisp program for the analysis of transit route configurations. *Transportation Research Record*, 1283:125–135, 1990.

[108] 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*, 14(3):1393–1402, Sept 2013.

[109] G. F. Newell. Some issues relating to the optimal design of bus routes. *Transportation Science*, 13(1):20–35, 1979.

[110] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the ACM Symposium on Applied Computing*, pages 863–868, 2008.

[111] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 334–343, New York, NY, USA, 2013. ACM.

[112] G. Pan, G. Qi, Z. Wu, D. Zhang, and S. Li. Land-use classification using taxi GPS traces. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):113–123, 2013.

[113] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng. On mining anomalous patterns in road traffic streams. In *Proceedings of the International Conference on Advanced Data Mining and Applications - Volume Part II*, pages 237–251, 2011.

[114] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4):42:1–42:32, Aug. 2013.

[115] S. Pattnaik, S. Mohan, and V. Tom. Urban bus transit route network design using genetic algorithm. *Journal of Transportation Engineering*, 124(4):368–375, 1998.

[116] C. Peng, X. Jin, K.-C. Wong, M. Shi, and P. Liò. Collective human mobility patter from taxi trips in urban Area. *PLoS ONE*, 7(4):e34487, 2012.

[117] S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti. Taxi-aware map: Identifying and predicting vacant taxis in the city. In *Proceedings of Ambient Intelligence*, pages 86–95, 2010.

[118] J. Powell, Y. Huang, F. Bastani, and M. Ji. Towards reducing taxicab cruising time using spatio-temporal profitability maps. In *Proceedings of the International Conference on Advances in Spatial and Temporal Databases*, volume 6849, pages 242–260. 2011.

[119] G. Qi, X. Li, S. Li, G. Pan, Z. Wang, and D. Zhang. Measuring social functions of city regions from large-scale taxi behaviors. In *Proceedings of 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 384 –388, 2011.

[120] G. Qi, G. Pan, S. Li, Z. Wu, D. Zhang, L. Sun, and L. Yang. How long a passenger waits for a vacant taxi – large-scale taxi trace mining for smart cities. In *2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and*

IEEE Cyber, Physical and Social Computing Green Computing and Communications (GreenCom), pages 1029–1036, Aug 2013.

[121] M. Rahmani and H. Koutsopoulos. Path inference of low-frequency gps probes for urban networks. In *Proceedings of International IEEE Conference on Intelligent Transportation Systems*, pages 1698–1701, Sept 2012.

[122] M. Rahmani and H. Koutsopoulos. *Path Inference of Sparse GPS Probes for Urban Networks: Methods and Applications*. PhD thesis, Stockholm, 2012. QS 2012.

[123] S. Rogers, P. Langley, and C. Wilson. Mining gps data to augment road models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 104–113, 1999.

[124] R.-P. Schäfer, K.-U. Thiessenhusen, and P. Wagner. A traffic information system by means of real-time floating-car data. In *World Congress on Intelligent Transport Systems*, 2002.

[125] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson. Mining GPS traces for map refinement. *Data Mining and Knowledge Discovery*, 9:59–87, 2004.

[126] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The optimal sequenced route query. *The VLDB Journal*, 17(4):765–787, 2008.

[127] R. R. Sillito and R. B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proceedings of British Machine Vision Conference*, 2008.

[128] W. Souffriau and P. Vansteenwegen. Tourist trip planning functionalities: State–of–the–art and future. In *Proceedings of Current Trends in Web Engineering*, pages 474–485, 2010.

[129] H. Su and S. Yu. Hybrid GA based online support vector machine model for short-term traffic flow forecasting. In *Proceedings of the International Conference on Advanced Parallel Processing Technologies*, pages 743–752, 2007.

[130] L. Sun, D. Zhang, C. Chen, P. S. Castro, S. Li, and Z. Wang. Real time anomalous trajectory detection and analysis. *Mobile Networks and Applications*, 18(3):341–356, 2013.

[131] W. Szeto and Y. Wu. A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong. *European Journal of Operational Research*, 209(2):141 – 155, 2011.

[132] T. Takayama, K. Matsumoto, and A. Kumagai. Waiting/cruising location recommendation for efficient taxi business. *International Journal of Systems Applications, Engineering & Development*, 5:224–236, 2011.

[133] J. Truscott and N. M. Ferguson. Evaluating the adequacy of gravity models as a description of human mobility for epidemic modelling. *PLoS Computational Biology*, 8(10):e1002699, 2012.

[134] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden. The city trip planner: An expert system for tourists. *Expert Systems with Applications*, 38(6):6540 – 6546, 2011.

[135] M. Veloso, S. Phithakkitnukoon, and C. Bento. Urban mobility study using taxi traces. In *Proceedings of the International Workshop on Trajectory Data Mining and Analysis*, pages 23–30, 2011.

[136] M. Veloso, S. Phithakkitnukoon, C. Bento, N. Fonseca, , and P. Olivier. Exploratory study of urban flow using taxi traces. In *The First Workshop on Pervasive Urban Applications*, 2011.

[137] F.-Y. Wang. Driving into the future with ITS. *IEEE Intelligent Systems*, 21(3):94–95, 2006.

[138] F.-Y. Wang. Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):630–638, 2010.

[139] H. Wang, H. Zou, Y. Yue, and Q. Li. Visualizing Hot Spot Analysis Result based on Mashup. In *Proceedings of the International Workshop on Location Based Social Networks*, pages 45–48, 2009.

[140] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *SIGCOMM Comput. Commun. Rev.*, 22(2):63–71, 1992.

[141] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering. Visual traffic jam analysis based on trajectory data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2159–2168, Dec 2013.

[142] H. Wen, Z. Hu, J. Guo, L. Zhu, and J. Sun. Operational analysis on beijing road network during the olympic games. *Journal of Transportation Systems Engineering and Information Technology*, 8(6):32–37, 2008.

[143] H. wen Chang, Y. chin Tai, and J. Y. jen Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18, 2010.

[144] K. Yamamoto, K. Uesugi, and T. Watanabe. Adaptive routing of cruising taxis by mutual exchange of pathways. In *Proceedings of the International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II*, pages 559–566, 2010.

[145] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. SeMiTri: A framework for semantic annotation of heterogeneous trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 259–270, New York, NY, USA, 2011. ACM.

[146] B. Yao, M. Tang, and F. Li. Multi-approximate-keyword routing in GIS data. In *Proc. of GIS*, pages 201–210, 2011.

[147] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 325–334, 2011.

[148] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and POIs. In *Proceedings of the ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 186–194, 2012.

[149] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 316–324, 2011.

[150] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):220–232, 2013.

[151] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-Drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108, 2010.

[152] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G. Sun. An interactive voting-based map matching algorithm. In *Proceedings of the International Conference on Mobile Data Management*, pages 43–52, 2010.

[153] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *Proceedings of the International Conference on Ubiquitous Computing*, pages 109–118, 2011.

[154] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2390–2403, 2013.

[155] Y. Yue, H. dong Wang, B. Hu, Q. quan Li, Y. guang Li, and A. G. Yeh. Exploratory calibration of a spatial interaction model using taxi GPS trajectories. *Computers, Environment and Urban Systems*, 36(2):140 – 153, 2012.

[156] Y. Yue, Y. Zhuang, Q. Li, and Q. Mao. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In *Proceedings of the International Conference on Geoinformatics*, pages 1–6, 2009.

[157] D. Zhang, B. Guo, and Z. Yu. The emergence of social and community intelligence. *Computer*, 44(7):21–28, 2011.

[158] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li. iBAT: detecting anomalous taxi trajectories from GPS traces. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, pages 99–108, 2011.

[159] 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 *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 9:1–9:14, New York, NY, USA, 2013. ACM.

[160] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011.

[161] W. Zhang, S. Li, and G. Pan. Mining the semantics of origin-destination flows using taxi traces. In *Proceedings of the Workshop of Ubiquitous Computing*, pages 943–949, 2012.

[162] Z. Zhang, D. Yang, T. Zhang, Q. He, and X. Lian. A study on the method for cleaning and repairing the probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):419–427, 2013.

[163] F. Zhao and X. Zeng. Optimization of transit route network, vehicle headways and timetables for large-scale transit networks. *European Journal of Operational Research*, 186(2):841 – 855, 2008.

[164] X. Zheng, X. Liang, and K. Xu. Where to wait for a taxi? In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pages 149–156, 2012.

[165] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, pages 89–98, 2011.

[166] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, pages 791–800, 2009.

[167] Y.-T. Zheng, Z.-J. Zha, and T.-S. Chua. Mining travel patterns from geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 3(3):56:1–56:18, 2012.

[168] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell. Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 322–331, 2008.

[169] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.

Appendix A

Appendix

Contents

A.1 Proof of the FIFO Property

Proof. For two users (A and B), if user A starts from v_i earlier than user B ($sT_A < sT_B$), then user A would arrive at v_{i+1} at least the same time as user B ($aT_A \leq aT_B$).

Since the taxi GPS traces can provide more information about potential routes, such as the length, estimated travel time, and popularity [31], we assume that we could recommend the user with the fastest driving routes at the given time of the day for any given pair of origin and destination (i.e. OD).

Consider the case in Figure A.1: both users (user A and B) drive from v_i to v_{i+1} through the same path (i.e. *Path 1*). User A would be ahead of user B since user A departures from v_i earlier than user B. Assuming that the traffic conditions in the former segment of *Path 1* might be better at the time when user B departures (i.e. sT_B), user B would take less time to complete this segment. Therefore, it is likely that user A and user B meet at a certain point (e.g. the point with star marker) in the route before reaching the destination. If so, user A and user B would arrive at v_{i+1} at the same time; otherwise, user A would arrive at v_{i+1} earlier than user B.

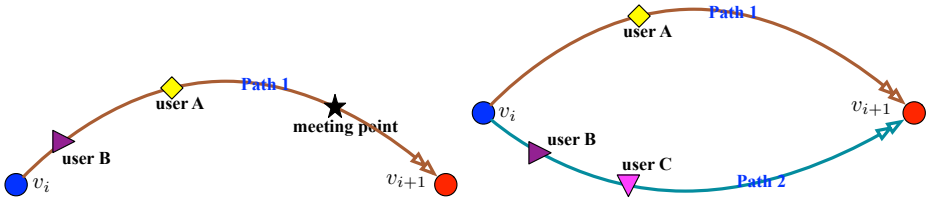


Figure A.1: User A and user B travel from v_i to v_{i+1} through the same path (a) and different paths (b).

Consider the case in Figure A.1: user A drives from v_i to v_{i+1} through *Path 1* at time sT_A, while user B drivers from v_i to v_{i+1} through *Path 2* at time sT_B. For comparison, we assume that a user C drives from v_i to v_{i+1} through *Path 2* at time sT_A. Since the fastest driving route between v_i and v_{i+1} when departing at the time sT_A is *Path 1*, we conclude that user A would arrive at v_{i+1} earlier than user C. Similar to the case in Figure A.1, as user C departures from v_i earlier than user B through the same path (i.e. *Path 2*), user C would arrive at v_{i+1} no later than user B. Therefore, user A would arrive at v_{i+1} earlier than user B. □

A.2 Edge Modelling

In this appendix, we first introduce the representation and indexing of the taxi GPS trajectory, then estimate the transit time between any two given venues (edge values), depending on the user's departure time.

A.2.1 Taxi Trajectory Representation and Indexing

Figure A.2 illustrates a taxi delivery trajectory. Each small circle point refers to a GPS sampling point; each triangle point refers to a venue in the targeted city. Note that the bigger circles (e.g., C_i) are the clusters of adjacent venues, constructed using the popular *mean-shift* algorithm [37, 76].

During a taxi ride, the driver may go through several venue clusters, and how long it needs to transit between any two passing-by clusters can be inferred. Thus, we represent the taxi trajectory as a sequence of venue clusters. For example, the taxi trajectory in Figure A.2 can be represented as:

$$\langle(\bar{T}_i, C_i), (\bar{T}_j, C_j), (\bar{T}_k, C_k)\rangle$$

where $\bar{T}_i$ is the average value of the sampling time between the taxi's first entry of and first exist from the venue cluster C_i. Consequently, from this trajectory, we can deduce that the

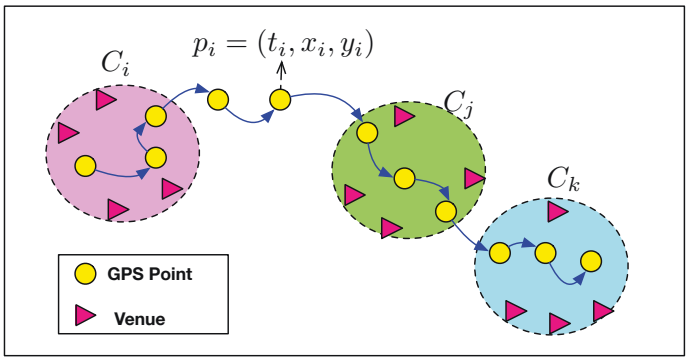


Figure A.2: Illustration of a taxi delivery trajectory.

transit time needed from C_i to C_j when leaving C_i at time $\bar{T}_i$ is $\bar{T}_j - \bar{T}_i$, from C_i to C_k at time $\bar{T}_i$ is $\bar{T}_k - \bar{T}_i$, and from C_j to C_k departing from C_j at time $\bar{T}_j$ is $\bar{T}_k - \bar{T}_j$. Thus, this trajectory can be further represented as three *quadruples* pairs:

$$1.\langle \lfloor \bar{T}_i \rfloor, C_i, C_j, \bar{T}_j - \bar{T}_i \rangle$$

$$2.\langle \lfloor \bar{T}_i \rfloor, C_i, C_k, \bar{T}_k - \bar{T}_i \rangle$$

$$3.\langle \lfloor \bar{T}_j \rfloor, C_j, C_k, \bar{T}_k - \bar{T}_j \rangle$$

Operation $\lfloor \cdot \rfloor$ is to get the corresponding time slot of the day for a given point of time. In particular, we divide a day into five time slots in the scope of this paper, as shown in Table A.1. Note that second and fifth time slots are the rush hours, and more transit time is needed usually. In total, there are $\frac{n(n-1)}{2}$ number of *quadruples* pairs for a given taxi trajectory, where n is the number of venue clusters that the taxi bypasses.

Table A.1: Divided time slots of a day.

Time slot	Specific time duration
1	00:00∼05:59
2	06:00∼07:59
3	08:00∼10:59
4	11:00∼16:59
5	17:00∼19:59
6	20:00∼23:59

We can derive hundreds of thousands of *quadruples* pairs from the taxi GPS trace dataset using this representation. The *quadruples* pairs with the same first three elements (i.e. time slot, the departure cluster, and the arrival cluster) will have the same Id.

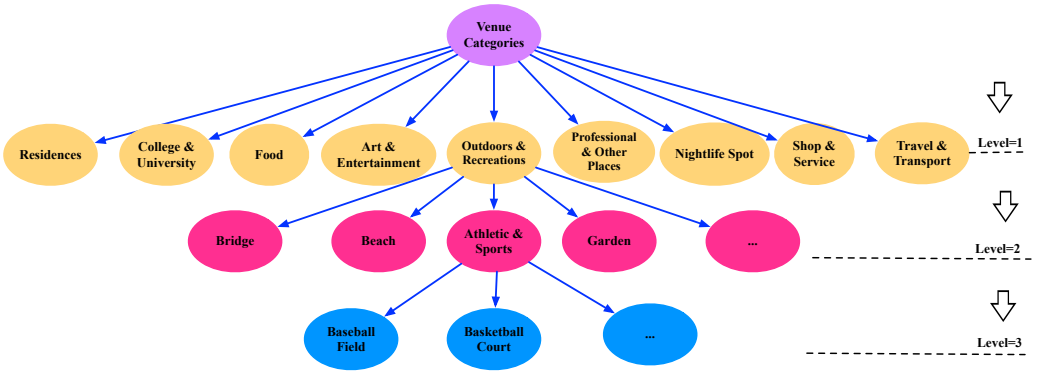


Figure A.3: Ontology structure of venue categories.

A.2.2 Transit Time Estimation

Since the best way to move from one venue to anther inside the same cluster is by walking, the transit time between within-cluster venues is simply estimated by the ratio between their spherical distance and the walking speed (i.e. 4.5 *km/h*). For venues in different clusters (e.g. C_i and C_j), the transit time depends on the user's departure time (e.g. 9:00 AM), and it can be estimated by averaging the fourth element in all *quadruples* pairs who has the index of $\langle 3, C_i, C_j \rangle$. Considering the difference in driving speed between a traveller and local taxi drivers, and the time spent on parking, we multiply the transit time estimated from taxi trajectories with a constant c_1 (=1.3) in the scope of this paper, and add a constant parking time c_2 (=3 minutes).

A.3 Venue Category Encoding and Retrieving

In this appendix, we propose a simple venue category encoding mechanism to retrieve user-preferred venues from all city venues efficiently, according to the user-preferred venue categories (CAT_u) in the itinerary query (IQ). We first briefly introduce how Foursquare organizes venue category, and then describe our proposed solution.

In Foursquare, venues are organized in a three-level ontology structure (Figure A.3). It has 9 categories on the first level and 412 sub-/sub-subcategories (i.e. level 2 and level 3). Further analysis showed that, in Foursquare, each venue is often marked with more than one category labels distributed across different levels. For example, the average number of labels per venue in San Francisco is 1.421. In addition, when planning for an actual visit, users may describe a venue at different level of details as their knowledge, background, and experiences vary. For instance, when a user specifies the "Food" category, not only venues marked with "Food", but also those associated with the child category labels, e.g. "New

American Restaurant" and "Bakery" should be returned.

Based on these two facts, we propose a simple venue encoding mechanism that assigns a unique number to each venue category label with information about its superordinate integrated. Specifically, we use a 6-digit number to encode a venue category label. For Level 1 categories, only the first two digits are used and the rest digits are set to zero. For example, 05—00—00 refers to the "Outdoors & Recreations" category label. For a Level 2 category label (sub-category), the first two digits refers to its parent category, the middle two digits encodes its position among all siblings, and the last two digits remain zero. Taking 05—01—00 as an example, it refers to the "Bridge" label which is a child of "Outdoors & Recreations". Similarly, for Level 3 category labels (sub-sub-category), all six digits are non-zeros. The former four digits denote to which the refer to its category and sub-category it belongs, respectively.

With this coding system, the user-preferred venues can be retrieved through the following four steps.

Step 1: For each user-specified venue category, we encode it into a 6-digit number. The number is denoted as Num_i.

Step 2: For each venue in the targeted city, we encode its venue category labels into corresponding 6-digit numbers. Note that one venue may be assigned to more than one label numbers.

Step 3: Given the encoded number for the user-specified venue category (i.e. Num_i, the output of Step 1), we retrieve the user-preferred venues from all venues in the targeted city. Specifically,

— if the user-specified venue category has no child labels (i.e. the last two digits of Num_i are non-zeros), venues in the targeted city with the exactly same encoded number as Num_i would be marked as user-preferred venues;

— otherwise, venues that contain the encoded numbers Num_i or any of child labels of Num_i should be retrieved. For example, if Num_i=01—01—00, venues contain encoded numbers from 01—01—00 to 01—01—99 are all returned.

Step 4: **Repeat** *Step 1∼3* **until** all user-specified venue category labels are checked. Venues retrieved in each round will be unified to form the final output.

List of figures

List of tables